Chapter 4: Routing in Wireless Sensor Networks

4.1 Introduction to Routing in Wireless Sensor Networks

Wireless Sensor Networks (WSNs) is a distributed wireless ad-hoc network comprising of a number of sensor nodes, which are used for sensing the environment about the climatic changes, seismic activities, movement of enemy troops in war zone, industrial monitoring and control, etc. In wireless sensor networks, the transmission range of a tiny sensor node is limited. However, the sensed information from such sensor node normally has to be transmitted to and processed at the base station or a control center (also called as sink), which could be far away from the sensor node and out of the transmission range of the sensor node. In other words, the data may have to travel multiple hops before reaching the sink. Similarly, the query commands issued by users or the sink may have to go through multiple hops through the network to obtain some particular information which is collected by different sensor nodes at different locations. Therefore, it is essential to deploy efficient scheme in the wireless sensor network to select paths going through multiple hops and forward data from source to destination, which is a major functionality of the routing process.

Why is routing important to Wireless Sensor Networks?

Routing plays an important role in wired networks, wireless networks, and mobile ad-hoc networks (MANETs), which attracts a large number of studies in the past. However, due to the
unique constraints and application requirements in WSNs, the routing schemes developed for the Internet and MANET are often not feasible or cannot produce promising performance as needed in wireless sensor networks. For example, most routing protocols of Internet assume highly reliable wired link with very low bit error rates while MANET routing solutions are normally optimized for highly mobile nodes with symmetric links between neighbors. However, these assumptions are not true for wireless sensor networks. While facing the challenges arisen from the wireless environment and links as MANET or wireless LAN, routing schemes in WSNs also have to consider unique issues including the limited resources (such as energy, bandwidth and computing), lossy wireless link and fault tolerance, data aggregation and data reporting, node deployment, scalability, coverage, network dynamics and node/link heterogeneity [Njamal04].

1. Limited resources in WSNs

Since the sensor nodes are normally powered by batteries which are not easy to replace or recharge, energy of each network node in WSNs is limited. Energy efficient routing is one critical design criteria for WSNs because power failure of a sensor node not only affects the node itself but also its ability to forward packets on behalf of others and thus the overall network lifetime. In addition, if the data is lost due to the failure in the routing scheme or suboptimal route selection, the retransmission consumes extra energy and results in additional delay to the network, which also wastes the limited available bandwidth in the sensor network.

Energy efficiency is one of the most common concerns to be effectively addressed in the design of routing schemes for WSNs. Many research efforts have been devoted to developing energy-aware routing protocols for WSNs. However, in general, energy consumption is not a concern at all in traditional Internet routing protocols.
Similarly, the limited resources such as bandwidth, memory, and computing deserve carefully consideration in the design of WSNs routing protocols. For example, with limited memory space, sensor node cannot store the whole topology information of a large-scale network for routing decision. Neither a large routing table is feasible in the sensor node.

2. Fault Tolerance

Unlike the traditional wired network, the nodes and the links in sensor networks are more prone to errors or failure. The sensor node may be out of work due to lack of power or physical damage. The wireless link can be broken by the failure of sensor nodes, environmental interference or obstacles. Thus, the routing procedures in WSNs should function effectively even when there are node or link failures in the network. To achieve this, routing schemes in WSNs may have to find alternative paths dynamically or take advantage of the redundancy in the network to tolerate the unpredictable failures in the sensor networks.

3. Data Reporting and Aggregation

The data sensed by the sensor nodes in WSNs needs to be reported to the users of the system by transmitting the data to the base station. Data reporting can be done in different approaches such as time-driven, event-driven, query-driven, and hybrid of all the three methods. In the time-driven approach, the nodes periodically report the data after certain interval of time. In the event-driven approach, whenever an event occurs in the environment, the nodes report the information
associated with that event to the base station. In the query-driven approach, the base station issues a query to some nodes in the network and expects the corresponding nodes to collect the necessary information delivered to the base station. The different data reporting approaches have different advantages and disadvantages, which requires different routing schemes to address the needs.

In addition, sensor nodes in WSNs may produce a significant amount of redundant data. For example, multiple sensors may report the same information or different aspects of the same events occurring in the vicinity of a particular location. To reduce the number of transmissions and related resources consumption, aggregating similar packets from multiple nodes based on some certain criteria should be considered. The aggregation techniques include duplicate suppression, signal processing, data fusion, and so on.

4. Node Deployment

Node deployment in WSN can be done in two ways: randomly and manually. When the node deployment is done in random fashion, the nodes in the network form a wireless ad-hoc structure. Thus, the routing protocols deployed in the network have to self-learn the topology information and dynamically forward data with energy-efficient operation. When the node deployment is done manually, the routes for transmitting the data can be calculated optimally to achieve some goals using offline algorithm. In other words, the route can be predefined. However, in case of the topology changes due to node/link failures, dynamical routing schemes are still necessary in manually deployed WSNs.
5. **Scalability and Coverage**

In many WSN applications, the number of deployed nodes in the physical area to be monitored can be significantly large. The routing scheme employed in WSNs should be scalable and working towards similar efficiency even though the network size is large. The number of events or the data information to be delivered in WSNs can be enormous in particular instance of time, which also requires great scalability from the routing process in the network.

The transmission and sense range of sensor nodes in WSNs are constrained by the physical size and capability of the nodes, which are normally small comparing with the physical area to cover by the network. Efficient routing schemes have to consider the particular requirements from the specified applications to maintain necessary network connectivity and coverage.

6. **Network Dynamics and Heterogeneity**

Some nodes in WSNs may contain the devices, which make the nodes move from one position to another position. Due to the movement of the nodes in the network, the network topology and connectivity will change. The number of the nodes in the network and network connectivity may also change from time to time due to the node/link failures. Such network dynamics have to be considered together with the dynamic events in the design of routing protocols for WSN.

In many WSN applications the nodes and link between any two nodes are assumed as homogenous. But in reality, the node and the links between any two nodes are not always homogenous. The amounts of energy, transmission range (i.e., the maximum distance which the node can directly transmit the data to), memory and processing capability may vary among the
nodes and during the lifetime of the nodes. For example, symmetric link is the default feature for the wired networks such as Ethernet or optical networking. However, this symmetric feature does not hold true for the wireless links in sensor networks, which necessitates different routing discovery process for sensor networks.

Sensor nodes are constrained in resources such as energy, bandwidth, memory, and computing capability. Such constraints combined with the aforementioned challenging issues necessitate the invention and development of new routing solutions for WSNs.

4.2 Layout for the book Chapter

With the challenging issues in mind, we first introduce some general concepts adopted in the design of routing protocols for sensor networks in next section. Those concepts include flooding, gossiping and ideal dissemination. Next, the classification of routing schemes developed in the literature is described. Then, this chapter highlights several typical routing protocols including Sensor Protocols for Information via Negotiation (SPIN), Directed Diffusion, Low Energy Adaptive Clustering Hierarchical routing protocol (LEACH), Threshold Sensitive Energy Efficient Sensor Network (TEEN), Geographical and Energy Aware Routing (GEAR), and multi-path routing in sensor networks.
4.3 Classification of Routing Protocols in WSNs [Njamal04]

Routing in sensor networks is very challenging and different from contemporary wired/wireless networks such as Ethernet and mobile ad hoc networks [Rwheinzelman99] [Jkulik02]. The popular IP-based protocols cannot be applied to sensor networks because it is infeasible to build a global addressing scheme for the deployment and maintenance of thousands of tiny sensor nodes with limited resources. Many new algorithms have hence been developed for routing and forwarding data in sensor networks.

**Proactive and Reactive Routing**

The routing protocols can be *proactive*, *reactive* or *hybrid* depending on how the route is found. Proactive protocols attempt to continuously evaluate the routes within the network so that all routes are computed before they are needed. In other words, when a packet needs to be forwarded, the route is already available and can be immediately adopted. Reactive protocols, on the other hand, invoke a route determination procedure on demand only. Hence, some sort of search procedure has to be employed to identify a route prior to data forwarding. Hybrid routing protocols attempt to integrate both the above two ideas and take the advantages from the above discussed ideas.

The advantage of the proactive schemes is that there is little or no delay to determine a route whenever a route is needed. On the other hand, reactive protocols have to start a route discovery process to identify proper path information when a route is needed, which means the time for determining a route can be quite significant. This leads to the increased latency for the packet delivery and may not be applicable to real-time communication. However, proactive schemes
are likewise not appropriate for the ad hoc networking environment whereas network topology changes fast and constantly. Such network dynamics may result in continuous route evaluation and maintenance which use a large portion of the network resources. Particularly, when the changes are more frequent than the route requests, the routing information from the continuous evaluation process may not be necessary and never be used.

**Flat and Hierarchical Routing**

Based on the network structure, routing protocols in WSNs can also be broadly divided into *flat* routing, and *hierarchical* routing. In *flat* routing schemes, equal roles and functionality are typically assigned to each node. *Flat* routing protocols distribute information as needed to any node that can be reached or receive information. *Hierarchical* routing protocols often group nodes together by function into a hierarchy or cluster. By assigning different roles for different type of nodes or performing traffic aggregation to reduce redundancy, a hierarchical protocol allows WSNs to make best use of the heterogeneous nodes capability. In many hierarchical routing protocols, each cluster designates a cluster-head node to aggregate and relay inter-cluster traffic. These cluster-head nodes may become the bottleneck, potentially resulting in network congestion and single point of failure. In addition, maintaining the hierarchy or cluster can be costly in terms of energy or bandwidth consumption for small to moderately sized WSNs, which indicates flat schemes are favorable in this case. On the other hand, *hierarchical* routing protocols are often better suited for large WSNs due to its scalability.

In fact, there are many other ways to classify the routing protocols based on different criteria such as protocol operation, network flow, energy and QoS awareness. In the rest of this chapter,
we will focus three typical categories: data-centric protocols, hierarchical routing, location-based routing protocols, and multi-path routing in WSNs.

4.4 Data-centric Routing Protocols in WSNs

In many applications of wireless sensor networks, the physical area covered by the sensors and the number of deployed sensor nodes can be enormous. Typically, the meaningful data traffic is generated due to the sensors’ response to a query from the users (e.g., sink, base station) or actively reporting a detected event. In either case, multiple sensors having the interested data will initiate the data transmission, which may result in significant redundancy and resources wastage. Certainly, if sensor nodes are as reliable as the IP routers and globally addressable, the redundancy issue may be trivial to resolve. However, it is infeasible (if not possible) to assign a unique identifier to each sensor node and make each sensor node globally addressable like the IP router in the Internet. Accordingly, data-centric routing protocols are proposed for wireless sensor networks. In data-centric routing scheme, the sink sends queries to specific regions and waits for answers from the selected regions. These queries are described in high level language. Since data is being requested through queries, attribute-based naming is necessary to specify the properties of the interested data.

The Sensor Protocols for Information via Negotiation) (SPIN) and Directed Diffusion are among the earliest work on data-centric protocol [Jkulik02] [Rwheinzelman99] [Cintanagonwiwat00], which consider data negotiation between nodes in order to eliminate redundant data and save energy. These two protocols motivated the design of many other protocols that followed similar
concepts. Examples of such data-centric routing protocols are Rumor Routing [Bdavid02], Minimum Cost Forwarding Algorithm (MCFA) [Fye01], Gradient Based Routing (GBR) [Cschurgers01], COUGAR [Yyao02], Energy Aware Routing [Rcshah02], etc. Rumor Routing is mainly intended for contexts in which geographic routing criteria is not applicable. Rumor routing protocol uses a set of long-lived agents to create paths that are directed toward the events they encounter. MCFA utilizes the information about the direction of routing. MCFA gets rid of the unique ID and routing table, instead each node in MCFA maintains the least cost estimate from the node itself to the base station. COUGAR considers the whole network as a distributed database system and uses declarative queries for query processing from network layer functions. COUGAR also utilizes in-network data aggregation for more energy savings. Energy Aware Routing uses a set of sub-optimal paths occasionally to increase the lifetime of the network. These paths are chosen by means of a probability function, which depends on the energy consumption of each path.

In the rest of the section, we will describe three typical data dissemination schemes including flooding/gossiping, SPIN and Directed Diffusion in details with a focus on the key ideas and performance issues.

4.4.1 Flooding and Gossiping

Flooding is a classical and straightforward mechanism to disseminate data in WSNs, which takes advantage of the broadcasting nature of the wireless medium. To deliver a particular packet from the source to the destination node with flooding, the source node broadcasts the data to all the neighbors. Upon receiving the packet, each neighbor will broadcast a copy of the packet to its
neighbors. This process continues until the packet arrives at the destination or the packet is dropped. Flooding is very easy to implement, but it has a major drawback in increasing the network load with redundant traffic. In classic flooding, a node may blindly broadcast whatever it receives, regardless of whether or not the neighbor has already received a copy from another source. This leads to the *implosion* problem [Rwheinzelman99] [Jkulik02]. Figure 4.1 shows the implosion problem where the same message goes to node C, from nodes A and B, thereby creating redundancy [Rwheinzelman99]. In Figure 4.1 a WSN with four nodes A, B, C and D is shown. Assume that the data needs to be sent from node D to node C using flooding. Node D broadcasts the data \((a)\) to its neighbors which are nodes A, B. The nodes A and B forward the same data \((a)\) to node C. Here the issue is that node C receives the same data twice. This *implosion* results in multiple copies of the same data packet floating around the network and a node may receive multiple copies of the data information.

![Figure 4.1 Implosion Problem](image_url)

Sensor nodes often cover overlapping geographic areas and gather overlapping pieces of event data. The sensed data received by the neighbors of the nodes would contain some part of the data which is redundant, which is known as *overlap* [Rwheinzelman99] [Jkulik02]. Figure 4.2 shows an example of the overlapping issue. Node A in Figure 4.2 senses the data in the regions q and r.
Similarly node B senses the data in the regions r and s. Assume that the data sensed in the regions q and r is (q, r) and the data sensed in the regions r and s is (r, s). After sensing the data, the nodes A and B send the data (q, r) and (r, s) to the node C. Obviously, the redundant copy of data (r) received at destination node C is not necessary.

![Overlap Region](image)

**Figure 4.2** Example of Overlap Region [Rwheinzelman99] [Jkulik02]

The *implosion* and *overlap* issues lead to additional traffic in the network which is not necessary. The limited resources such as energy and bandwidth in WSNs will be wasted by this naïve flooding process. Hence, many studies have brought up techniques such as probability and packet ID to control the redundancies generated from the flooding process. For example, after assigning a unique ID for the packet, sensor node can remember the IDs for the packets it broadcasted earlier. Then the node can ignore the broadcast requests when it sees the same packet ID again. Similarly, a node may ignore a broadcast request according to a certain probability distribution. However, such techniques still cannot totally eliminate the flooding redundancies and may have considerable negative impacts on the network performance. To avoid the problem of flooding redundancy, gossiping takes a step further by just selecting one
random node to forward the packet rather than broadcasting. In other words, in gossiping the receiving node sends the packet to a randomly selected neighbor. The received packet is forwarded to another next-hop neighbor, which is also picked randomly to forward the packet and so on. However, the random selection of next-hop can cause delays in propagation of data through the network.

Gossiping can suppress the implosion issues in the flooding scheme. However, both flooding and gossiping routing schemes actually do nothing in reducing the redundant reports and packets in the overlap scenario, where an event is detected by all the sensors in the region or multiple sensors in the region reply with similar information on a particular query.

**Ideal Dissemination**

To disseminate data in WSNs, ideally, sensor nodes send observed data along the optimal routing path (taking into consideration the number of hops, time to transmit the data, energy consumption) and the intended nodes receive each piece of distinct data only once. This phenomenon is called *ideal dissemination* in [Rwheinzelman99] [Jkulik02]. For example, assume node A initially possesses data \((a, c)\), and node B only possesses data \((c)\) as shown in Figure 4.3. To efficiently disseminate the data throughout the network node D in Figure 4.3, uses *ideal dissemination* scheme and will only transmit the data in the order shown in the boxed number. First, node D delivers data \((a,c)\) and \((a)\) to node A and B, respectively, while node B sends data \((c)\) to node D. Then, either node B or node C sends data \((a)\) to node D. *Ideal*
dissemination does not waste energy on transmitting and receiving useless data. Of course, in a real distributed ad-hoc sensor network, it is extremely challenging (if not possible) to achieve ideal dissemination.

**Figure 4.3** Example of Ideal Dissemination [Rwheinzelman99] [Jkulik02]

4.4.2 Sensor Protocols for Information via Negotiation (SPIN) [Jkulik02] [Rwheinzelman99]

To overcome the aforementioned implosion and overlap issues, a family of adaptive protocols, called SPIN (Sensor Protocols for Information via Negotiation) were proposed to use negotiations for diffusing data in WSNs [Jkulik02] [Rwheinzelman99]. SPIN uses meta-data for describing the sensed data. The meta-data are exchanged among sensors via data advertisement. Upon obtaining new data, each node advertises the availability of new data information to its neighbors. The interested neighbor node, who also would like to possess the new data, can send a request message to the advertiser. Then the advertiser will reply with the data to the requested nodes. Unlike the classical flooding and gossiping protocols which are blind to the resources consumption in the network, SPIN uses resource manager to become resource-aware and resource-adaptive in the process of data dissemination. The major goal of SPIN's meta-data negotiation is to resolve the classic problems of flooding such as redundant information passing,
overlapping of sensing areas and resource blindness and thus, achieving better energy efficiency.

**Design of SPIN**

The design of SPIN is motivated by the Application Level Framing [Dclark90] (ALF). Using ALF, the network protocols choose transmission units that are meaningful to applications. In other words, the packetization is best done in terms of application data units, Hence, SPIN design meta-data to ensure common naming data in both the transmission protocol and the application. Instead of sending the actual data, sensor nodes send meta-data to interested neighbors in the form of advertisement. The metadata must be smaller than the actual data for SPIN to be energy efficient. If the actual data is distinguishable, then the corresponding meta-data should also be distinguishable. Similarly, two pieces of data which happen to be indistinguishable should have the same meta-data. Generally, the format of meta-data depends on the particular application [Rwheinzelman99].

Another important aspect of SPIN is to use resource manager to monitor the available resources in the node and make corresponding decision whether to participate in particular data dissemination. Applications probe the resource manager before transmitting or processing data. The nodes using SPIN calculate the energy and resources available by means of polling the resource system. Hence, the routing decisions in SPIN are made by combining knowledge of not only topology information, but also, application data layout and the status of resources available at each node.
Messages in SPIN

There are three different types of messages in SPIN which are new data advertisement (ADV), data request (REQ), and message DATA. When a node has data message to share, the node can advertise this fact by transmitting an ADV packet containing the meta-data of the message. A node, which is interested in the details of the message based on the received meta-data packet, can send an REQ packet to the advertiser. Then, the requesting node will receive the DATA message containing actual details of the message with a meta-data header from the advertiser.

Different types of SPIN

The above SPIN philosophy is tuned to accommodate different WSN application and network scenarios. Four SPIN protocols are proposed in [Rwheinzelman99]: SPIN-PP, SPIN-BC, SPIN-EC, and SPIN-RL.

- SPIN-PP – for point-to-point transmission media. Assume that there is plentiful energy and packets are never lost in the network.
- SPIN-BC – for broadcast transmission media. Assume that there is plentiful energy and packets are never lost in the network.
- SPIN-EC – an energy-conserving version of SPIN-PP.
- SPIN-PP – a reliable version of SPIN-BC.

SPIN-PP

The SPIN-PP employs three stages of message exchange for networks using point-to-point transmission media which allow nodes A and B to communicate exclusively with each other without interfering with other nodes. The three stages are corresponding to the three messages
described above. The protocol starts with a node sending an ADV message to its neighbors to advertise the data it intends to disseminate. In next stage, the neighbors check whether they are interested in the advertised data after receiving the ADV packet. If a node determines to possess a copy of the data, the node sends back an REQ message to the node that sent the ADV message. Then in the final stage, the actual data in the form of DATA message is delivered from the advertiser to the requester. Based on the received new DATA message and its own data in the memory, a node could perform some aggregation or redundancy reducing processes prior to re-advertising the aggregated meta-data to the neighbors.

SPIN-EC

SPIN-EC adds an energy conserving scheme to the SPIN-PP protocol. When a node receives a new data, it will consult the resource manager before initiating the SPIN protocol and advertising the new meta-data. The SPIN protocol will be started if and only if it turns out that the node has enough energy to complete all the stages of the protocol. Otherwise, it simply refrains from participating in the protocol. Similarly, upon receiving an advertisement, a node does not send out a request if it does not have enough energy to transmit the request and receive the corresponding data.

SPIN-BC

SPIN-BC is developed for broadcast transmission media. In SPIN-BC, the nodes use a single channel to broadcast the data to all the nodes in the receiving range. SPIN-BC employs the one-to-many communication scheme for delivering the same data to multiple sensor nodes in one transmission. Similar to the SPIN-PP, SPIN-BC also operates in three stages. There are three
primary aspects in which SPIN-BC is different from the SPIN-PP.

1. In SPIN-PP, one transmission can only target on one specific node. Hence, a node has to send the advertise meta-data to every neighbor in separate transmission. However, taking advantage of the broadcast transmission media, every node within the transmission range could receive the same data in SPIN-BC.

2. Unlike SPIN-PP, SPIN-BC does not allow nodes to respond to the ADV packets immediately. In SPIN-BC, upon receiving the ADV packet, the nodes check whether they already possess the data advertised. If a node does not possess the data, the node sets a random timer. When the timer expires, the node broadcasts a REQ message to the original advertiser if the node does not receive the advertised data yet. Then the node advertises the meta-data will respond to the REQ with the DATA message. When nodes other than the original advertiser receive the REQ, they cancel their own request timers to avoid redundant copies of the same request.

3. SPIN-BC will broadcast the DATA message only once and will not respond to multiple requests for the same data.

**SPIN-RL**

To handle the lossy link in WSNs, the SPIN-RL protocol makes two adjustments on SPIN-BC for reliable transmission. First, nodes employing SPIN-RL protocol keep track of all the advertisements that are received. If a node does not receive the data within a particular period of time after sending out the request, the node consults the track of all advertisements received and sends another request to a randomly selected advertiser with same piece of data. Second, nodes in SPIN-RL limit the frequency with which they will resend data to the neighbors. After the node sends the requested data say (a) to other nodes, the node waits for some period of time before responding to any further requests demanding the same piece of data (a).
Evaluating SPIN Protocols [Rwheinzelman99] [Jkulik02]

Using meta-data names, nodes in SPIN negotiate with each other about the necessary data exchange. These negotiations ensure that nodes only transmit data when necessary and energy is not wasted on useless or redundant transmissions. With the resource manager, each node is aware of the available resources and is able to cut back on the activities to enlarge the lifetime of the network.

Table 4.1 shows the related parameters in the simulation with a randomly generated 25-node network [Jkulik02]. Each node in the network is initialized with 3 data item, randomly chosen from a set of 25 possible data items. No network loss and queuing delay is considered.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>25</td>
</tr>
<tr>
<td>Edges</td>
<td>59</td>
</tr>
<tr>
<td>Average degree</td>
<td>4.7 neighbors</td>
</tr>
<tr>
<td>Diameter</td>
<td>8 hops</td>
</tr>
<tr>
<td>Average shortest path</td>
<td>3.2 hops</td>
</tr>
<tr>
<td>Antenna reach</td>
<td>10m</td>
</tr>
<tr>
<td>Radio propagation delay</td>
<td>3x8 m/s</td>
</tr>
<tr>
<td>Processing delay</td>
<td>5-10ms</td>
</tr>
<tr>
<td>Radio speed</td>
<td>1 Mbps</td>
</tr>
<tr>
<td>Transmit cost</td>
<td>600 mW</td>
</tr>
<tr>
<td>Receive cost</td>
<td>200 mW</td>
</tr>
<tr>
<td>Data size</td>
<td>500 bytes</td>
</tr>
<tr>
<td>Meta-data size</td>
<td>16 bytes</td>
</tr>
</tbody>
</table>

*Table 4.1* Simulation test bed for SPIN [Jkulik02]
Table 4.2 shows the simulation results from SPIN-PP in which the *ideal dissemination* scheme is used as the baseline. Comparing the *flooding* and *gossiping* schemes, SPIN-PP consumes much less energy SPIN-PP uses approximately a factor of 3.5 less energy than flooding. This is partially due to the fact flooding and gossiping schemes introduce much redundancy data. As shown in Table 4.2, simulation shows 77% of the transmitted DATA messages are redundant in flooding scheme and 96% of them are redundant in gossiping scheme. Note that SPIN-PP also introduces limited overhead traffic such as the ADV and REQ packets to the network.

The convergence time is defined as the time it takes to ensure all the nodes in the network receive the intended data. SPIN-PP takes 80 ms longer to converge than flooding, whereas flooding takes only 10 ms longer to converge than *ideal dissemination* scheme. Although it appears that SPIN-PP performs much worse than the flooding scheme in terms of the convergence time, this increase is actually a constant amount, regardless of the length of the simulation. Thus, for longer simulations, the increase in convergence time for the SPIN-PP protocol will be negligible [Rwheinzelman99].

<table>
<thead>
<tr>
<th>Performance Relative to Ideal</th>
<th>SPIN</th>
<th>FLOODING</th>
<th>GOSSIPING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase in Energy Dissipation</td>
<td>0.45 J</td>
<td>6.3 J</td>
<td>44.1 J</td>
</tr>
<tr>
<td>Increase in Convergence Time</td>
<td>90 ms</td>
<td>10 ms</td>
<td>3025 ms</td>
</tr>
<tr>
<td>Slope of Energy Dissipation vs. Node Degree Correlation Line</td>
<td>1.25x</td>
<td>5x</td>
<td>25x</td>
</tr>
<tr>
<td>% of Total Data Messages that are Redundant</td>
<td>0</td>
<td>77%</td>
<td>96%</td>
</tr>
</tbody>
</table>

Table 4.2 Results for simulations of SPIN-PP protocol [Rwheinzelman99]
Other simulation and analysis results also indicate that SPIN-EC distribute 60% more data per unit energy than the flooding scheme. SPIN-PP and SPIN-EC outperform the gossiping scheme and come close to the ideal dissemination protocol. In addition, SPIN-BC and SPIN-RL are able to use one-to-many communications exclusively, while still acquiring data faster and using less energy than the flooding scheme. SPIN-RL can efficiently handle packet loss and dissipate twice the amount of data per unit energy as the flooding scheme.

4.4.3 Directed Diffusion [Cintanagonwiwat00]

Directed Diffusion differs from SPIN in terms of the way the data transmission is initiated. The basic idea of Directed Diffusion is to diffuse data through sensor nodes by using a naming scheme for the data. With the naming scheme, the sink can issue query to the sensor nodes regarding the data the sink is interested in. Then the corresponding sensor nodes reply the necessary information to the sink. To achieve this, Direct Diffusion assigns attribute-value pairs to the data and queries in an on demand basis. In order to issue a query indicating the type of data the sink is looking for, an interest is defined using the attribute-value pairs such as name of objects, geographical area, duration, interval, etc. The sink disseminates the interest through its neighbors. The interest is cached in the sensor nodes. Whenever a node receives data, the node can compare the received data with the values of the interest. If there is a match, the node will establish paths to the sink from which the node receives the interest. These paths are known as events. Then the sink can choose paths to resend the interest and expect the sensor node to reply data back to the sink.
Directed diffusion consists of several elements.
- Data: is named using attribute-value pairs.
- Interest: is a sensing task for named data.
- Gradient: is a reply link to a neighbor from which the interest is received.
- Events: start flowing towards the originators of interests along multiple paths.
- Reinforcement: is a mechanism for the sink to select paths receiving the sensed data.

Naming

The task descriptions in Directed Diffusion are named by several attribute-value pairs that identify the specific task. For example a task of tracing animals may be described as:

```
Type = animal       //detect animals
Interval=0.5s      //send back events every 0.5s
Timestamp = 02:02:19  //interest generated time
ExpiresAt = 02:12:19  //not interested in this afterwards
RECT= [-100, 100, 200, 400] // sensors within the specified region perform the task
```

Since a task description is called an interest, which specifies an interest in a particular kind of data which matches the attributes in the task description. Similarly, the data sent in response to the interest is also named using attribute-value pairs. For example, a sensor that detects animals in the specified region may create a return data message (or reply) as following:

```
Type = animal       //detect animals
Instance = cow     //instance type
Location = [122, 210]  //node location
Confidence = 0.90   //confidence in the match
Timestamp = 02:02:20  //event occurring time
```
How to choose the naming schemes such as attributes and value ranges heavily depends on the applications the sensor networks are deployed for. The choice has to be done carefully, since the choice of the naming scheme can affect the expressivity of tasks and many impact performance of the network.

*Interest Propagation and Gradient Establishment*

An interest message is a query which specifies what the sink wants the sensors to report. Assume that there are five fields in an *interest* as in previous example: Type, Interval, Timestamp, ExpiresAt, and RECT. The sink periodically broadcasts the interest message to its neighbors. Initially, this *interest* message serves as an exploratory method to see if there are sensors that detect animals in the specified region. Hence, the initial interest just specifies the node to return the event information at a lower rate. In our example, the sink is expecting an event report every half second. Later on, when the sink learns can revise the interest and adopt the reinforcement scheme (to be described later) to ask for event reports at higher data rate.

![Figure 4.4 Interest Propagation in the network [Cintagonwiwat00]](image)
Figure 4.4 shows an example how the interest is propagated in WSNs. Basically, upon receiving the sensing task from the end-user (e.g., an application reside in the sink), the sink broadcasts the interest message to all its neighboring nodes. The neighboring nodes of the sink then further forward the message to network. There are several possible options for the node to re-send the interest to some subset of its neighbors as shown in Table 4.3.

<table>
<thead>
<tr>
<th>Diffusion Element</th>
<th>Potential Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interest Propagation</td>
<td>• Flooding</td>
</tr>
<tr>
<td></td>
<td>• Constrained or directional flooding based on location</td>
</tr>
<tr>
<td></td>
<td>• Directional propagation based on previously cached data</td>
</tr>
<tr>
<td>Data Propagation</td>
<td>• Reinforcement to single path delivery</td>
</tr>
<tr>
<td></td>
<td>• Multipath delivery with selective quality along different paths</td>
</tr>
<tr>
<td></td>
<td>• Multipath delivery with probabilistic forwarding</td>
</tr>
<tr>
<td>Data caching and aggregation</td>
<td>• For robust data delivery in the face of node failure</td>
</tr>
<tr>
<td></td>
<td>• For coordinated sensing and data reduction</td>
</tr>
<tr>
<td></td>
<td>• For directing interests</td>
</tr>
</tbody>
</table>
Each node maintains a cache for the interests where each entry is unique. When a node receives a particular interest, it is compared with all the entries in the cache. If the particular entry is not present in the cache, the entry is entered into cache and a gradient is set up towards the neighbor node which sends the interest packet. Figure 4.5 shows the gradients are established from the source node to the sink node in the opposite direction of the interests’ propagation as in Figure 4.4. Each gradient contains data rate and expiration time which are derived from the interest packet. On the other hand, if there is an interest entry in the cache, aggregation on the interests and updation of the gradient fields can be performed by the node. The interest entry is removed from the cache when all the gradients associated with the interest entry have expired.

| Reinforcement         | • Rules for how many neighbors to reinforce  
|                       | • Rules for deciding when to reinforce  
|                       | • Negative reinforcement mechanisms and rules  

**Table 4.3** Design Options for Diffusion [Cintanagonwiwat00]
Data Propagation

When a sensor node detects a target corresponding to one interest entry in its interest cache, the node computes the highest requested rate among all its outgoing gradients. After generating the corresponding reply (or return data message), the source node sends the reply to each neighbor for whom it has a gradient. Upon receiving the data message, intermediate node checks its cache to see whether there is a match. If a match is found, the node checks the data cache associated with the matched interest entry to decide either drop redundant message or update the cache and resend the new data message to its neighbors according to the gradients settings. On the other hand, if no interest match is identified, the data message is simply dropped. As shown in Table 4.3, in the process of data propagation, each node can prevent loop by using the data cache and can down convert to the appropriate gradient with lower data rates when necessary.

Reinforcement

Initially the sink node disseminates the interest message at a low rate. The source sensors that have data match the interest, also reply data at a low rate, possibly along multiple paths towards the sink. When the sink receives the low rate data, it chooses one particular neighbor node and sends a reinforcement packet to request higher rate data report. The reinforcement packet is similar to the original interest message, but with a smaller interval or higher data rate as follows.

\[
\begin{align*}
Type & = \text{animal} & & \text{//detect animals} \\
Interval & = 10\text{ms} & & \text{//send back events every 10ms} \\
Timestamp & = 03:02:19 & & \text{//interest generated time}
\end{align*}
\]
When the specific node receives this reinforcement, the node notice that a gradient towards the neighbor already exists and a higher rate is requested. If the new data rate is higher than that of any existing gradients, the node must also forward the reinforcement to at least one neighbor node. As shown in Table 4.3, different techniques can be applied to choose which neighbor node the reinforcement will be sent to. In general, whenever one path delivers an event faster than other paths, the sink tends to use this path to request high quality data. Through this sequence of reinforcement, paths with corresponding data rate are established from the source node to sink node.

However above process may result in more than one path being reinforced. Assume two paths: $P_1$ and $P_2$ are used for delivering data as result of above reinforcement. Now if the sink prefer path through $P_2$ over $P_1$, the sink can use above scheme to continuously send reinforcement through path $P_2$, but also need scheme of negative reinforcement to degrade the necessary rate through path $P_1$. Two approaches can be adopted for negative reinforcement. One approach is to time out the high data rate gradients in the network unless explicitly reinforced. Hence, the sink can just be kept reinforcing path $P_2$ and eventually path $P_1$ will degrade to lower data rate.

Another approach is to explicitly degrade the path $P_1$ by re-sending the interest with lower data rate.

In fact, in networks with multiple sources or multiple sinks, the schemes we described above can
also be effectively applied. In addition, in Directed Diffusion, the intermediate nodes on a previously reinforced path can also initiate the reinforcement process when local repair of failed or degraded paths are identified. Causes for failure or degradation include node energy depletion, environmental interference, raid fade, and so on.

<table>
<thead>
<tr>
<th>WSNs</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The data dissemination with Directed Diffusion is different from traditional networking in several aspects. All communication in Directed Diffusion is neighbor-to-neighbor, not end-to-end. Directed Diffusion in WSNs is suitable for task-oriented application. No globally unique identifier or address is needed.</td>
</tr>
</tbody>
</table>

**Evaluating Directed Diffusion**

One of the motivations of the Directed Diffusion scheme is the physical systems (e.g., ant colonies) that build up transmission paths using strictly local communication well and are extraordinarily robust. Another reason behind using such a scheme is to get rid of unnecessary and complicated operations of network layer routing for energy saving. Hence, *interest* and *gradients* are defined and utilized in Directed Diffusion to establish paths between sink and sources for data transmission.

Studies have shown that Directed Diffusion has the potential for significant energy saving and outperforms the idealized traditional data dissemination scheme like omniscient multicast [Cintanagonwiwat00]. In addition, *Directed Diffusion* owns features such as data-centric
dissemination, reinforcement-based adaptation to the empirically best path, and in-network data aggregation and caching. Since it is data centric, all communication is neighbor-to-neighbor with no need for a global addressing mechanism and no need for maintaining global network topology. Aggregation and caching offers better network performance in terms of energy efficiency and delay. However, Directed Diffusion cannot be applied to all sorts of sensor network applications since it is based on a query-driven data delivery model. The applications that require continuous data delivery to the sink may not work efficiently with a query-driven on demand model such as Directed Diffusion [Njamal04].

4.5 Hierarchical Routing protocols WSNs

Scalability is one of the major design concerns of sensor networks, particularly, for many applications with a huge number of sensor nodes deployed to cover a pretty large physical area. A single-tier or flat network operation in such large-scale sensor networks can cause

- large converge time for many algorithms and protocols,
- overload with increase in sensor density,
- large memory space required for storing the network information,
- increased latency, complexity, instability in communication and
- inadequate tracking of events

Since the tiny sensors with limited resources are typically not capable of performing long-haul communication, the concept of clustering or hierarchical network routing has been pursued in many routing approaches to allow the system cover a large area of interest without degrading the
service. A cluster is generally a collection of nodes with similar missions, within similar vicinity or having similar functionalities/resources. A hierarchical routing protocol can be viewed as a set of flat routing protocols, each operating at different levels of granularity. For example, in a two-tier hierarchical routing protocol, the inter-cluster component essentially a flat routing protocol that computes routes between clusters. Likewise, the intra-cluster component is a flat routing protocol, which generates routes between nodes in each cluster. Hierarchical routing protocols provide global routes to the network clusters, rather than individual nodes, which can ease many aforementioned scalability issues in the network and, are often better suited for very large networks than flat routing protocols. In addition, data aggregation and fusion can be performed within the cluster in order to decrease the number of transmitted messages to the sink, which can enhance the network performance in terms of energy efficiency.

Examples of hierarchical network routing include Low-energy adaptive clustering hierarchy (LEACH) [Bwendi02], Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [Marati01], Adaptive Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) [Marati02], Power-efficient Gathering in Sensor Information Systems (PEGASIS) [Slindsay02], Hierarchical-PEGASIS [Asavvides01], Minimum energy consumption network (MECN) [Vrodoplu99], Small Minimum energy consumption network (SMECN) [Lli01], Self Organizing Protocol (SOP) [Lsubramanian00], Sensor Aggregates routing [Qfang03], Virtual grid architecture routing [JNaal-karaki04], Hierarchical power aware routing [Qli01], Two Tier Data Dimension (TTDD) [Fye02], etc.

LEACH forms clusters of the sensor nodes based on the received signal strength and use the
local cluster heads as gateway to the base station. TEEN is a hierarchical protocol designed to be responsive to sudden and drastic changes in the sensed attributes such as temperature, pressure, rainfall, etc. APTEEN aims at both capturing periodic data collections and reacting to time critical events. PEGASIS forms chains of sensor nodes so that each node transmits and receives from a neighbor and only one node is selected from that chain to transmit to the base station rather than forming multiple clusters. Hierarchical-PEGASIS, an extension to PEGASIS, aims at decreasing the delay incurred for packets during transmission to the base station. MECN finds a sub network of the WSN with less number of nodes and finds the minimum global energy required for data transfer. SMECN, an extension of MECN, considers the obstacles in the data transmission while relaxing the assumption that every node in the network can transmit to each other in MECN. Sensor Aggregate Routing comprises of the sensor nodes with a grouping predicate for collaborative cooperative processing task. The parameters of the predicate depend on the task and the resource requirements. Virtual Grid Architecture Routing utilizes data processing and in-network processing to maximize the network lifetime. The network is divided into zones based on the Geographical Positioning System (GPS) information. In Virtual Grid Architecture Routing, the data aggregation is performed at two levels: local aggregation and global aggregation. Each zone has a local aggregator and master aggregator. In Hierarchical Power Aware Routing, the network is divided into groups based on geographical proximity and each group is allowed to decide how to route the data such that the energy consumed for routing will be minimum. In TTDD, each source node builds a grid structure for disseminating the data to the mobile sinks. The nodes which sense an event, process the signal and one of the nodes in the group which sensed the event becomes the source of the sensed data. The source node then builds a grid structure to route the data to the other nodes in the network.
4.5.1 Low Energy Adaptive Clustering Hierarchy (LEACH) Protocol [Bwendi02]

Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol is a self-organizing, adaptive clustering protocol that uses randomization to distribute the energy consumption evenly among the sensor nodes in the network. LEACH protocol aims at increasing the system lifetime and reducing the latency for transferring the data. LEACH protocol uses the following techniques to achieve its goals [Bwendi02].

1. Localized control for data transfers
2. Low energy medium access control
3. Self-configuring, randomized and adaptive cluster formation
4. Application-specific data processing like data compression and data aggregation

LEACH routing protocol divides the sensor nodes in the network into groups called clusters. The clusters have a special type of nodes called cluster head nodes. The cluster head nodes are used for transmission of the data to the base station. The cluster head nodes are also responsible for the medium access among the nodes in the cluster.

The cluster head nodes in the cluster consume more energy when compared to that of the non cluster head nodes. In order to make the energy consumption uniform among the nodes in the network, LEACH protocol uses randomized rotation for selection of cluster heads among the other nodes in the network. LEACH uses the cluster head nodes for transmission of the data from the non cluster head nodes to the base station. The data sensed by the nodes is sent to the cluster head nodes initially and then, the cluster head nodes transmit the data to the base station (BS).
Protocol Design

As described in early chapters and sections, in sensor networks deployed for environment monitoring or surveillance applications, overlap in sensing range and application-specific requirements make the sensed data in a specific region redundant and strongly correlated. The basic idea of LEACH is to form sensor nodes into clusters and locally process the correlated data such that the useless or redundant transmissions in the network are reduced.

In LEACH, the sensor nodes in the network organize themselves into groups, also called clusters. Then LEACH randomly selects a few nodes as cluster heads (CHs) and rotates this role to evenly distribute the energy load among all the sensors in the network. All non-cluster head nodes will collect sensed data and send the data to the cluster head. The cluster head aggregate and compress the data arriving from nodes that belong to the respective cluster before it sends the aggregated packet to the sink (or based station).

The operation of LEACH is divided into rounds. Each round consists of two phases, the setup phase and the steady state phase. Each round starts with a setup phase when the clusters are organized and cluster heads are selected. What is followed is the steady state phase when the cluster heads collect and process the data from the nodes within their clusters before the aggregated data is transferred to the sink.
Setup Phase: Cluster formation and Cluster-head Selection

In LEACH, sensor nodes are organized into clusters by using a distributed algorithm where nodes make autonomous decisions without any centralized control. The goal is to maintain $k$ clusters during each round and evenly distributed the load among all the nodes such that no node is overloaded and runs out of energy before others. LEACH protocol assumes that every node is initialized with equal power and can apply power control to vary transmission power. LEACH assumes that every node in the network can reach the sink with enough power. Intuitively, the cluster head will suffer more energy consumption than other nodes in the cluster due to its responsibility to process data aggregation and deliver data to the remote sink. To avoid quick energy depletion in the cluster head, LEACH incorporates randomized rotation of the cluster head role among the high-energy sensors [Hwendi00].

At the beginning of the setup phase, each node decides whether or not to act as a cluster head of the current round. This decision is based on the predetermined percentage of cluster heads in the network and the number of times the node has been a cluster-head so far. More specifically, a sensor node $n$ chooses a random number between 0 and 1. If this random number is less than a threshold value, $T(n)$, the node becomes a cluster head for the current round. The threshold value is calculated as in Equation (4.1):

$$T(n) = \begin{cases} \frac{p}{1 - p \times (r \mod \frac{1}{p})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

(4.1)

where $p$ is the desired percentage of cluster heads (e.g., 5%), $r$ is the current round, and $G$ is the
set of nodes that have not been cluster heads in the last $\frac{1}{p}$ rounds.

As we can see from Equation (4.1), each node has a probability $p$ of becoming a cluster head in round 0 (i.e., $r = 0$). After that, the cluster head nodes in round 0 cannot be elected as cluster heads for the next $\frac{1}{p}$ rounds. Hence, the probability that other node in $G$ is elected as cluster head increases since there are fewer nodes that are eligible to become cluster-heads. Eventually, the threshold in Equation (4.1) will ensure a node to be a cluster-head at some point within $\frac{1}{p}$ rounds.

After electing itself as the cluster head, the node broadcasts an advertisement message to the network indicating that a cluster is created and the advertiser is the CH. All the non-CH nodes, after receiving this advertisement, decide on the cluster to which they want to join. This decision is based on the received signal strength of the advertisement from the cluster heads. A non-CH node then sends a join-request message to the appropriate CH. After receiving all the join-request messages, the cluster head node sets up a TDMA schedule and assigns each node a time slot when it can transmit. This schedule is broadcast to all the nodes in the cluster. As shown in the flowchart of the distributed cluster formation scheme in Figure 4.6, the setup phrase ends with the reception of the TDMA schedule by all nodes in the cluster.
**Steady-State Phase**

After the non CH nodes receive the TDMA schedule created by their cluster head nodes, the nodes start transmission of data depending on the TDMA schedule. To synchronize and start the steady state phase at the same time, the sink can issue corresponding synchronization pulses to all the nodes. As shown in Figure 4.7, the *Steady state* phase is further divided into frames. During the assigned frame, sensor node can transmit the data to the cluster head node. The duration of each frame slot is constant and depends on the number of nodes in the cluster.

Since the cluster head is normally nearby, each non Ch node can apply power control scheme to set the minimal amount energy (based on the received signal strength of the cluster head.

**Figure 4.6** Flowchart of the formation of clusters in LEACH protocol [Bwendi02]
advertisement) required for the data transmission to the CH. To further improve the energy efficiency in the network, the radio of each non-CH node is turned off (sleep) until its allocated transmission time. However, the cluster head node must keep its receiver on to receive all the data from the nodes in the cluster. When the data from all the nodes in the cluster has been received, the cluster head node performs aggregation and signal processing functions to compress the data. Then the cluster head node sets necessary power level and sends the aggregated data to the sink. After a certain time, which is determined a priori, the network goes into the next round to start the setup and steady-state phases again. The duration of the steady state phase is longer than the duration of the setup phase in order to minimize the overhead.

Cluster formed

<table>
<thead>
<tr>
<th>Slot for node i</th>
<th>Slot for node (i+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Frame</td>
</tr>
</tbody>
</table>

**Figure 4.7** Time line for LEACH protocol [Bwendi02]

**LEACH-Centralized (LEACH-C)**

There are advantages in the previous approach for forming the cluster using the previous algorithm. But the previous algorithm has many disadvantages such as LEACH protocol offers no guarantee about the number of clusters in a particular area and the placement of the clusters. LEACH-C uses centralized algorithm for clusters formation and produces better results when compared to that of the LEACH protocol. In LEACH-C protocol, each node scans the current location using GPS during the setup phase and transmits its current location as well as energy...
level to the sink. Based on the energy level and location information of all the nodes in the network, the sink can select the cluster heads and form the clusters optimally in terms of minimizing the amount of energy consumed for data transmission. Then the sink broadcasts the information of the cluster formation to the network. If a node is not assigned as cluster head in this round, the node can go to sleep based on its TDMA transmission schedule. However, the cluster head node has to receive, aggregate, and forward data to the sink.

**Evaluating LEACH protocol**

In hierarchical routing protocols, each cluster designates a single cluster head node to relay inter-cluster traffic. To prevent cluster head node from becoming traffic/energy "hot-spot", potentially resulting in network congestion and single point of failure, LEACH adopts a distributed scheme to rotate cluster head roles to evenly distribute the load among all the nodes in the network. In addition, LEACH employs dynamic clustering, in-network data processing, power-controlled transmission, and collision avoidance schemes to increase the network life time. Studies in [Bwendi02] shows LEACH can achieve over a factor of 7 reductions in energy dissipation compared to direct communication and a factor of 4-8 compared to the minimum transmission energy routing protocol. Furthermore, LEACH-C protocol can further improve the network performance by forming better clusters with the global knowledge of the location and energy levels of each node in the network.

However, restricting nodes access through cluster heads can lead to sub-optimal routes and data transmission, as potential neighbors in different clusters are prohibited from communicating directly. The idea of dynamic clustering brings extra overhead for the cluster
formation/maintenance, which may diminish the gain in energy consumption. Moreover, LEACH assumes each node can transmit directly to the cluster head and the sink, which may be not applicable for networks deployed in large regions. Hence, LEACH has been extended to account for heterogeneous sensor nodes, better scalability and energy-efficiency in the literature.

4.5.2 Threshold Sensitive Energy Efficient Sensor Network (TEEN) Protocol [Marati01]

SPIN, LEACH and Directed Diffusion protocols have been developed for applications requiring periodic environment monitoring or querying a snapshot of the relevant parameters at some intervals. On the other hand, two hierarchical routing protocols called Threshold-Sensitive Energy Efficient Sensor Network Protocol (TEEN) and Adaptive Periodic TEEN (APTEEN) are proposed in [Marati01] [Marati02] for time-critical applications where responsiveness to changes in the sensed attributes is important. TEEN pursues a hierarchical approach along with the use of a data-centric mechanism to provide the end user with the ability to control the trade-off between energy efficiency, accuracy and response time dynamically.

Sensor Network Model in TEEN

In TEEN, the sink or base station can transmit data to all the nodes in the network at any point of time. However, the sensor node cannot always reach the sink directly due to the constraints of power and transmission range. Unlike the LEACH with only one-tier hierarchy, the network architecture in TEEN is based on multi-level hierarchical grouping as shown in Figure 4.8, where closer nodes form clusters and this process goes on the multiple levels (or tiers). The cluster head in each cluster collects data from its cluster members, aggregates the data, and sends the data to an upper level cluster head or the base station. Figure 4.8 shows an example of multi-tier
clustering. Nodes 1.1.1, 1.1.2, 1.1.3, 1.1.4 and 1.1.5 form a low-level cluster with node 1.1 being the cluster head. Similarly nodes 1.2 and 1 serve as the cluster heads for respective low-level clusters. The cluster-heads 1.1, 1.2, and 1 from the low-level clusters, in turn form a cluster with node 1 as the cluster-head. Hence, node 1 also becomes the cluster head of the second-level cluster. This hierarchy pattern is repeated through the network to form multi-level hierarchies. The uppermost level cluster nodes will be able to directly send data to the base station which acts as the root of the uppermost hierarchy and supervises the entire network.

Figure 4.8 An Example of Network Hierarchies in TEEN [Marati01]

With this network architecture, TEEN allows the nodes communicating with their immediate
cluster head. Hence, node does not have to be able to reach the base station directly (as required in LEACH). The data from low-level clusters may travel through multiple cluster heads before reaching the base station. The cluster heads at each level will perform necessary data processing such as aggregation and compression to conserve energy for the transmission. In order to evenly distribute the energy consumption, the nodes take turns to serve as cluster head, which is similar to LEACH.

Operation of TEEN protocol

Figure 4.9 shows the time line of the TEEN operation. After the clusters are formed, the cluster head broadcasts two thresholds to the nodes: Hard Threshold and Soft Threshold.

Hard Threshold

Hard threshold is the threshold value of the attribute beyond which, the sensing node must switch on its transmitter and report the value to its cluster head. Therefore, the hard threshold allows the nodes to transmit only when the sensed attribute is in the range of interest, which may result in significant reduction in the number of transmissions.

Soft Threshold

Soft threshold is the small change in the value of the sensed attribute which triggers the node to switch on its transmitter and transmit the sensed data to the base station. In other words, once a node senses a value at or beyond the hard threshold, it transmits data only when the values of that attribute change by an amount equal to or greater than the soft threshold. As a consequence, the soft threshold will further reduce the number of transmissions that might otherwise occur when there is little or no change in the sensed attribute.
One can adjust both hard and soft threshold values in order to control the number of data transmissions. A smaller value of the soft threshold gives a more accurate picture of the network, at the expense of increased data transmission and thus energy consumption. This indicates that the end user can control the tradeoff between energy efficiency and data accuracy by adjusting the values of the threshold. In fact, TEEN allows the user to assign new threshold values and broadcast them to the network when CHs are to change (as shown in Figure 4.9).

**Figure 4.9** Operation of TEEN protocol [Marati01]

As shown in Figure 4.9, TEEN protocol initially forms the clusters and the parameters are sent to the nodes in the network. The nodes continuously monitor their environment. The first time the value of an attribute reaches its hard threshold value, the node switches on its transmitter and transmits the sensed data to the cluster head. The sensed data is also stored in an internal variable of the node, called the *sensed value* (SV), which is also updated whenever a node transmits data. The nodes will transmit data in any cluster period, only when both of the following conditions are true [Marati01].

1. The current value of the sensed attribute has to be greater than the hard threshold
2. The current value of the sensed attribute differs from SV by an amount equal to or greater...
than the soft threshold

Evaluating the TEEN protocol

The important features of TEEN protocol include its suitability for time-critical sensing applications. The sudden or drastic changes in the value of a sensed attribute from the applications will reach the sink or user almost instantaneously. Also, since message transmission consumes much more energy than data sensing, TEEN can reduce unnecessary transmission and hence the energy consumption in this scheme can potentially be much less when compared to that in the proactive network. By adjusting the threshold values according to the criticality of the sensed attribute and the target application, TEEN can quickly adapt to the network real condition and user’s specific requirements.

The simulation has been performed on a network of 100 nodes with a fixed base station in [Marati01]. The nodes are placed in a random fashion with an initial energy of 2J in each node. Cluster formation is done as in the LEACH protocol. The energy consumption of the node is modeled as idle time power dissipation (equal to the radio electronics energy) and sensing power dissipation (equal to 10% of the radio electronics energy). Two performance metrics are used to analyze and evaluate the protocols: average energy dissipated and total number of nodes alive. The average energy dissipated is defined as the average dissipation of energy per node over time in the network (as it performs various functions such as transmitting, receiving, sensing, aggregation of data, etc.). The total number of nodes alive indicates the overall lifetime of the network. Simulation results show that TEEN performs better than LEACH-C and LEACH.
However, TEEN is not suitable for applications where periodic reports are needed because the user may not get any data at all whether the thresholds are not reached and whether the thresholds are not received. Thus, the user may not get data and will never be able to know whether there are any nodes in the network that are alive.


As an extension to TEEN, Adaptive Threshold sensitive Energy Efficient (APTEEN) protocol, on the other hand, is a hybrid protocol that changes the periodicity or threshold values used in the TEEN protocol according to user needs and the application type. APTEEN aims at proactively capturing periodic data collections and reactively respond to time-critical events. The network clustering architecture is same as in TEEN. When the base station forms the clusters, the cluster head nodes broadcast the attributes, the threshold values, and the transmission schedule to all nodes.

- **Attributes:** are a set of physical parameters which, need to be sensed in the network.
- **Thresholds:** include soft and hard thresholds which are the same as the thresholds in TEEN protocol and serve the same purposes as in TEEN protocol.
- **Count time (CT):** is the period of time after which, the sensed data needs to be sent to the cluster heads.
- **Schedule:** refers to Time Division Multiple Access schedule which, is used for sharing the transmission medium among the sensor nodes in the network.

Similar to the TEEN, the node in APTEEN senses the environment continuously, and
only those nodes that sense a data value at or beyond the thresholds report data to cluster heads. If a node does not send data for a time period equal to CT, APTEEN forces the node to sense and transmit the data. APTEEN supports three different query types:

- **Historical**: to analyze past data
- **One-time**: to take a snapshot view of the network
- **Persistent**: to monitor an event for a period of time

A TDMA schedule is used, and each node in the cluster is assigned a transmission slot. APTEEN also allows the user to set the CT interval, and the threshold values for energy efficiency. Simulations show that APTEEN’s performance is somewhere between LEACH and TEEN in terms of energy dissipation and network lifetime. TEEN gives the best performance since it decreases the number of transmissions more significantly than APTEEN does. The drawbacks of the two APTEEN are the overhead and complexity associated with forming clusters at multiple levels, threshold-based functions, manage counter time and schedule, as well as how to deal with attribute-based naming of queries.

### 4.6 Location-based Routing protocols WSNs

With advances in sensor technologies, many applications densely deploy a large number of sensor nodes carrying Global Positioning System (GPS) unit or a ranging device to facilitate the monitoring, tracing or surveillance tasks. In the absence of GPS unit, the location of nodes can be estimated through intelligent localization methods based on techniques such as coarse-grained connectivity, tri-lateration principle, robust quadrilaterals, acoustic and multimodal sensing, etc [Bulusu00] [Ward97] [Moore04] [Girod01]. The location information of the sensors can be used
to calculate the distance between the source and destinations so that the energy consumption can be estimated or the transmission power level can be properly adjusted. In addition, recall the routing scheme called Directed Diffusion described in the early part of this chapter, the location information can facilitate the sink to issue the query specifying the region in the *interest* message. Accordingly, location-based protocols are proposed to utilize position information to relay the data to the desired regions. Instead of diffusing the data to the whole network, nodes can target data on particular region or direction with help of the geographical information, which potentially reduces the number of transmissions significantly, hence improving the network performance. Examples of the location based routing are Geographic adaptive fidelity (GAF) [Yxu01], Geographic and Energy Aware routing (GEAR) [Yyan01], Greedy other adaptive face routing (GOAFR) [Fkuhn03], SPAN [Bchen02], etc.

More specifically, GAF is an energy-aware location-based routing algorithm is designed primarily for mobile ad hoc networks, but may be applicable to sensor networks as well. GAF conduct routing based on the location of the node which is associated with a point in the virtual grid formed for the covered area. GEAR uses energy aware and geographically informed neighbor selection heuristics to route a packet towards the target region. The protocol suggests the use of geographic information while disseminating queries to appropriate regions since data queries often include geographic attributes. GOAFR routes the data by picking up the nearest neighbor to the node to be next-hop in the routing process. SPAN identifies some nodes as coordinators based on their positions to form a back bone network for data transmission.
4.6.1 Geographical and Energy Aware Routing (GEAR) Protocol [Yyan01]

Unlike unicast communication, Geographical and Energy Aware Routing (GEAR) Protocol attempts to deliver data to all the nodes inside a target region, which is a common primitive in data-centric WSN applications. GEAR uses energy aware and geographically informed neighbor selection heuristics to route data towards the specified region. Each node keeps an estimated cost and a learned cost of reaching the destination region through each neighbor. The estimated cost is a combination of residual energy and distance to destination region while the learned cost is a refinement of the estimated cost that accounts for routing around holes in the network. Based on the cost information, GEAR picks the next-hop neighbors intelligently to route the data to the destination region in an energy efficient way. Once the data reaches the region, GEAR employs a recursive geographic forwarding technique to disseminate the packet within the region.

In fact GEAR compliments the Directed Diffusion by restricting the number of interest’s dissemination to a certain region rather than sending the interests to the whole network, thus conserving more energy.

Phases of GEAR

GEAR employs two phases in the process of forwarding data to all the nodes in the target region:

1. Forwarding the packet towards the target region

2. Disseminating the packet within the region

In the first phase, GEAR routes the data towards the target region. To forward the data towards the target region in an energy efficient way, GEAR take advantage of the geographical and energy information of sensor nodes to make routing decision.
In the second phase, GEAR disseminates the data in the target region by using either recursive geographical forwarding or restricted flooding schemes. When the density in the target region is high, the region is further divided into four sub regions. Four copies of the data are created and delivered to the sub regions. This splitting and forwarding process continues until all the nodes in the target region are covered. On the other hand, when the density in the target region is low, restricted flooding is a better fit in order to save energy.

*Energy Aware Neighbor Computation*

Assume that node $N$ is forwarding the packet $P$, towards the target region $R$, where $D$ is the centroid. When receiving the packet $P$, node $N$ progressively routes the packet $P$ to the target region while trying to balance the energy consumption among all $N$’s neighbors. To achieve this, GEAR introduces the concepts of *estimated* cost and *learned* cost to facilitate the routing decision.

Each node, say $N$, maintains a state $h(N_i, R)$ called the *learned* cost to region $R$. If a node does not maintain the *learned* cost of region $R$, $h(N_i, R)$, then, an estimated cost $c(N_i, R)$ is computed as the default value of $h(N_i, R)$, which is defined in Equation (4.2):

$$c(N_i, R) = \alpha d(N_i, R) + (1 - \alpha) e(N_i)$$

(4.2)

where $\alpha$ is the tunable coefficient, $d(N_i, R)$ is the distance from $N_i$ to the centroid $D$ of region $R$ normalized by the largest such distance among all neighbors of $N$, and $e(N_i)$ is the energy consumed at node $N_i$ normalized by the largest consumed energy among neighbors of $N$.
When a node picks a next-hop neighbor $N_{min}$ to forward the packet, the learned cost of region $R$ is updated as in Equation (4.3):

$$h(N, R) = h(N_{min}, R) + C(N, N_{min})$$

(4.3)

where $C(N, N_{min})$ is the cost of transmitting a packet from $N$ to $N_{min}$ and can also be a combination function of both the remaining energy levels of $N, N_{min}$ and the distance between these two nodes.

Once node $N$ has a learned cost or estimated cost for each neighbor, node $N$ has to determine which neighbor to be next-hop node for following two scenarios:

1. There is at least one neighbor of node $N$ who is closer to $D$ than $N$;
2. All $N$’s neighbors are further away from $D$ than $N$.

1. Closer neighbor exists

When there are neighbor nodes closer to the destination, GEAR uses a greedy technique to forward the data to the destination. In specific, node $N$ picks the next-hop node among the neighbors that are closer to the destination, at the same time, minimizing the learned cost value $h(N, R)$. According to Equations (4.2) and (4.3), we have three observations:

- If all $N$’s neighbors are equal in terms of energy consumption, node $N$ will choose the neighbor who has shortest distance to $D$;
• If all $N$'s neighbors have the same distance to $D$, node $N$ will split the load among neighbors;

• Otherwise, node $N$ selects next-hop node based on the trade-off between routing towards neighbor nearest to the destination and balancing energy consumption.

2. All the nodes are further away from the node $N$

When there are no neighbor nodes closer to the destination, i.e., all neighbors are farther away from the destination, we say a hole is identified. In other words, a hole occurs when a node does not have any neighbor closer to the target region than itself. In this scenario, the learned cost will be combined with an update rule in order to forward the packets circumventing the holes.

![Diagram](image)

**Figure 4.10** Example of learning routes when holes are present [Yyan01]
For example, assume that nodes $G$, $H$, and $I$ have their energy depleted completely as shown in Figure 4.10 which is a grid topology with a distance of 1 between two neighbors in the same row or column. Thus, these nodes could not forward the data. For simplicity purposes, we set the coefficient $\alpha$ in equation (4.2) as 1 and use the distance instead of normalized distance mentioned earlier. Initially, node S assumes that the neighbor nodes B, C, and D are closer to T based on the learned cost in following equations.

\[
\begin{align*}
\hat{d}_{(s, b)} &= \sqrt{5} \\
\hat{d}_{(s, c)} &= 2 \\
\hat{d}_{(s, d)} &= \sqrt{5}
\end{align*}
\]  

(4.4)

Hence, to route a packet to T, node S will choose C (which has the lowest learned cost) as the next-hop node and forward the packet to C. However, node C will find itself in a hole since all C’s neighbors are further away from T than itself. Then node C will perform two operations.

- Node C will forward the packet to a neighbor with minimal learned cost. Ties are broke based on some predefined ordering (e.g., node ID). In this case node B will receive the packet forwarded by node C.
- Node C updates its own learned cost as $h(C, T) = h(B, T) + c(C, G)$ where $h(B, T) = \sqrt{5}$ and $c(C, B) = 1$ (assume one hop transmission cost is 1) and send the learned cost back to node S.

Next time, upon receiving a packet destined to T, the learned cost values of its neighbors are as following equations.

\[
h(B, T) = \sqrt{5}
\]
\[ h(C, T) = \sqrt{5} + 1 \]
\[ h(D, T) = \sqrt{5} \]

(4.5)

At node S, instead of delivering the packet to node C (which will forward the packet to B, causing two transmissions from node S to B), node S will forward the packet to node B directly to circumvent the hole.

Hence, the learned cost is propagated one hop back every time a packet reaches the destination so that route setup for next packet will be adjusted. By propagating the learned cost values upstream through the update rule, GEAR will enable the packet to have an earlier chance to avoid holes (i.e., more effectively circumnavigate holes), and at the same time avoid depleting the nodes surrounding the holes. In addition, since the cost is a combination of the normalized distance and energy consumption, the coefficient \( \alpha \) in equation (4.2) can be tuned to emphasize minimizing path length to the destination or balancing energy consumption.

**Recursive Geographic Forwarding**

When the query packet destined to all nodes in region, \( R \), reaches the target region, a simple flooding with duplicate suppression (or restricted flooding [Finn87]) scheme can be adopted to disseminate the packet inside the region, particularly in low density scenarios. Restricted flooding exploits the broadcast medium of the wireless channel, only sends one broadcast message to all its neighbors, but every node in its transmission range receives this broadcast message.
However, flooding is expensive in terms of energy consumption, due to the fact that a significant number of redundant and useless transmissions may be introduced by the flooding. The redundant transmission can be especially expensive in high-density networks, which is the case for some WSN applications where nodes are densely and redundantly deployed for robustness. Hence, Recursive Geographic Forwarding is proposed to disseminate the packet inside target region when the node density is high. As shown in Figure 4.11, assume the big rectangle is the target region, $R$ and a particular node, $N_i$ receives a data packet $P$ for this region, $R$. The node $N_i$ finds that the packet $P$ is sent to the region where it resides. Then node $N_i$ creates four new copies of the packet $P$ and forwards it to 4 sub-regions of region $R$. This recursive splitting and forwarding procedure goes on until the current node finds itself as the only member in the sub-region.
In the case with low node density the network is subject *recursive geographic forwarding* to non-terminated and useless packet transmission. In *recursive geographic forwarding*, packet forwarding and splitting terminates if the sub region is found empty. However, the transmission range of a sensor node is small compared to the sub-region size. Hence, the node that is closed to the sub-region cannot reach the other end of the sub-region and have no idea whether the region is empty or not. As a result, *recursive geographic forwarding* still searches for routes to get into the empty sub-region. This search will not terminate until the packet is dropped because the number of hops it traversed exceeds the limit (e.g., time-to-live, or TTL). This kind of daunted search process can heavily drain the node around the sub-region, particularly in network with

**Figure 4.11** Recursive forwarding and splitting process [Yyan01]
low density in which the probability of the target region being empty is high. In addition, the unicast communication in recursive geographic forwarding cannot take advantage of the broadcast nature of the wireless medium and requires multiple transmissions in this scenario, which could suboptimal energy usage. For these reasons, in the case with low node density, restricted flooding is employed by GEAR replacing the recursive geographic forwarding [Yyan01].

GEAR proposes to use node degree for differentiation of the low density from high density. When the packet reaches the first node $N$, in a region, whether to use restricted flooding or recursive geographic forwarding depends on the number of neighbors of node $N$. If the number is below a threshold, then the packet is flooded inside the region, otherwise, recursive geographic forwarding will be triggered.

*Evaluating GEAR Protocol*

GEAR use energy aware metrics, together with geographical information, to make energy efficient routing decisions. While balancing the energy consumption and thereby increasing the network lifetime, GEAR progressively forward data to the target region based the proposed cost function and update rule. Within a region, it uses a restricted flooding or recursive geographic forwarding technique to disseminate the data. GEAR is compared to a similar non-energy-aware routing protocol GPSR [Bkrap00], in which the packets follow the perimeter of the planar graph to find their route. GEAR not only reduces energy consumption for the route setup, but also outperforms GPSR in terms of packet delivery. The simulation results show that for an uneven traffic distribution, GEAR delivers 70% to 80% more packets than GPSR. For uniform traffic
pairs GEAR delivers 25%-35% more packets than GPSR. Moreover, in both cases, GEAR achieves better connectivity after initial partition [Yyan01].

4.7 Multipath and QoS-based Routing

To maintain the network reliability, enhance the throughput or balance the traffic, the techniques employing multipath routing are often employed. Multipath routing can provide route resilience through redundant packets delivering over multiple paths or fast route recovery from network disruption. As the bandwidth may be limited in a sensor network, routing along a single path may not provide enough bandwidth for some applications such as camera or video capture. If multiple paths are employed simultaneously to route the data, larger aggregated bandwidth and smaller end-to-end delay may be achieved. Similarly, load balancing can be achieved by spreading the traffic along multiple routes, which can alleviate congestion and bottlenecks in the network.

Therefore, we can see that different strategies to utilize the multiple paths can result in enhancements in different network performance metrics, which actually occurs in many QoS-based routing protocols. In the rest of this Chapter, we introduce some basic principles of multipath routing followed by QoS-based routing schemes in sensor networks.

4.7.1 Multipath Routing

In multipath routing, there are multiple, say \( k \), paths between source and destination nodes. The \( k \) paths are \textit{link-disjoint} if they have no common links. The \( k \) paths are \textit{node-disjoint} if they have no common intermediate nodes. We call two or more paths non-disjoint (or braided) if they
share some links or intermediated nodes (i.e., the node/link disjointedness constraint is relaxed). Multipath routing has been explored for several important reasons. The first is to increase the likelihood of *reliable data delivery*. Sending multiple copies of data along different paths simultaneously offers resilience to failure of a certain number of paths [Ganesan01]. Duplicate data transmission along multipaths can result in more accurate delivery and better data quality for WSNs, at the possible expense of increased traffic redundancy and energy consumption. The second is to enhance the *throughput* from a source to a destination. In these approaches, data for the same source-destination pair are sent out through multiple paths which create multiple data flows and hence potentially increasing the throughput from the source to the destination. These multiple flows are better considered together with the wireless interferences among the nodes in the MAC layer to achieve optimized performance. Another major benefit of multipath routing is *load balancing*. In this case, the source and destination use only one path for routing the data, which is called the primary path. Multiple path candidates alternatively serve as the primary path for routing data from the same source-destination pair, which can spread energy consumption across nodes on multipath in the network. This approach can avoid depleting the energy resources of some nodes through constant usage of the same route, potentially resulting in longer network lifetime. Moreover, if there are node failures in the primary path, multipath routing can immediately employ the alternate paths which are constructed along with the primary path to continuously deliver data from the source node to the destination node.

Generally, in routing when there are node failures in primary path, then the nodes in the primary path use flooding for routing the data in the network, to reconstruct the path and recover from the failure. However, multipath routing scheme can quickly recover from the failure by selecting the alternate paths which are already constructed along with the primary path (without any cost for searching for another one).
For example, the authors in [Chang04] assume that the transmitter power level can be adjusted to use the minimum energy required to reach the intended next hop receiver. Hence, the energy consumption rate per unit information transmission heavily depends on the choice of the next hop node, i.e., the routing decision. The routing problem is formulated as a linear programming problem with the objective of maximizing the network lifetime. A routing algorithm is also proposed to route data through a shortest cost path routing whose link cost is a combination of transmission and reception energy consumption and the residual energy levels at the two end nodes. Alternative path is employed whenever a better path is discovered. Two different models are considered for the information-generation processes: constant rate and arbitrary. Simulation results with both information-generation process models show that the proposed routing algorithm can achieve network lifetime that is very close to the optimal performance obtained by solving the linear programming formulations [Chang04]. Another example of multipath routing is demonstrated in [Dulman03], whereas the techniques of multipath routing are used to enhance the reliability of WSNs. As mentioned earlier, network reliability can be increased by providing several paths from source to destination and sending the same packet on multiple paths, which may result in significant traffic redundancy. Hence, there is a trade-off between the amount of traffic redundancy and the reliability of the network. This trade-off is investigated in [39] using a redundancy function that is dependent on the multipath
degree and failing probabilities of the available paths. The proposed idea is to split the original data packet into subpackets which are sent through the multiple paths. As a result, even if some of these subpackets are lost, the original message can still be reconstructed due to the redundancy added into the data transmission process. In addition, Directed Diffusion [Cintanagonwiwat00] is also a good example employing robust multipath routing and delivery. Based on the Directed Diffusion paradigm, the authors in [Ganesan01] investigates how to construct a small number of multipaths in WSNs such that failures on the primary path can be recovered without invoking network-wide flooding for path discovery (thus enhancing network energy performance). Two typical multipath designs: node-disjoint multipath and braided multipath (that consists of partially disjoint/overlapped alternate paths) schemes are evaluated in terms of the energy/resilience tradeoffs under independent and geographically-correlated failures. The study has found that, for a disjoint multipath configuration whose patterned failure resilience is comparable to that of braided multipaths, the braided multipaths have about 50% higher resilience to isolated failures and a third of the overhead for alternate path maintenance [Ganesan01]. Therefore, the braided multipaths are a viable alternative for energy-efficient recovery from isolated and patterned failures with lower maintaining cost in WSNs.

4.7.2 QoS-based Routing Protocols in WSNs [Ksohrabi00]

For different WSN applications, the constraints and the QoS metrics (delay, energy, priority, bandwidth, fairness, robustness, etc.) to be optimized can be different. Many principles for QoS in the traditional networks can find a counterpart in sensor networks. Some of the QoS metrics are still important and challenging under the constraints of WSNs and others are not as significant as they are in Internet. For example, delay, robustness, bandwidth, and energy are
paramount and optimized goals in many WSN applications. However, in many cases, the sensors are designed to collectively and cooperatively carry out a task, which diminish the importance of QoS metrics such as fairness. Examples of QoS-based and Multipath routing include SPEED [The03] and Sequential Assignment Routing (SAR) [Ksohrabi00]. SPEED is a routing protocol for sensor networks that provides soft real-time end-to-end guarantees, requires each node to maintain information about its neighbors and uses geographic forwarding to find the paths.

Demand-driven routing protocols are the protocols that find the route between the source and destination systems after a request or demand is issued. These protocols eliminate the overhead associated with table or neighbor update in high mobility scenarios. However, demand-driven protocols may take a longer time and energy to find the route in a reactive way. Sequential assignment routing (SAR), on the other hand is a table-driven multi-path approach striving to achieve energy efficiency and fault tolerance. Based on the observation that the possibility of protection on the failures in WSNs is tightly related to the degree of disjointedness (i.e., the number of paths with no common branches) in the network, SAR protocol creates trees rooted at one-hop neighbors of the sink by taking QoS metric, energy resource on each path and priority level of each packet into consideration. By using the trees, multiple paths from sink to sensors are formed. One of these paths is selected according to the energy resources, QoS on each path and the priority level of a packet. Failure recovery is done by enforcing routing table consistency between upstream and downstream nodes on each path, which is done using a handshaking procedure. Any local failure is taken care of, by an automatic path restoration procedure, which is done locally. The simulation studies in [Ksohrabi00] show that SAR can achieve energy-efficiency while taking packet priorities into account. The multiple paths maintained by SAR
ensure that the system is fault-tolerant and easily recoverable. However, the overhead of table maintenance at each sensor node makes SAR infeasible for very large-scale WSNs.

4.8 Conclusion

In this chapter, we have gone through the challenges and concerns in designing routing protocols for wireless sensor networks. The classification of the routing protocols proposed in the literature and several typical routing schemes such as SPIN, Directed Diffusion, LEACH, TEEN, GEAR SAR and multipath routing schemes are elaborated.
Problems & Exercises

3.1 Multi-choice questions:

(1) Out of the following protocols, which is an example data-centric protocol?

A. Rumor routing
B. Minimum Cost Forwarding Algorithm
C. SPEED
D. Gradient Based Routing

(2) Which of the following is not a choice for data diffusion?

A. Flooding
B. Gossiping
C. Directional Propagation
D. None of the above

(3) SPIN does not have one of these data packets?

A. ADV
B. REQ
C. ACK
D. Data

3.2 Explain three challenges for data routing in sensor networks?
3.3 Explain the differences between SPIN-PP and SPIN-BC. Explain the use of resource manager in SPIN protocol.

3.4 Explain how the gradient in Directed Diffusion is created.

3.5 Explain how the clusters are formed in LEACH protocol. How the cluster head are determined in LEACH and LEACH-C?

3.6 Explain the purposes of hard threshold and soft threshold in TEEN.

3.7 Explain the different phases in GEAR. Explain why restricted flooding and recursive geographic forwarding are used in GEAR?

3.9 Explain the differences between Disjoint Multipath routing and Braided Multipath routing schemes.
References


