6

Clustering Techniques in Wireless Sensor Networks

6.1. INTRODUCTION

Advances in MEMS technology have resulted in cheap and portable devices with formidable sensing, computing and wireless communication capabilities. A network of these devices is invaluable for automated information gathering and distributed microsensing in many civil, military and industrial applications. The use of wireless media for communication provides a flexible means of deploying these nodes without a fixed infrastructure, possibly in an inhospitable terrain. Once deployed, the nodes require minimal external support for their functioning.

Topology discovery algorithm for wireless sensor networks finds a set of distinguished nodes to construct the approximate topology of the network. The distinguished nodes reply to the topology discovery probes, thereby minimizing the communication overhead. The algorithm forms a tree of clusters, rooted at the monitoring node, which initiates the topology discovery process. This organization is used for efficient data dissemination and aggregation, duty cycle assignments and network state retrieval. The mechanisms are distributed, use only local information, and are highly scalable.

The vision of ubiquitous computing is based on the idea that future computers will merge with their environment until they become completely invisible to the user. Distributed wireless microsensor networks are an important component of this ubiquitous computing and small dimensions

Wireless Sensor Network Designs A. Hać
© 2003 John Wiley & Sons, Ltd ISBN: 0-470-86736-1

are a design goal for microsensors. The energy supply for the sensors is a main constraint in the intended miniaturization process. It can be reduced only to a limited degree since energy density of conventional energy sources increases slowly. In addition to improvements in energy density, energy consumption can be reduced. This approach includes the use of energy-conserving hardware. Moreover, a higher lifetime for sensor networks can be accomplished through optimized applications, operating systems, and communication protocols. Particular modules of the sensor hardware are turned off when they are not needed.

Wireless distributed microsensor systems enable fault-tolerant monitoring and control of a variety of applications. Due to the large number of microsensor nodes that may be deployed and the long required system lifetimes, replacing the battery is not an option. Sensor systems must utilize the least possible energy while operating over a wide range of scenarios. These include power-aware computation and communication component technology, low-energy signaling and networking, system partitioning considering computation and communication trade-offs, and a power-aware software infrastructure.

Many dedicated network protocols (e.g. routing, service discovery, etc.) use flooding as the basic mechanism for propagating control messages. In flooding, a node transmits a message to all of its neighbors which, in turn, transmit to their neighbors until the message has been propagated to the entire network. Typically, only a subset of the neighbors is required to forward the message in order to guarantee complete flooding of the entire network. If the node geographic density (i.e. the number of neighbors within a node's radio reach) is much higher than what is strictly required to maintain connectivity, the flooding becomes inefficient because of redundant, superfluous forwarding. This superfluous flooding increases link overhead and wireless medium congestion. In a large network, with a heavy load, this extra overhead can have a severe impact on performance.

6.2. TOPOLOGY DISCOVERY AND CLUSTERS IN SENSOR NETWORKS

Wireless sensor networks pose many challenges, primarily because the sensor nodes are resource constrained. Energy is constrained by the limited battery power in sensor nodes. The form factor is an important node design consideration for easy operability and specified deployment of these nodes, which limit the resources in a node. The protocols and applications designed for sensor networks should be highly efficient and optimize the resources used.

Sensor network architectures use massively distributed and highly complex network systems comprising hundreds of tiny sensor nodes. These nodes experience various modes of operation while maintaining local knowledge of the network for scalability. The nodes may also use networking functionalities such as routing cooperatively to maintain network connectivity. The behavior of the network is highly unpredictable because of randomness in individual node state and network structure.

Topology discovery algorithm for sensor networks uses data dissemination and aggregation, duty-cycle assignments and network-state retrieval. Network topology provides information about the active nodes, their connectivity, and the reachability map of the system.

The topology discovery algorithm uses the wireless broadcast medium of communication. The nodes know about the existence of other nodes in their communication range by listening to the communication channel. The algorithm finds a set of distinguished nodes, and by using their neighborhood information constructs the approximate topology of the network. Only distinguished nodes reply to the topology discovery probes, thereby reducing the communication overhead of the process. These distinguished nodes form clusters comprising nodes in their neighborhood. These clusters are arranged in a tree structure, rooted at the monitoring or the initiating node.

The tree of clusters represents a logical organization of the nodes and provides a framework for managing sensor networks. Only local information between adjacent clusters flows from nodes in one cluster to nodes in a cluster at a different level in the tree of clusters. The clustering also provides a mechanism for assigning node duty cycles so that a minimal set of nodes is active in maintaining the network connectivity. The cluster heads incur only minimal overhead to set up the structure and maintain local information about its neighborhood.

Sensor networks have fundamentally different architecture than wired data networks. Nodes are designed with a low cost and small form factor for easy deployment in large numbers. Hence limited memory, processor and battery power is provided. Energy constraints also limit the communication range of these devices. These nodes have various modes of operation with different levels of active and passive states for energy management. They maintain only local knowledge of the network as global information storage is not scalable, and may provide networking functionalities like routing, to maintain cooperatively the network connectivity.

The behavior of the network can be highly unpredictable because of the operating characteristics of the nodes and the randomness in which the network is set up. Hence the algorithms consider failure of a network as a rule rather than as an exception, and can handle this more efficiently.

A sensor network model incorporates the specific features as follows:

- Network topology describes the current connectivity and reachability of the network nodes and assists routing operations and future node deployment.
- The energy map provides the energy levels of the nodes in different parts of the network. The spatial and temporal energy gradient of the network nodes coupled with network topology can be used to identify the low energy areas of the network.
- The usage pattern describes node activity, data transmitted per unit of time, and emergency tracking in the network.
- The cost model provides equipment cost, energy cost, and human cost for maintaining the network at desired performance level.
- Network models take into account that sensor networks are highly unpredictable and unreliable.

The above models form the Management Information Base (MIB) for sensor networks. To update the MIB with the current state of the network, a monitoring node measures various network parameters. Measurements have spatial and temporal error, and the measurement probes have to operate at a finer granularity. A probe uses energy from the system, and in this way can change the state of the network.

The models are used for different network management functions as follows:

- Sensors are deployed randomly with little or no prior knowledge of the terrain. Future deployment of sensors depends upon the network state.
- Setting network operating parameters involves routing tables, node duty cycles, timeout values of various events, position estimation, etc.
- Monitoring network states using network models involves periodic measurements to obtain various states like network connectivity, energy maps, etc.
- Reactive network maintenance is served by monitoring the network when the regions of low network performance are traced to identify the reasons for poor performance. Corrective measures like deployment of new sensors or directing network traffic around those regions are useful.
- Proactive network maintenance allows predicting of future network states from periodic measurement of network states to determine the dynamic behavior of the network, and to predict the future state. This is useful for predicting network failures and for taking a preventive action.
- Design of sensor-network models with cost factor and usage patterns is used for design of sensor network architectures.

6.2.1. Topology Discovery Algorithm

The topology discovery algorithm used in sensor networks constructs the topology of the entire network from the perspective of a single node. The algorithm has three stages of execution as follows:

- A monitoring node requires the topology of the network to initiate a topology discovery request.
- This request diverges throughout the network reaching all active nodes.
- A response action is set up that converges back to the initiating node with the topology information.

The request divergence is through controlled flooding so that each node forwards exactly one topology discovery request. Note that each node should send out at least one packet for other nodes to know its existence. This also ensures that all nodes receive a packet if they are connected. Various methods may be employed for the response action.

When topology discovery request diverges, every node receives the information about neighboring nodes. In the response action, each node can reply with its neighborhood list. To illustrate the response action of these methods, the network in Figure 6.1 is presented with node A as the initiating node. The topology discovery request reaches node B from node A, and nodes C and D from node B. Requests are forwarded only once so that no action takes place even though node C and D may hear requests from each other.

In the direct response approach we flood the entire network with the topology discovery request. When a node receives a topology discovery request it forwards this message and sends back a response to the node from which the request was received. The response action for the nodes in Figure 6.1 is as follows:

- node B replies to node A;
- node C replies to node B; node B forwards the reply to node A;

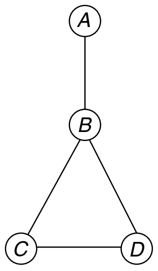

Figure 6.1 An example illustrating topology discovery.

- node D replies to node B; node B forwards the reply to node A;
- node A gets the complete topology.

Note that even though parent nodes can hear the children while they forward a request (for example, node A knows about node B when node B forwards), this is not useful because its neighborhood information is incomplete. Hence an exclusive response packet is needed for sending the neighborhood information.

In an aggregated response, all active nodes send a topology discovery request but wait for the children nodes to respond before sending their own responses. After forwarding a topology discovery request, a node gets to know its neighborhood list and children nodes by listening to the communication channel. Once this is set up, the node waits for responses from its children nodes. Upon receiving the responses, the node aggregates the data and sends it to its own parent. The response action for the nodes in Figure 6.1 is as follows:

- nodes C and D forward request; node B listens to these nodes and deduces them to be its children;
- node C replies to node B; Node D replies to node B;
- node B aggregates information from nodes C, D and itself; node B forwards the reply to node A;
- node A gets the complete topology.

In a clustered-response approach, the network is divided into set of clusters. Each cluster is represented by one node (called the cluster head) and each node is part of at least one cluster. Thus each node is in the range of at least one cluster head. The response action is generated only by the cluster heads, which send the information about the nodes in its cluster. Similarly to the aggregated response method, the cluster heads can aggregate information from other cluster heads before sending the response. The response action for the nodes in Figure 6.1 is as follows:

- assume that node B is a cluster head and nodes C and D are in its cluster;
- nodes C and D do not reply;
- only node B replies to node A;
- node A does not receive the information about the link $C \longleftrightarrow D$.

The information may be incomplete in using the clustered response approach. Direct and aggregated response methods provide an accurate view of the network topology. The clustered response creates a reachability map in which all reachable cluster heads allow all other nodes to be reachable from at least one cluster head.

The overhead incurred in topology discovery by the clustered-response approach is significantly lower than the direct or aggregated-response approaches.

6.2.2. Clusters in Sensor Networks

The communication overhead for the clustered response approach depends on the number of clusters that are formed and the length of the path connecting the clusters. Thus, to achieve the minimum communication overhead, the following problems need to solved:

- *Set cover problem*: to find a minimum cardinality set of cluster heads, which have to reply.
- *The Steiner tree problem*: to form a minimal tree with the set of the cluster heads.

These are the combinatorial optimization problems. Moreover, for an optimal solution we need to have global information about the network whereas the nodes only have local information. Thus, a heuristics approach is used, which provides an approximate solution to the problems. The algorithm is simple and completely distributed, and can thus be applied to sensor networks.

The topology discovery algorithm for finding the cluster heads is based on the simple greedy log(n)-approximation algorithm for finding the set cover. At each stage a node is chosen from the discovered nodes that cover the maximum remaining undiscovered nodes. In the case of topology discovery, the neighborhood sets and vertices in the graph are not known at runtime, thus the implementation of the algorithm is not straightforward. Instead the neighborhood sets have to be generated as the topology discovery request propagates through the network. Two different node-coloring approaches are used to find the set of cluster heads during request propagation: the first approach uses three colors and the second approach uses four colors. The response generation mechanism is the same in both cases.

In the request propagation with three colors, all nodes, which receive a topology discovery request packet and are alive, are considered to be discovered nodes. The node coloring describes the node state as follows:

- White is an undiscovered node, or node that has not received a topology discovery packet.

- Black is a cluster head node, which replies to topology discovery request with its neighborhood set.
- Grey is a node that is covered by at least one black node, i.e. it is a neighbor of a black node.

Initially all nodes are white. When the topology discovery request propagates, each node is colored black or gray according to its state in the network. At the end of the initial phase of the algorithm, each node in the network is either a black node or the neighbor of a black node (i.e. grey node). All nodes broadcast a topology discovery request packet exactly once in the initial phase of the algorithm. Thus all nodes have the neighborhood information by listening to these transmissions. The nodes have the neighborhood lists available before the topology acknowledgment is returned.

Two heuristics are used to find the next neighborhood set determined by a new black node, which covers the maximum number of uncovered nodes. The first heuristic uses a node coloring mechanism to find the required set of nodes. The second heuristic applies a forwarding delay that is inversely proportional to the distance between the receiving and sending nodes. These heuristics provide a solution quite near to the centralized greedy set cover solution. The process is as follows:

- The node that initiates the topology discovery request is assigned the color black and broadcasts a topology discovery request packet.
- All white nodes become grey nodes when they receive a packet from a black node. Each grey node broadcasts the request to all its neighbors with a random delay inversely proportional to its distance from the black node from which it received the packet.
- When a white node receives a packet from a grey node, it becomes a black node with some random delay. If, in the meantime, that white node receives a packet from a black node, it becomes a grey node. The random delay is inversely proportional to the distance from that grey node from which the request was received.
- Once nodes are grey or black, they ignore other topology discovery request packets.

A new black node is chosen to cover the maximum number of as-yet uncovered elements. This is achieved by having a forwarding delay inversely proportional to the distance between the sending and receiving nodes. The heuristic behind having a forwarding delay inversely proportional to distance from the sending node is explained as follows.

The coverage region of each node is the circular area centered at the node with radius equal to its communication range. The number of nodes covered by a single node is proportional to its coverage area times the local node density. The number of new nodes covered by a forwarding node is proportional to its coverage area minus the already-covered area. This is illustrated in Figure 6.2 where node *A* makes nodes *B* and *C* grey. Node *B* forwards a packet before node *C* does, so that more new nodes can receive the request. The delay makes node *D* more likely to be black than is node *E*. The intermediate node between two black nodes (node *B* in Figure 6.2) is always within the range of both the black nodes since three colors were used for their formation.

In the request propagation with four colors, all nodes that receive a topology discovery request packet and are alive, are considered to be discovered nodes. The node coloring describes the node state as follows:

- White is an undiscovered node, or nodes, that has not received a topology discovery packet.
- Black is a cluster head node that replies to a topology discovery request with its neighborhood set.
- Grey is a node that is covered by at least one black node, i.e. it is a neighbor of a black node.
- Dark grey is a discovered node that is not currently covered by any neighboring black node and is hence two hops away from a black node. The white node changes to dark grey on receiving a request from a grey node.

This method propagates in similar fashion to the three-color method. Initially all nodes are white. When the topology discovery request propagates,

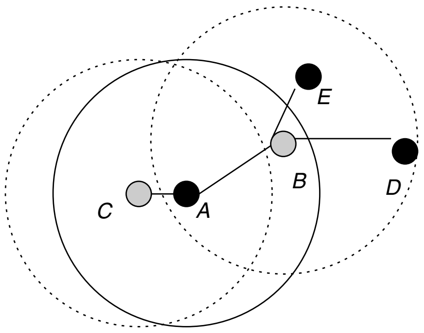

Figure 6.2 Illustration of the delay heuristic for three colors.

each node is colored black, gray or dark grey according to their state in the network. Thus at the end all nodes in the network are either black nodes or neighbors of black nodes (i.e. gray nodes) as follows:

- The node that initiates the topology discovery request is assigned color black and broadcasts a topology discovery request packet.
- All white nodes become grey nodes when they receive a packet from a black node. These grey nodes broadcast the request to all their neighbors with a delay inversely proportional to its distance to the black node from which they received the request.
- When a white node receives a packet from grey node it becomes dark grey. It broadcasts this request to all its neighbors and starts a timer to become a black node. The forwarding delay is inversely proportional to its distance from the grey node from which it received this request.
- When a white node receives a packet from dark grey node, it becomes a black node with some random delay. If, in the meantime, that white node receives a packet from a black node, it becomes a grey node.
- A dark grey node waits for a limited time for one of its neighbors to become black. When the timer expires, the dark grey node becomes a black node because there is no black node to cover it.
- Once nodes are grey or black they ignore other topology discovery request packets.

The heuristic behind having four colors for the algorithm is explained by using Figure 6.3. A new black node should be chosen so that it covers the maximum number of as-yet uncovered elements. The black nodes are separated from each other by two hops so that nodes belong to only one black node neighborhood (for instance, nodes *A* and *D*). This may not be possible in all cases and some black nodes are formed just one hop away from another

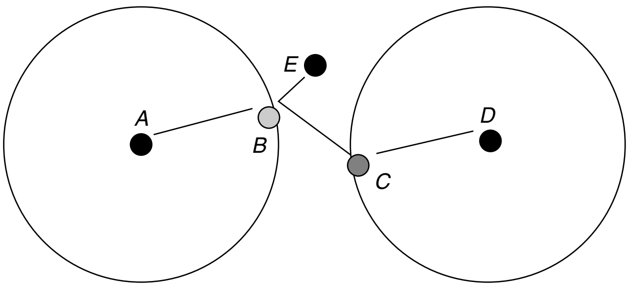

Figure 6.3 Illustration of the delay heuristic for four colors.

(for instance, node *E*). The heuristic behind the forwarding delay principle is similar to the three color heuristic.

The number of clusters formed by the four color heuristic is slightly lower than by the three color heuristic. In four color heuristic the clusters are formed with lesser overlap. There are some solitary black nodes, created from dark grey nodes that timed out to become black, which do not need to cover any of their neighbors. Thus, even though the number of black nodes is similar to the three color heuristic, the number of bytes transmitted is lower. However, the three-coloring approach generates a tree of clusters, which is more amenable to the network management applications.

In the topology discovery algorithm response mechanism, the first phase of the algorithm is to set up the node colors. The initiating node becomes the root of the black node tree where the parent black nodes are at most two hops away (using four colors) and one hop away (using three colors) from its children black node. Each node has the following information:

- A cluster is identified by the black node, which heads the cluster.
- A grey node knows its cluster ID (identifier).
- Each node knows its parent black node, which is the last black node from which the topology discovery was forwarded to reach this node.
- Each black node knows the default node to which it forwards packets in order to reach the parent black node. This node is essentially the node from which the black node received the topology discovery request.
- All nodes have their neighborhood information.

Using the above information, the steps for topology discovery algorithm response are described as follows:

- When a node becomes black, it sets up a timer to reply to the discovery request. Each black node waits for this time period during which it receives responses from its children black nodes.
- The node aggregates all neighborhood lists from its children and itself, and when its time period for acknowledgment expires, it forwards the aggregated neighborhood list to the default node the next hop to its parent.
- All forwarding nodes in between black nodes may also add their adjacency lists to the list to black nodes.

For the algorithm to work properly, timeouts of acknowledgments should be properly set. For example, the timeouts of children black nodes should always expire before a parent black node. Thus, timeout value is set inversely proportional to the number of hops a black node is away from the monitoring node. An upper bound on the number of hops between extreme nodes is required. If the extent of deployment region and communication range of nodes is known initially, the maximum number of hops can be easily calculated. However, if that information is not available to the nodes, the topology discovery runs in stages where it discovers only a certain extent of the area at each stage. A typical tree of clusters obtained by the topology discovery algorithm is shown in Figure 6.4. The example shows a 100×100 square meters area with 200 nodes and communication range of 20 meters. The arrow represents the initiating node. The characteristics of the clusters are as follows:

- The total surface area and the communication range of nodes bound the maximum number of black nodes formed.
- The number of nodes in each cluster depends on the local density of network.
- The depth of the tree is bounded.
- Routing paths are near optimal for data flow between source (sensor nodes) and sink (monitoring node).

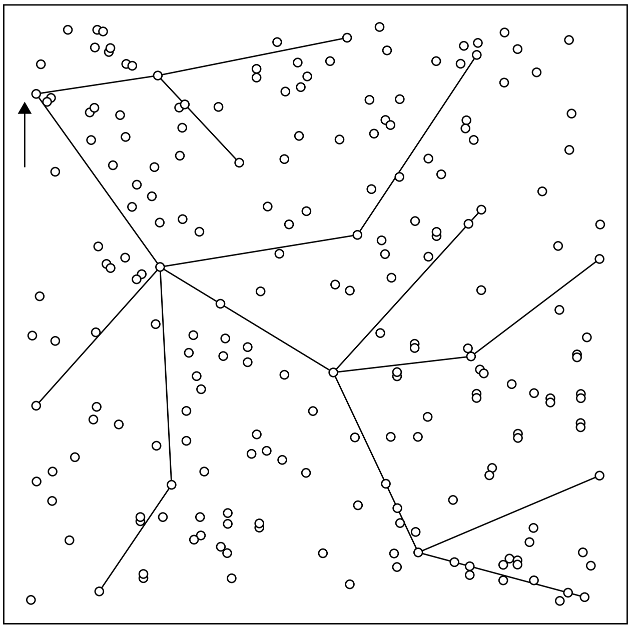

Figure 6.4 Illustration of a tree of clusters with 200 nodes and 20 meters range.

These algorithms assume a zero error rate for channels. However, minor adjustments to the protocols are needed to account for dropped packets due to channel errors.

The topology discovery initially floods the entire network. Hence channel error is not a problem as long as topology discovery requests reach a node from any path. Since the sensor networks under consideration here are dense, with many paths existing between source and destination, channel error does not create a significant impact. The number of black nodes formed may be increased due to packet losses.

However, topology acknowledgment packets are returned through single prescribed paths and hence packets may be lost. Also, the algorithm decreases the redundancy of topology information propagated among different packets, and the loss of a packet may be significant. As the packets are aggregated while moving up the cluster tree, the magnitude of loss may increase.

This problem has a simple solution if all links are symmetrical. If node A can listen to node B when node B is transmitting, then node B can listen to node A when node A transmits. When a topology discovery response has to be sent from node A, it forwards the packet to its default node (say node B). Node B, upon receiving this packet, will again forward it to its own default node, the next hop to the parent. Now node A can listen to any packet forwarded by node B and hence node A would know whether node B forwarded the same packet. If node A does not hear such a packet, it retransmits the packet assuming that node B never received the packet due to channel error.

Eavesdropping can be used as an indirect acknowledgment mechanism for reliable transmission. The only added overhead for this simple method is that every forwarded packet has to be stored at a node until the packet is reliably transmitted. The node uses energy while listening to transmissions and cannot switch itself off immediately after forwarding a packet.

6.2.3. Applications of Topology Discovery

The main purpose of the topology discovery process is to provide the network administrator with the network topology as follows:

- *Connectivity map*. The direct response and the aggregated response mechanisms provide the entire connectivity map of the region. Note that clustered response methods cannot provide this information.
- *Reachability map*. The topology discovery algorithm mechanism provides a reachability map of the region. The connectivity map is a superset of the reachability map.

- *Energy model.* When a node forwards the topology discovery request, it can include its available energy in the packet. Each node can cache the energy information of all its neighbors. If a node does not become black, it can discard the cached value. Thus all black nodes have energy information about all their neighbors, which can be sent as part of the reply. A black node can also estimate the energy consumption of nodes in its cluster by listening to the transmitted packets.
- *Usage model.* As in the previous case, each node can transmit the number of bytes received and transmitted by this node during the last several minutes. A black node will have this information cached at the time it sends its response.

This way the topology discovery algorithm provides different views of the network to the user.

We assume that in a sensor network the information flows from a sensor to the monitoring node with some control information transmitted from monitoring node to the sensors. The topology discovery process sets up a tree of clusters rooted at the initiating node. Thus, any data flow from a sensor to the monitoring node has to flow up the tree of clusters.

Each cluster has a minimal number of nodes, which are active to transfer packets between a parent and child cluster pair. Whenever a sensor needs to send some data to the monitor, it can just wake up and broadcast. The duty cycle assignment mechanism ensures that at least one node is active and responsible for forwarding the data to the next cluster. There is also at least one node in the next cluster active to receive this packet.

Each black node covers a region given by its communication range. The parent black node, logically, also covers the area covered by its children black nodes. Thus the area covered propagates up the tree and the monitor covers the entire area. The area covered by each black node may be cached during the topology response phase. The parent black node receives such areas from its children and, in turn, makes the larger area to approximate to its logical coverage region. Region based queries from the monitor node can be channeled to the appropriate region by the black nodes using their coverage information. On the return path the data may be aggregated at the black nodes.

The duty cycle of nodes for data forwarding is set up as follows. Each node in a cluster has at least the following information: the cluster ID (identifier) and the parent black node, which is the last black node from which the topology discovery request was forwarded. In each cluster, by using this information, the sets of nodes between two clusters are chosen to forward packets between clusters. At least one node in each set is active at a given time to maintain a link between a parent and child cluster pair.

In the assignment with location information, the nodes have knowledge about their geographical location. After a black node has sent a topology acknowledgment, it has knowledge of both its parent black node and the children black nodes. By using this information, the sets of nodes for each parent and child pair, need to be set up, so that in any set only one node needs to be active to transfer or receive packets from the clusters.

Figure 6.5 shows a general case in which a cluster (with black node *B*) may be formed as child of another cluster (with black node *A*). Since three colors are used to set up the tree of clusters, there is an intermediate node between the clusters (node *C*).

The communication range of nodes is equal to *R*. In a circular area with radius $R/2$, shown by the dotted circle, nodes always form a completely connected graph, as each node is within communication range of other nodes. This region is centered at the midpoint (point *P*) of a parent and child cluster pair. If there is at least one node active in both clusters inside this region, then a packet can be forwarded from one cluster to the other cluster. The algorithm to set up the sets of nodes is as follows.

- Black nodes send a packet with information about its parent cluster and children clusters to all its neighbors. This packet also contains the location information about the black nodes, which are the heads of the respective clusters.
- Nodes decide to be a part of the packet forwarding set by considering a circular region of radius $R/2$ centered at midpoint of the particular pair of black nodes.
- If a node is inside such region for a particular packet pair, this node becomes an active forwarding node for that cluster pair with some random delay.
- When the node becomes a forwarding node it sends a packet to signal this event. All other nodes go to the sleep mode for this pair of clusters. However, they may be in an active mode for the other pairs of clusters.

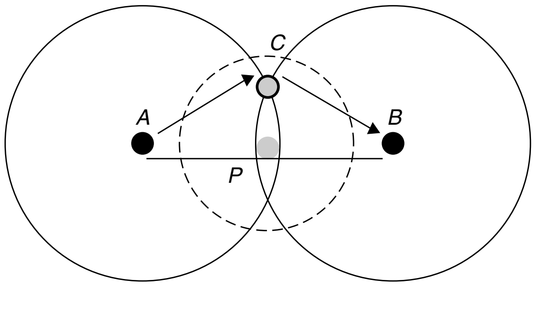

Figure 6.5 Assigning up the duty cycle with location information.

- A node may give up its active state for a cluster pair after this node has spent a certain amount of energy. The node sends a signal so that one of the other sleeping nodes can become active. When the active node receives a response signal from another node it goes to sleep mode for that cluster pair.
- Although the circular region of radius $R/2$ overlaps two clusters, there may not be other nodes in both clusters. Since all nodes can receive transmission from each other in this region, when an active node in a cluster receives a signal about activation of another node in the other cluster, it signals the black node that there exists at least one node in each cluster and the overlap regions may be used for packet forwarding.
- The intermediate node between two black nodes is used for forwarding if the overlap region does not have the other nodes in both clusters.
- During forwarding, the black node listens to all packets and forwards only the packets from the sending node which is out of range for the active forwarding node.

Figure 6.6 illustrates how a packet is forwarded between clusters. There are two clusters with black nodes A and B. The respective forwarding nodes in the overlap regions are node C and node D. When node P sends a packet, node A determines if node P is within the range of node C. If it is not, then node A forwards the packet to node C. Otherwise, node C can listen to the packet from node P. Node C forwards the packet in the overlapping region where node D receives it. Note that since node C is in the range of node P, the black node A does not need to forward this packet.

In the assignment without location information, the nodes do not have information about their location. The three-color cluster tree has the property that any parent and child pair is, at the most, one hop away from each other. This means that there is, at most, one intermediate node between any two black nodes.

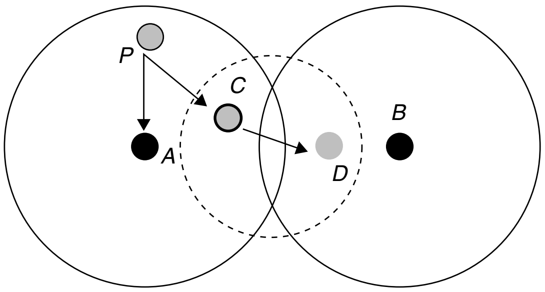

Figure 6.6 Node forwarding between clusters.

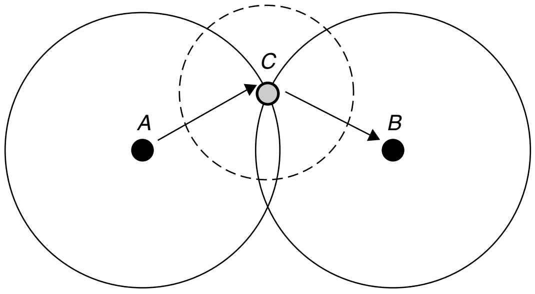

Figure 6.7 Assigning up the duty cycle without location information.

In the algorithm discussed, the locations of black nodes were known, and the actual mid point was calculated. The nodes inside a circular region of radius $R/2$ centered at this point, were considered for forwarding. This is only possible with location information.

In the approach without location information, a circle of radius $R/2$ is centered at the intermediate node between two black nodes. Figure 6.7 illustrates the mechanism.

The intermediate node C sends out a message to set up the forwarding nodes. Nodes within a distance of $R/2$ (shown by dotted circle) from this intermediate node consider themselves for forwarding between a particular pair of clusters. The remaining procedure is exactly the same as the approach with location information.

Due to lack of location information, a black node cannot decide the reachability of packets between forwarding nodes. The black node, instead of forwarding a packet immediately, waits for a random amount of time before forwarding the packet. In the meantime, if the black node hears that the active forwarding node forwarded the same packet, it does not forward this packet.

6.3. ADAPTIVE CLUSTERING WITH DETERMINISTIC CLUSTER-HEAD SELECTION

Reducing the power consumption of wireless microsensor networks increases the network lifetime. A communication protocol LEACH (Low-Energy Adaptive Clustering Hierarchy) can be extended from stochastic cluster-head selection algorithm to include a deterministic component. This way, depending on the network configuration an increase of network lifetime can be accomplished. Lifetime of microsensor networks is defined by using three metrics FND (First Node Dies), HNA (Half of the Nodes Alive), and LND (Last Node Dies).

LEACH is a communication protocol for microsensor networks. It is used to collect data from distributed microsensors and transmit it to a base station. LEACH uses the following clustering-model: some of the nodes elect themselves as cluster-heads, which collect sensor data from other nodes in the vicinity and transfer the aggregated data to the base station. Since data transfers to the base station dissipate much energy, the nodes take turns with the transmission by rotating the cluster-heads, which leads to balanced energy consumption of all nodes and hence to a longer lifetime of the network.

Modification of LEACH's cluster-head selection algorithm reduces energy consumption. For a microsensor network the following assumptions are made:

- the base station (BS) is located far from the sensors and is immobile;
- all nodes in the network are homogeneous and energy constrained;
- all nodes are able to reach the BS;
- nodes have no location information;
- the propagation channel is symmetric;
- cluster-heads perform data compression.

The energy needed for the transmission of one bit of data from node A to node B, is the same as to transmit one bit from node B to node A because of the symmetric propagation channel. Cluster-heads collect nk-bit messages from n adjacent nodes and compress the data to $(c \times n)$ k-bit messages, which are transmitted to the BS, with $c \leqslant 1$ as the compression coefficient.

The operation of LEACH is divided into rounds, each of which consists of a set-up and a steady-state phase. During the set-up phase, cluster-heads are determined and the clusters are organized. During the steady-state phase, data transfers to the base station occur.

LEACH cluster-heads are stochastically selected by each node n determining a random number between 0 and 1. If the number is less than a threshold $T(n)$, the node becomes a cluster-head for the current round.

Considering a single round of LEACH, a stochastic cluster-head selection will not automatically lead to minimum energy consumption during data transfer for a given set of nodes. All cluster-heads can be located near the edges of the network or adjacent nodes can become cluster-heads. In these cases some nodes have to bridge long distances in order to reach a cluster-head. However, considering two or more rounds, a selection of favorable cluster-heads results in an unfavorable cluster-head selection in later rounds, since LEACH tries to distribute energy consumption among all nodes. An example case is shown in Figure 6.8. In the bad-case scenario, cluster-heads

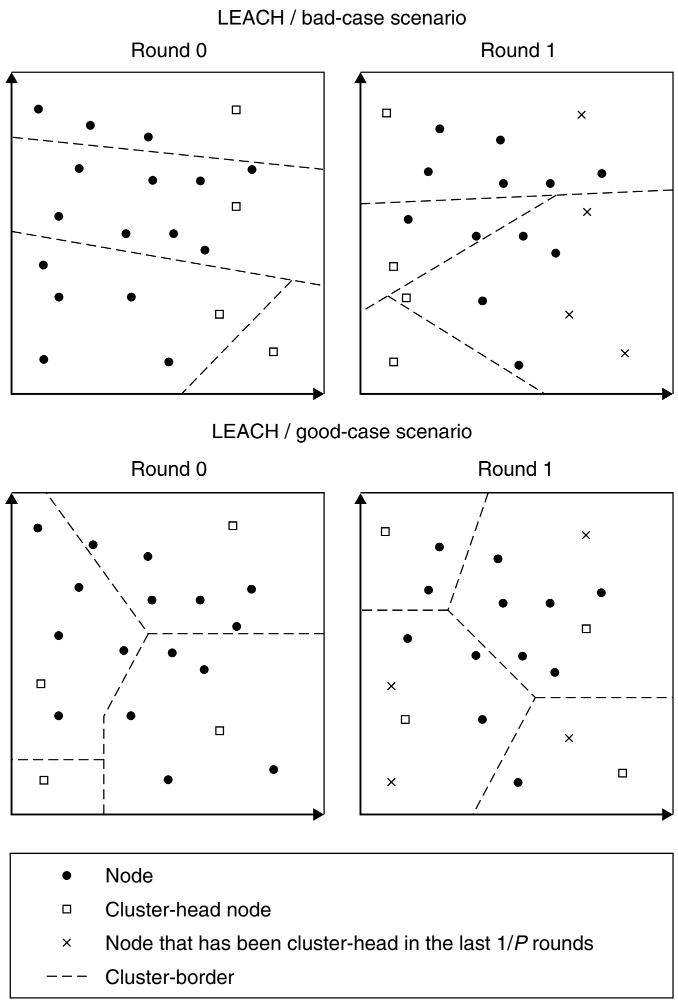
LEACH / bad-case scenario
Round 0
Round 1
LEACH / good-case scenario
Round 0
Round 1
● Node
□ Cluster-head node
× Node that has been cluster-head in the last 1/P rounds
--- Cluster-border

Figure 6.8 LEACH network with $P = 0.2$, $n = 20$, and network dimension of 100×100 meters. Above, cluster-heads are placed in proximity to each other and near the edges which leads to high energy consumption since nodes have to transmit over long distances. Below, energy is saved by uniformly distributing cluster-heads over the network. The set of nodes that have not been cluster-heads in round 0 and 1 is equal for both cases.

are selected unfavorably near the edges, in round 0 on the right-hand side and in round 1 on the left-hand side of the network. In the good-case scenario cluster-heads are not distributed optimally across the network, but better than in the bad-case scenario. A selection of favorable cluster-heads will not automatically lead to a higher energy consumption in later rounds.

A selection of favorable cluster-heads in earlier rounds does not result in an unfavorable cluster-head selection in later rounds. Therefore, energy savings from earlier rounds will not be consumed by higher energy dissipation in later rounds. Regarding energy consumption, a deterministic cluster-head selection algorithm can out perform a stochastic algorithm.

The definition of the lifetime of a microsensor network is determined by the kind of service it provides. Hence, three approaches to defining lifetime are considered. In some cases it is necessary for all nodes to stay alive as long as possible, since network quality decreases considerably as soon as one node dies. Scenarios for this case include intrusion or fire detection. In these scenarios it is important to know when the first node dies. The metric First Node Dies (FND) denotes an estimated value for this event in a specific network configuration. Furthermore, sensors can be placed in proximity to each other. Thus, adjacent sensors could record related or identical data. Hence, the loss of a single or a few nodes does not automatically diminish the quality of service in the network. In this case, the metric Half of the Nodes Alive (HNA) denotes an estimated value for the half-life period of a microsensor network. Finally, the metric Last Node Dies (LND) gives an estimated value for the overall lifetime of a microsensor network.

For a cluster-based algorithm like LEACH, the metric LND is not interesting since more than one node is necessary to perform the clustering algorithm. The discussion of algorithms includes the metrics FND and HNA.

An approach to increasing the lifetime of a LEACH network is to include the remaining energy level available in each node. This can be achieved by reducing the threshold relative to the node's remaining energy. This modification of the cluster-head threshold can increase the lifetime of a LEACH microsensor network by 30 % for FND and by more than 20 % for HNA.

A modification of the threshold equation by the remaining energy has a crucial disadvantage, in that after a certain number of rounds the sensor network cannot perform, although there are still nodes available with enough energy to transmit data to the base station. The reason for this is a cluster-head threshold that is too low, because the remaining nodes have a very low energy level.

A possible solution of this problem is a further modification of the threshold equation, which is expanded by a factor that increases the threshold for any node that has not been cluster-head for a certain number of rounds. The chance of this node becoming a cluster-head increases because of a higher threshold. A possible blockade of the network is solved. This way the data is transmitted to the base station as long as the nodes are alive.

6.4. SENSOR CLUSTERS' PERFORMANCE

The goals of a wireless sensor network are to detect events of interest and estimate parameters that characterize these events. The resulting information is transmitted to one or more locations outside the network. For example, a typical scenario might include a number of sensors spread over an outdoor area for the purpose of determining vehicle traffic. The first step is to determine if there is a vehicle present, and the second step is to classify the type of vehicle. Parameters such as speed, direction, and cargo are of interest. Figure 6.9 shows a conceptual diagram of the three layers in the physical system. The cluster layer is where the collaborative signal processing occurs, while the wireless Mobile *Ad hoc* Network (MANET) is responsible for routing and dissemination of the information. Note that conceptually, the wireless network is larger than the sensor network, because it includes additional nodes.

The issues in designing a sensor network include:

- selection of the collaborative signal processing algorithms run at each sensor node;
- selection of multi-hop networking algorithms, and
- optimal matching of sensor requirements with communications performance.

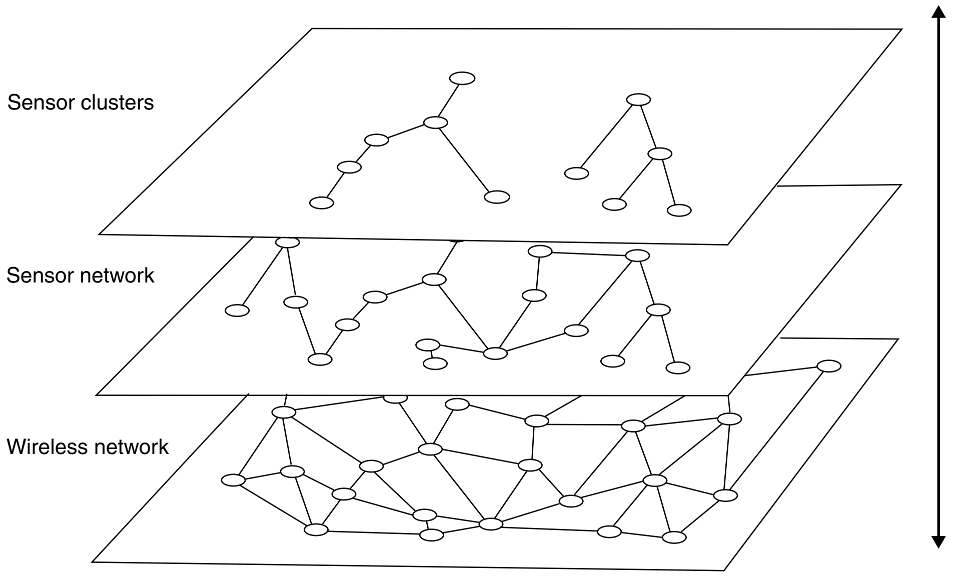

Figure 6.9 The conceptual layers in wireless sensor network.

For military networks, additional issues are:

- low probability of detection and exploitation;
- resistance to jamming, reliability of data;
- latency;
- survivability of the system.

To make the design and optimization efforts tractable, the problem should be decomposed as much as possible. This is done by clearly defining interfaces between the different layers containing the various sensor, networking, and communication processes. Moreover, the wireless sensor network must be coupled with the environment and the target(s); there are two or more transmission media, one for the radio propagation and the other for the propagation of the sensor input (acoustic, seismic, etc.).

To reduce the amount of power spent on long distance radio transmissions, the sensor nodes are aggregated into clusters. This concept is especially useful when the ranges of the sensors are relatively short. During the process of distributed detection, estimation and data fusion, the radio transmissions are among nodes within a cluster, under the control of a cluster-head or master node. While it is quite possible that all the nodes in a cluster are identical, it may be more desirable to provide the cluster-head with more functionality. Location awareness using GPS (Global Positioning System) and a longer-range radio are two useful additions.

Figure 6.10 shows the processing occurring at different layers in the protocol stack for such a cluster-based system. A short range radio (Radio 1) is used to communicate among the sensors in a cluster. The sensor layer is responsible for the collaborative signal processing, which processing can include beamforming, as well as distributed detection, estimation and data fusion. The system operates by using an emitter which generates observations at one or more sensors. In the figure, only node A receives a particular observation. The sensor layer processes the observation and makes a tentative decision, thereby performing data reduction down to a few bits. (For beamforming, either the raw data or a finely quantized version thereof is transmitted instead, requiring significantly more bandwidth.) This information is placed in a very short data packet that is to be sent to all other nodes in the cluster (Nodes B_i and Node C), assumed to be within one hop. Therefore, the packet can bypass the transport and network layers and go directly to the MAC layer for transmission at the appropriate time.

Upon reception of the packet, the other nodes update their tentative decisions. These decisions may then be re-broadcast to all nodes in the cluster. The number of iterations depends on the distributed algorithm, and

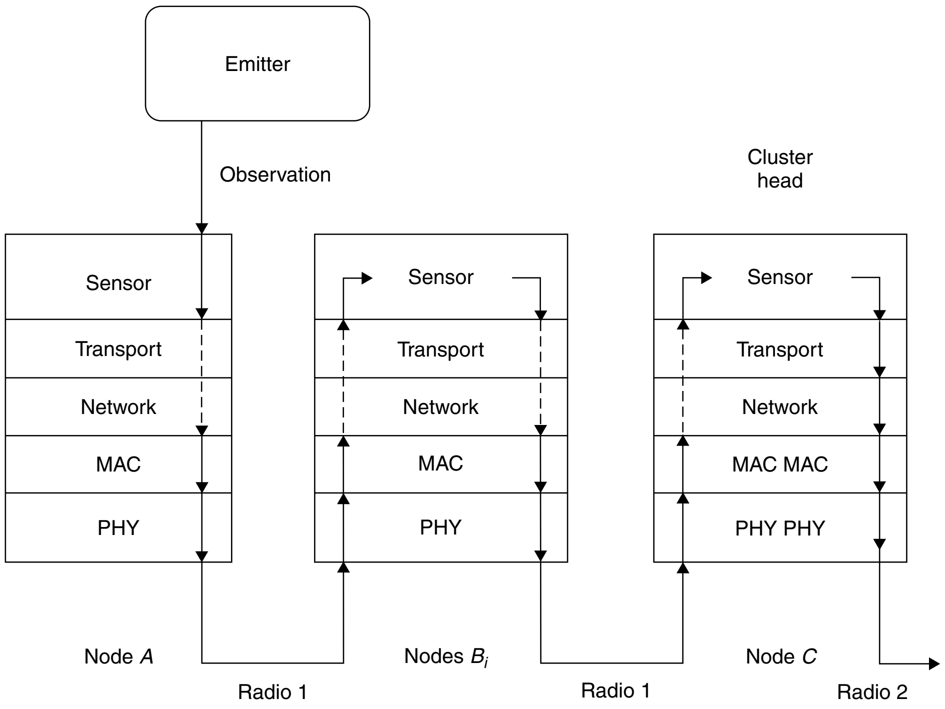

Figure 6.10 The movement of data through the different protocol layers in a cluster-based wireless sensor network.

eventually convergence is achieved. A number of parameters, such as the decision and a confidence measure, now need to be transmitted from the cluster to a remote location using the larger mobile *ad-hoc* network.

A summary packet is generated and sent down to the network layer, as shown in the figure by the solid lines in the right side of node *C*. The network layer uses its routing protocol to select the next hop in the MANET. The network packet is encapsulated by the MAC and transmitted. The actual transmission may use the same radio system as used for cluster-based processing, albeit with increased power and changes in other radio parameters. For example, the virtual subnet approach uses different channels for intracluster and intercluster communications. However, it is also possible to use a completely different radio, as shown in Figure 6.10.

6.4.1. Distributed Sensor Processing

In distributed detection, a number of independent sensors each make a local decision, often a binary one, and then these decisions are combined at a fusion center to generate a global decision. In the Neyman–Pearson formulation, a bound on the global probability of false alarm is assumed, and the goal is to determine the optimum decision rules at the sensors and at the fusion center that maximize the global probability of detection. Alternatively, the Bayesian formulation can be used, and the global probability of error be computed. If the communication network is able to handle the increased load, performance can be improved through the use of decision feedback.

Swaszek and Willet (1995) proposed an extensive feedback approach that they call 'parleying'. The basic idea is that each sensor makes an initial binary decision that is then distributed to all the other sensors. The goal is to achieve a consensus on the given hypothesis through multiple iterations. The algorithm constrains the consensus to be optimum in that it matches the performance of a centralized processor having access to all the data. The main algorithmic performance issue is the number of parleys (iterations) required to reach this consensus. An extension to the parley algorithm uses soft decisions in order to reduce both the number of channel accesses required and the total number of bits transmitted. The parley algorithm leads to the same global decision being made at each node in the cluster.

When classifying a target, the true misclassification probability is the main metric of interest. For any parameter, the maximum likelihood (or maximum *a posteriori*, if possible) estimate of these parameters is desired, along with the variance of the estimate. Additionally, the total energy expended in making the detection decision and doing any parameter estimation and classification is important.

While the use of decision feedback in a sensor cluster can certainly improve performance, there is an additional cost in the complexity of the sensor nodes and a possible increase in transmission energy requirements. Advances in integrated circuitry mitigate the first problem, and the use of short range transmissions helps with the second one. In general, the trend is to put more signal processing in the node in order to reduce the number of transmissions. Cluster-based collaborative signal processing provides a good trade-off between improved performance and low energy use. Within a node, multispectral or multimode sets of colocated sensors, combined in a kind of local data fusion, may be used to improve the performance. This type of data fusion is generally different from the data fusion that may occur at a fusion center (cluster-head).

The overall utility of the sensor network may be improved if each sensor is context-aware, that is, it has some knowledge of its environment. Schmidt *et al.* (1999) studied the use of context awareness for adapting the operating parameters of a GSM (Global Standard for Mobile) cellular phone and a personal digital assistant, and proposed a four-layer architecture. The lowest layer is the sensor layer, which consists of the actual hardware sensors. For each sensor, a number of cues is created. Cues are abstractions of a sensor, and they allow calibration and post-processing; when a sensor is replaced by one of a different type, only the cues must be modified.

Typical cues include:

- the average of the sensor data over a given interval;
- the standard deviation over the same interval;
- distance between the first and third quartiles;
- first derivative of the sensor data.

Multiple sets of contexts can be defined from the cues. For example, a single context is the terrain surrounding the sensor node, such as forest, urban area, open field, etc. Here, the choices are mutually exclusive, but this is not a requirement. Another context is the number of other sensor nodes with direct (single-hop) radio connectivity. A third context is the required level of transmission security or stealth. Determining the cue to context mapping is, in general, a difficult challenge. Once the sensor's context is known, parameters such as transmit power, waveform, distributed detection algorithm, etc., can be set.

Regardless of the application, there are certain critical features that can determine the efficiency and effectiveness of a dedicated network. These features can be categorized into quantitative features and qualitative features. Quantitative features include:

- *Network settling time*: the time required for a collection of mobile wireless nodes to organize itself automatically and transmit the first message reliably.
- *Network join time*: the time required for an entering node or group of nodes to become integrated into the special network.
- *Network depart time*: the time required for the network to recognize the loss of one or more nodes, and reorganize itself to route around the departed nodes.
- *Network recovery time*: the time required for a collapsed portion of the network, due to traffic overload or node failures, to become functional again once the load is reduced or the nodes become operational.
- *Frequency of updates (overhead)*: the number of control packets required in a given period to maintain proper network operation.
- *Memory requirement*: the storage space requirements in bytes, including routing tables and other management tables.
- *Network scalability*: the number of nodes that the dedicated network can scale to and reliably preserve communication.

Qualitative critical features include:

- *Knowledge of nodal locations*: does the routing algorithm require local or global knowledge of the network?
- *Effect of topology changes*: does the routing algorithm need complete restructuring or only incremental updates?
- *Adaptation to radio communication environment*: do nodes use estimated knowledge of fading, shadowing, or multi-user interference on links in their routing decisions?
- *Power consciousness*: does the network employ routing mechanisms that consider the remaining battery life of a node?
- *Single or multichannel*: does the routing algorithm utilize a separate control channel? In some applications, multichannel execution may make the network vulnerable to counter measures.
- *Bidirectional or unidirectional links*: does the routing algorithm perform efficiently on unidirectional links, e.g. if bidirectional links become unidirectional?
- *Preservation of network security*: do routing and MAC layer policies support the survivability of the network, in terms of low probability of detection, low probability of intercept, and security?
- *QoS routing and handling of priority messages*: does the routing algorithm support priority messaging and reduction of latency for delay sensitive real-time traffic? Can the network send priority messages and voice even when it is overloaded with routine traffic levels?
- *Real-time voice and video services*: can the network support simultaneous real-time multicast voice or video while supporting traffic loads associated with situation awareness, and other routine services?

Thread-task level metrics include average power expended in a given time period to complete a thread (task), including power expended in transmitting control messages and information packets, and task completion time. Diagnostic metrics, which characterize network behavior at the packet level, include end-to-end throughput (average successful transmission rate) and delay, average link utilization, and packet loss rate.

The performance of the sensor network depends on the routing of the underlying dedicated network. MANET routing algorithms include the dynamic source routing protocol (DSR) and the *ad-hoc* on-demand distance vector routing protocol (AODV), either of which can be used as basis for the underlying wireless network. Perhaps of more relevance is the zone routing protocol

(ZRP), which is a hybrid of proactive and reactive routing protocols. This means that the network is partitioned into zones, and the routes from a node to all other nodes in its zone are determined. Routes to nodes in other zones are found as needed. ZRP may allow the sensor network to implement decision feedback among all nodes in a zone in a straightforward manner.

As an example, consider the sensor network shown in Figure 6.11. The sensors have been placed along the roads, with the greatest concentration at the fork. The Linked Cluster Algorithm (LCA) was used to self-organize the network, leading to the creation of four clusters. Clusters 1 and 2 overlap, as

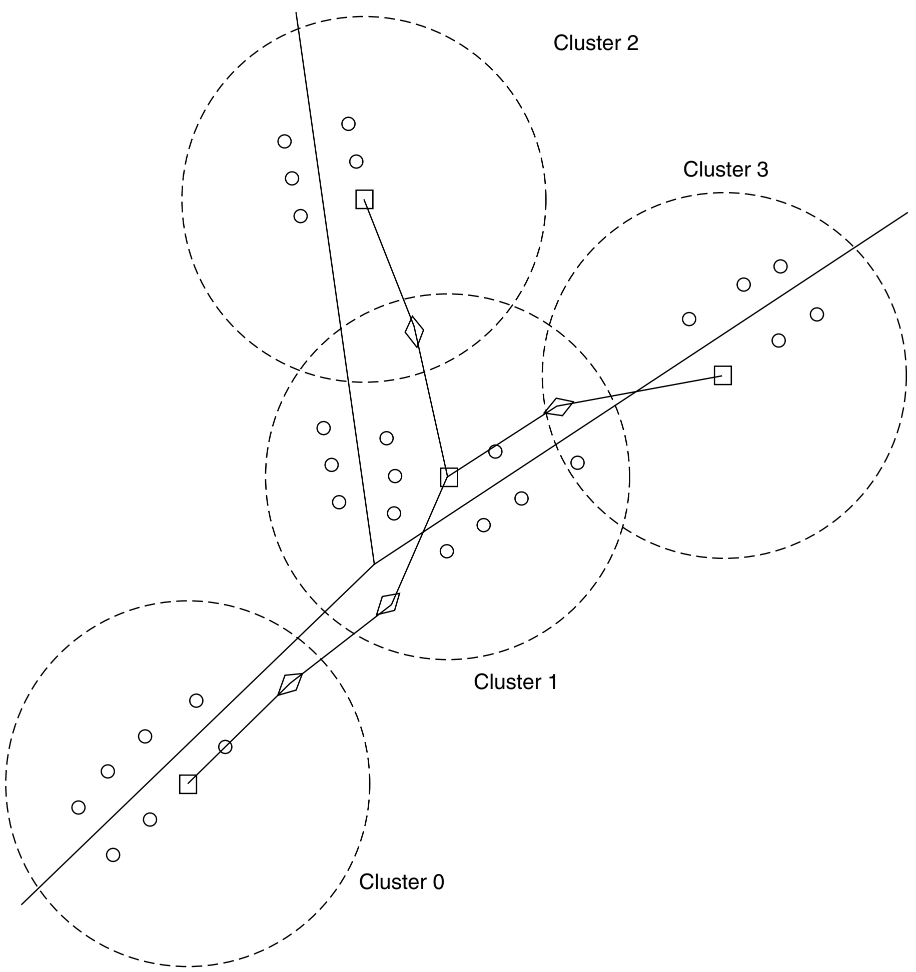

Figure 6.11 Self-organized sensor network. The cluster heads are squares, the gateway nodes are diamonds, and the ordinary sensor nodes are circles. The transmission areas of the four cluster heads are indicated by the four large circles.

do clusters 1 and 3, so only a single gateway node is used to connect each pair. Since clusters 0 and 1 do not overlap, a pair of gateways is created; the resulting backbone network that connects the cluster heads is shown in the illustration.

The numbering of the nodes in the LCA determines which nodes become cluster-heads and gateways. Since the initial topology is known, four specific nodes are assigned the highest node numbers, thereby ensuring that they would become cluster-heads. Essentially, by choosing the cluster-heads in advance, the clusters have shapes that are well suited to collaborative signal processing. To decompose a cluster further into subclusters, for example, cluster 1 could easily be divided into two or three sensor groups for the purpose of distributed detection. Once a decision is reached in a subcluster, it would be sent to the cluster-head for dissemination.

6.5. POWER-AWARE FUNCTIONS IN WIRELESS SENSOR NETWORKS

The design of micropower wireless sensor systems has gained increasing importance for a variety of civil and military applications. With advances in MEMS technology and its associated interfaces, signal processing, and RF circuitry, the focus has shifted away from limited macrosensors communicating with base stations, to creating wireless networks of communicating microsensors that aggregate complex data to provide rich, multidimensional pictures of the environment. While individual microsensor nodes are not as accurate as their macrosensor counterparts, the networking of a large number of nodes enables high quality sensing networks with the additional advantages of easy deployment and fault tolerance. These are the characteristics that make microsensors ideal for deployment in otherwise inaccessible environments, where maintenance would be inconvenient or impossible.

The unique operating environment and performance requirements of distributed microsensor networks require fundamentally new approaches to system design. As an example, consider the expected performance versus longevity of the microsensor node, compared with battery-powered portable devices. The node, complete with sensors, DSP (Digital Signal Processing), and radio, is capable of a tremendous diversity of functionality. Throughout its lifetime, a node may be called upon to be a data gatherer, a signal processor, and a relay station. Its lifetime, however, must be of the order of months to years, since battery replacement for thousands of nodes is not an option. In contrast, much less capable devices, such as cellular telephones, are only expected to run for days on a single battery charge. High diversity also exists within the environment and user demands upon the sensor network. Ambient noise in the environment, the rate of event arrival, and the user's quality requirements of the data may vary considerably over time.

A long node lifetime under diverse operating conditions demands power-aware system design. In a power-aware design, the node's energy consumption displays a graceful scalability in energy consumption at all levels of the system hierarchy, including the signal processing algorithms, operating system, network protocols, and even the integrated circuits themselves. Computation and communication are partitioned and balanced for minimum energy consumption. Software that understands the energy–quality tradeoff collaborates with hardware that scales its own energy consumption accordingly.

Once the power-aware microsensor nodes are incorporated into the framework of a larger network, additional power-aware methodologies emerge at the network level. Decisions about local computation versus radio communication, the partitioning of computation across nodes, and error correction on the link layer offer a diversity of operational points for the network.

A network protocol layer for wireless sensors allows for sensor collaboration. Sensor collaboration is important for two reasons. First, data collected from multiple sensors can offer valuable inferences about the environment. For example, large sensor arrays have been used for target detection, classification and tracking. Second, sensor collaboration can provide trade-offs in communication versus computation energy. Since it is likely that the data acquired from one sensor are highly correlated with data from its neighbors, data aggregation can reduce the redundant information transmitted within the network. When the distance to the base station is large, there is a large advantage in using local data aggregation (e.g. beamforming) rather than direct communication. Since wireless sensors are energy constrained, it is important to exploit such trade-offs to increase system lifetimes and improve energy efficiency.

The energy-efficient network protocol LEACH (Low Energy Adaptive Clustering Hierarchy) utilizes clustering techniques that greatly reduce the energy dissipated by a sensor system. In LEACH, sensor nodes are organized into local clusters. Within the cluster is a rotating cluster-head. The cluster-head receives data from all other sensors in the cluster, performs data aggregation, and transmits the aggregate data to the end-user. This greatly reduces the amount of data that is sent to the end-user for increased energy efficiency. LEACH can achieve reduction in energy of up to a factor of 8 over conventional routing protocols such as multi-hop routing. However, the effectiveness of a clustering network protocol is highly dependent on the performance of the algorithms used for data aggregation and communication.

It is important to design and implement energy-efficient sensor algorithms for data aggregation and link-level protocols for the wireless sensors.

Beamforming algorithms is one class of algorithms that can be used to combine data. Beamforming can enhance the source signal and remove uncorrelated noise or interference. Since many types of beamforming algorithms exist, it is important to make a careful selection based upon their computation energy and beamforming quality.

Algorithm implementations for a sensor network can take advantage of the network's inherent capability for parallel processing to reduce energy further. Partitioning a computation among multiple sensor nodes and performing the computation in parallel permits a greater allowable latency per computation, allowing energy savings through frequency and voltage scaling.

As an example, consider a target tracking application that requires sensor data to be transformed into the frequency domain through 1024-point FFT (Fast Fourier Transform). The FFT (Fast Fourier Transform) results are phase shifted and summed in a frequency-domain beamformer to calculate signal energies in 12 uniform directions, and the Line-Of-Bearing (LOB) is estimated as the direction with the most signal energy. By intersecting multiple LOBs at the base station, the source's location can be determined. Figure 6.12 demonstrates the tracking application performed with traditional clustering techniques for a seven sensor cluster. The sensors (S1–S6) collect data and transmit the data directly to the cluster-head (S7), where the FFT, beamforming and LOB estimation are performed. Measurements on the SA-1100 at an operating voltage of 1.5 V and frequency of 206 MHz show that the tracking application dissipates 27.27 mJ of energy.

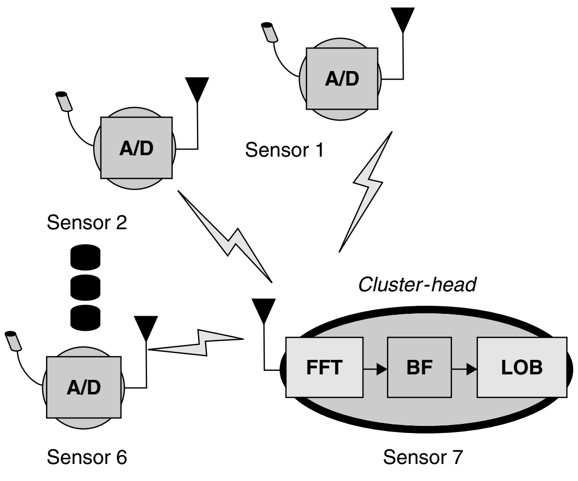

Figure 6.12 Approach 1: All computation is done at the cluster-head.

Distributing the FFT computation among the sensors reduces energy dissipation. In the distributed processing scenario of Figure 6.13, the sensors collect data and perform the FFTs before transmitting the FFT results to the cluster-head. At the cluster-head, the FFT results are beamformed and the LOB estimate is found. Since the seven FFTs are done in parallel, the supply voltage and frequency can be reduced without sacrificing latency. When the FFTs are performed at 0.9 V, and the beamforming and LOB estimation at the cluster-head are performed at 1.3 V, then the tracking application dissipates 15.16 mJ, a 44 % improvement in energy dissipation.

Energy–quality trade-offs appear at the link layer as well. One of the primary functions of the link layer is to ensure that data is transmitted reliably. Thus, the link layer is responsible for some basic form of error detection and correction. Most wireless systems utilize a fixed error correction scheme to minimize errors and may add more error protection than necessary to the transmitted data. In a energy-constrained system, the extra computation becomes an important concern. Thus, by adapting the error correction scheme used at the link layer, energy consumption can be scaled while maintaining the Bit Error Rate (BER) requirements of the user.

Error control can be provided by various algorithms and techniques, such as convolutional coding, BCH (Bose–Chaudhuri–Hocquenghem) coding, and turbo coding. The encoding and decoding energy consumed by the various algorithms can differ considerably. As the code rate increases, the algorithm's energy also increases. Hence, given bit error rate and latency requirements, the lowest power FEC (Forward Error Control) algorithm that satisfies these needs should continuously be chosen. Power consumption can

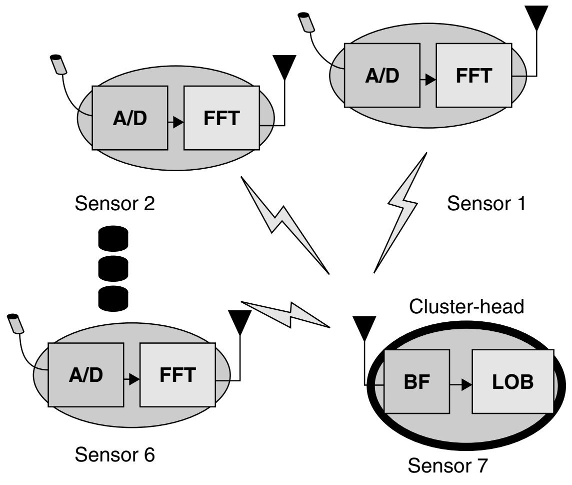

Figure 6.13 Approach 2: Distribute the FFT computation among all sensors.

be further reduced by controlling the transmit power of the physical radio. For a given bit error rate, FEC lowers the transmit power required to send a given message. However, FEC also requires additional processing at the transmitter and receiver, increasing both the latency and processing energy. This is another computation versus communication trade-off that divides available energy between the transmit power and coding processing to best minimize total system power.

6.5.1. Power Aware Software

The overall energy efficiency of wireless sensor networks crucially depends on the software that runs on them. Although dedicated circuits can be substantially more energy efficient, the flexibility offered by general purpose processors and DSPs have engineered a shift towards programmable solutions. Power consumption can be substantially reduced by improving the control software and the application software.

The embedded operating system can dynamically reduce system power consumption by controlling shutdown, the powering down of all or parts of the node when no interesting events occur, and dynamic voltage scaling. Dynamic power management using node shutdown, in general, is a nontrivial problem. The sensor node consists of different blocks, each characterized by various low power modes and overheads to transition to them. The node sleep states are a combination of various block shutdown modes. If the overheads in transitioning to sleep states were negligible, then a simple greedy algorithm could make the system go into the deepest sleep state as soon as it is idle. However, in reality, transitioning to a sleep state and waking up has a latency and energy overhead. Therefore, implementing the right policy for transitioning to the available sleep states is critical.

It is highly desirable to structure the algorithms and software such that computational accuracy can be traded off with energy consumption. Transforming software such that most significant computations are accomplished first improves the energy–quality scalability can be improved. Consider an example of a sensor node performing an FIR filtering operation. If the energy availability to the node were reduced, the algorithm may be terminated early to reduce the computational energy. In an unscalable software implementation, this would result in severe quality degradation. By accumulating the partial products corresponding to the most significant coefficients first (by sorting them in decreasing order of magnitude), the scalable algorithm produces far more accurate results at lower energies.

An application programming interface is an abstraction that hides the underlying complexity of the system from the end-user. Hence, a wireless sensor network API is a key enabler in allowing end-users to manage the tremendous operational complexity of such networks. While end-users are experts in their respective application domains (say, remote climate monitoring), they are not necessarily experts in distributed wireless networking and do not wish to be bothered with the internal network operation. By defining high-level objects, a functional interface and the associated semantics, APIs make the task of application development significantly easier.

An API consists of a functional interface, object abstractions, and detailed behavioral semantics. Together, these elements of an API define the ways in which an application developer can use the system. Key abstractions in a wireless sensor network API are the nodes, base station, links, messages, etc. The functional interface itself is divided into the following:

- functions that gather the state (of the nodes, part of a network, a link between two nodes, etc.);
- functions that set the state (of the nodes, of a cluster or the behavior of a protocol);
- functions that allow data exchange between nodes and the base station;
- functions that capture the desired operating point from the user at the base station;
- functions that help visualize the current network state;
- functions that allow users to incorporate their own models (for energy, delay, etc.).

An API is much more than the sum of its functional interface and object abstractions. This is because of the (often implicit) application development paradigm associated with it. The API is especially crafted to promote application development based on certain philosophies which the designers of the network consider to be optimal in the sense of correctness, robustness and performance. For example, a good overall application framework for wireless sensor networks is the Get-Optimize-Set paradigm. This paradigm basically implies collecting the network state, using this state information along with the knowledge of the desired operating point to compute the new optimal state, and then setting the network to this state. The entire application code is based on this template.

Power aware computation and communication is the key to achieving long network lifetimes due to the energy constrained nature of the nodes. An important responsibility of the API is not only to allow the end-user to construct the system in a power-aware manner but also to encourage such an approach. For starters, functions in a high quality network API have explicit energy, quality, latency and operating point annotations. Hence, instead of demanding a certain function from the network, one can demand a certain function subject to constraints (energy, delay, quality, etc.). Next, the API has basic energy modeling allowing the end-user to calibrate the energy efficiency of the various parts of the application. For users requiring models beyond the level of sophistication that the API offers, there are modeling interfaces that allow users to register arbitrarily complex models. Next, a good wireless sensor API allows what have come to be known in the software community as thick and thin clients. These adjectives refer to the complexity and overhead of typical application layers. Finally, the Get-Optimize-Set paradigm promulgated by the API allows the network to beat the optimal operating point thus enhancing energy efficiency.

6.6. EFFICIENT FLOODING WITH PASSIVE CLUSTERING

Clustering and route aggregation techniques have been proposed to reduce the flooding overhead. These techniques operate in a proactive, background mode. They use explicit control packets to elect a small set of nodes (cluster-heads, gateways or flood-forwarding nodes), and restrict to such a set the flood forwarding function. These proactive schemes cause traffic overhead in the network.

A flooding mechanism based on passive, on-demand clustering reduces flooding overhead without loss of network performance. Passive clustering is an on-demand protocol which dynamically partitions the network into clusters interconnected by gateways. Passive clustering exploits data packets for cluster formation, and is executed only when there is user data traffic. Passive clustering has the following advantages compared with active clustering and route aggregation techniques.

(1) Passive clustering eliminates cluster set-up latency and extra control overhead (by exploiting on-going packets).
(2) Passive clustering uses an efficient gateway selection heuristic to elect the minimum number of forwarding nodes (thus reducing superfluous flooding).
(3) Passive clustering reduces node power consumption by eliminating the periodic, background control packet exchange.

Multi-hop *ad hoc* networks (MANETs) are self-creating, self-organizing and self-administrating without deploying any kind of infrastructure. They offer special benefits and versatility for wide applications in military (e.g. battlefields, sensor networks, etc.), commercial (e.g. distributed mobile computing, disaster discovery systems, etc.), and educational (e.g. conferences, conventions, etc.) environments, where fixed infrastructure is not easily acquired. With the absence of pre-established infrastructure (e.g. no router, no access point, etc.), two nodes communicate with one another in a peer-to-peer fashion. Two nodes communicate directly if they are within transmission range of each other. Otherwise, nodes can communicate via a multi-hop route with the cooperation of other nodes. To find such a multi-hop path to another node, each MANET node widely use flooding or broadcast (e.g. hello messages). Many *ad hoc* routing protocols, multicast schemes, or service discovery programs depend on massive flooding.

In flooding, a node transmits a message to all neighbors. The neighbors in turn relay the message to their neighbors until the message has been propagated to the entire network. This is blind flooding with performance related to the average number of neighbors (neighbor degree) in the CSMA/CA network. As the neighbor degree becomes higher, the blind flooding suffers from the increases in:

- redundant and superfluous packets;
- the probability of collision, and
- congestion in wireless medium.

When topology or neighborhood information is available, only a subset of neighbors is required to participate in flooding to guarantee the complete flooding of the network. This is efficient flooding. The characteristics of MANETs (e.g. node mobility, the limited bandwidth and resource), however, make collecting topological information very difficult. It generally needs extra overhead due to the periodic message exchanges or event driven updates with optional deployment of GPS (Global Positioning System). For this reason, many on-demand dedicated routing schemes and service discovery protocols use blind flooding. With periodic route table exchanges, proactive *ad-hoc* routing schemes, unlike on-demand routing methods, can gather topological information without a significant overhead (through piggybacking topology information or learning neighbors). Thus, a few proactive *ad hoc* routing mechanisms use route aggregation methods so that the route information is propagated by only a subset of nodes in the network.

Passive clustering is an efficient flooding suitable for on-demand protocols, and does not require the deployment of GPS or explicit periodic control messages. This scheme has several contributions compared with other efficient flooding mechanisms (such as multipoint relay, neighbor coverage, etc.) as follows:

(1) Passive clustering does not need any periodic messages, instead, it exploits existing traffic to piggyback its small control messages. Based on passive clustering technique, it is very resource efficient regardless of the degree of neighbor nodes or the size of the network. Passive clustering provides scalability and practicality for choosing the minimal number of forwarding nodes in the presence of dynamic topology changes. Therefore, it can be easily applied to on-demand routing schemes to improve the performance and scalability.
(2) Passive clustering does not have any set-up latency, and it saves energy with no traffic.
(3) Passive clustering maintenance is well adaptive to dynamic topology and resource availability changes.

The problem of finding a subset of dominant forwarding nodes in MANETs is NP-complete. Thus, the work on efficient flooding focuses on developing efficient heuristics that select a suboptimal dominant set with low forwarding overhead.

There are several heuristics to reduce rebroadcasts. Upon receiving a flooding packet, a node decides whether it relays the packet to its neighbor or not, by using one of following heuristics:

- probabilistic scheme where this node rebroadcasts the packet with the randomly chosen probability;
- counter-based scheme where this node rebroadcasts if the number of received duplicate packets is less than a threshold;
- distance-based scheme that uses the relative distance between hosts to make the decision;
- location-based scheme based on pre-acquired location information of neighbors;
- cluster-based scheme where only cluster-heads and gateways forward the packet.

The passive clustering is different from those ideas in that it provides a platform of efficient flooding based on locally collected information (e.g. neighbor information, cluster states, etc.). Each node participates in flooding based on the role or state in the cluster structure.

Another approach to efficient flooding is to exploit topological information. With the node mobility and the absence of pre-existing infrastructure in the *ad-hoc* network, all works use the periodic hello message exchange method to collect topological information. Passive clustering does not require periodic control messages to collect topological information. Instead, it exploits ongoing data packets to exchange cluster-related information.

The two schemes are called self pruning and dominant pruning. Self pruning is similar to the neighbor-coverage scheme. With self-pruning schemes, each forwarding node piggybacks the list of its own neighbors on the outgoing packet. A node rebroadcasts (becomes a forwarding node) only when this node has neighbors not covered by forwarding nodes. While the self-pruning heuristic utilizes information of directly connected neighbors only, the dominant-pruning scheme extends the range of neighbor information to two-hop-away neighbors. The dominant-pruning scheme is similar to Multipoint Relay (MPR) scheme in which a node periodically exchanges a list of adjacent nodes with its neighbors so that each node can collect the information of two-hop-away neighbors. Each node, based on the gathered information, selects the minimal subset of forwarding neighbors, which covers all neighbors within two hops. Each sender piggybacks its chosen forwarding nodes (MPRNs) onto the outgoing broadcast packet. Moreover, based on topological information, many schemes choose a dominant set. They still depend on the periodic hello messages to collect topological information. The extra hello messages, however, consume resources and drop the network throughput in MANETs. The extra traffic brings about congestion and collision as geographic density increases. The collision probability of hello messages in a single-hop and a two-hop network as the number of neighbors increases shows that the neighbor degree increases the collision probability of broadcast (the collision probability is more than 0.1 with more than 15 neighbors), and hidden terminals aggravate the collision in the multi-hop network. We assume there is no other traffic except for hello messages in the network. With user-data packets, the collision probability of hello messages increases. Thus, it is very difficult to collect the complete neighbor topology using hello messages.

These schemes (e.g. neighbor-coverage, MPR, etc.) are not scalable to offered load and the number of neighbors.

Clustering selects forwarding nodes, and groups nodes into clusters. A representative of each group (cluster) is named a cluster-head and a node belonging to more than two clusters at the same time is a gateway. Other members are the ordinary nodes. The transmission area of the cluster-head defines a cluster. Two-hop clustering is used where any node in a cluster can reach any other node in the same cluster with, at most, two hops. With clustering, nonordinary nodes can be the dominant forwarding nodes as shown in Figure 6.14.

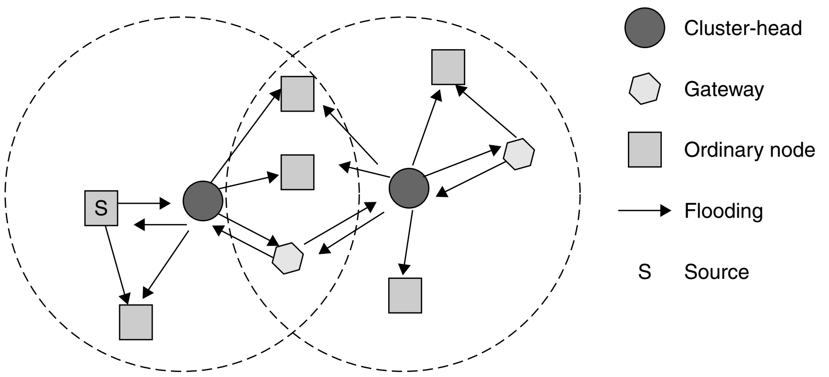

Figure 6.14 An example of efficient flooding with clustering. Only cluster-heads and gateways rebroadcast and ordinary nodes stop forwarding.

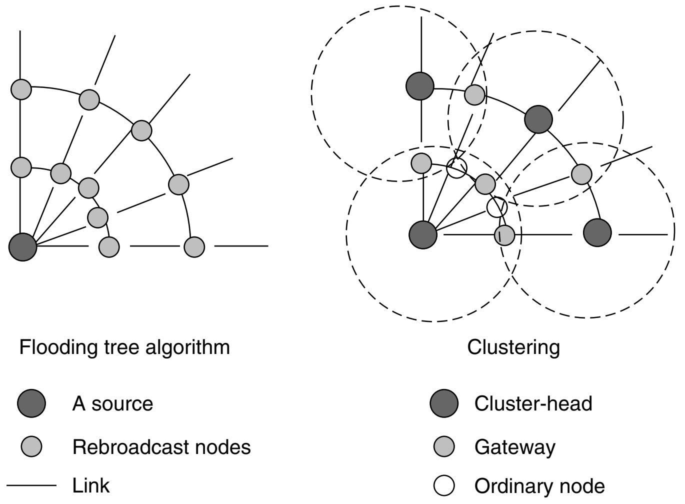

Figure 6.15 In flooding tree algorithms, every neighbor of a source has to rebroadcast since each neighbor is, at most, one adjacent node of some node. In clustering, however, ordinary nodes are not forwarding nodes.

Figure 6.15 illustrates the difference between clustering and the MPR scheme. Clustering partitions the network into several groups based on the radio range of a cluster head. The network topology, therefore, does not have a serious impact on the clustering performance. MPR, on the other hand, chooses the dominant set using topological information so that the performance of MPR is closely related to the network topology.

Clustering in *ad-hoc* networks includes hierarchical routing schemes, the master election algorithms, power control, reliable broadcast, and efficient broadcast. The cluster architecture has also been used for efficient flooding. Some clustering schemes are based on the complete knowledge of the neighbors. However, the complete knowledge of neighbor information in such networks is difficult to collect and requires a control overhead caused by periodic exchanges of hello messages. The clustering algorithms use a large number of gateways in the dense network and do not use a gateway reduction mechanism to select a minimal number of gateways. The clustering incurs a maintenance cost in case of a high mobility.

The three important observations are as follows:

(1) The selection mechanism to choose the dominant set should be efficient and dynamic. Otherwise, the scheme cannot be used effectively and practically.
(2) In a MANET, collecting accurate topological information is very difficult and carries an overhead.
(3) Clustering scheme is independent of the network topology unlike the route aggregation protocols (e.g. MPR).

6.6.1. Passive Clustering

Passive clustering is an on-demand protocol. It constructs and maintains the cluster architecture only when there are on-going data packets that piggyback cluster-related information (e.g. the state of a node in a cluster, the IP (Internet Protocol) address of the node). Each node collects neighbor information through packet receptions. Passive clustering, therefore, eliminates set-up latency and major control overhead of clustering protocols.

Passive clustering has the following mechanisms for the cluster formation:

- First Declaration Wins rule, and
- Gateway Selection Heuristic.

With the First Declaration Wins rule, a node that first claims to be a cluster-head rules the remaining nodes in its clustered area (radio coverage). There is no waiting period (to make sure all the neighbors have been checked) unlike that in all the weight-driven clustering mechanisms. The Gateway Selection Heuristic provides a procedure to elect the minimal number of gateways (including distributed gateways) required to maintain the connectivity in a distributed manner.

Passive clustering maintains clusters using implicit timeout. A node assumes that some nodes are out of locality if they have not sent any data for longer than timeout duration. With reasonable offered load, a node can respond to dynamic topology changes.

When a node joins the network, it sets the cluster state to *initial*. Moreover, the state of a floating node (a node that does not belong to a cluster yet) also sets to *initial*. Because passive clustering exploits on-going packets, the implementation of passive clustering resides between layers 3 and 4.

The IP option field for cluster information is as follows:

- Node ID (identifier) is the IP (Internet Protocol) address of the sender node. This is different from the source address of the IP packet;
- state of the cluster is the cluster state of the sender node;
- if a sender node is a gateway, then it tags two IP addresses of cluster heads which are reachable from the gateway;

The passive clustering algorithm is as follows:

- Cluster states. There are six possible states; *initial, cluster-head, ordinary node, gateway, cluster-head gateway, gateway ready*, and *distributed gateway*.
- The packet handling. Upon sending a packet, each node piggybacks cluster-related information. Upon a packet reception, each node extracts cluster-related information of neighbors and updates the neighbor information table.
- A cluster-head declaration is done by a node in *initial* state which changes its state to *cluster-head ready* (a candidate cluster-head) when a packet arrives from another node that is not a cluster-head. With outgoing packet, a *cluster-head ready* node can declare as a cluster-head. This helps the connectivity because it reduces isolated clusters.
- A node becomes a member of a cluster once it has heard or overheard a message from any cluster head. A member node can serve as a gateway or an ordinary node depending on the collected neighbor information. A member node can settle as an ordinary node only after it has learned enough neighbor gateways. In passive clustering, however, the existence of a gateway can be found only by overhearing a packet from that gateway. Thus, another internal state is *gateway ready*, for a candidate gateway node that has not yet discovered enough neighbor gateways. A gateway selection mechanism is developed to reduce the total gateways in the network. A candidate gateway finalizes its role as a gateway upon sending a packet (announcing the gateway's role). A candidate gateway node can become an ordinary node any time with the detection of enough gateways.

A gateway is a bridge node that connects two adjacent clusters. Thus, a node that belongs to more than two clusters at the same time is eligible to be a gateway. Only one gateway is needed for each pair of two adjacent clusters. The gateway selection mechanism allows only one gateway for each pair of two neighboring cluster-heads. However, it is possible that there is no potential gateway between two adjacent clusters, that is, two cluster-heads are not mutually reachable via a two-hop route. If there is a three-hop route between two nodes, then the clustering scheme selects those intermediate nodes as distributed gateways. Without the knowledge of complete two-hop neighbors' information, choosing a minimal number of distributed gateways is difficult. Topological knowledge carries an overhead and works inefficiently, thus, a counter-based distributed gateway selection mechanism is considered.

The gateway selection mechanism can be summarized as follows:

- Gateway means that a node belonging to more than two clusters at the same time becomes a candidate gateway. Upon sending a packet, a potential gateway chooses two cluster-heads from among known cluster-heads. This node will serve as an intermediate node between those cluster-heads. This node cannot be an intermediate node of two cluster-heads that were announced by another neighbor gateway node. If the node finds two cluster-heads, then it finalizes its role as a gateway and announces two cluster-heads to neighbors. If a gateway has received a packet from another gateway that has announced the same pair of cluster-heads, then this node compares the node ID of itself with that of the sender. If this node has the lower ID, it keeps its role as the gateway. Otherwise, it chooses a different pair of cluster-heads or changes its state. If this node can find another pair of neighbor cluster-heads that is not announced by any other gateway, then it keeps its state as *gateway* for the new pair of cluster-heads, otherwise it changes its state to *ordinary node*.
- Passive clustering allows one distributed gateway for each cluster-head and each node. A node that belongs to only one cluster can be an ordinary node when at least two (distributed) gateways are known to this node. Otherwise, it keeps the candidate gateway state. A candidate gateway node can be a distributed gateway if there is no neighbor-distributed gateway that also belongs to the same cluster. If an ordinary node has received a packet from a distributed gateway and no gateway is a neighbor node of that node, then this node changes to a distributed gateway.

Figure 6.16 shows an example of cluster architecture developed by passive clustering. With moderate on-going traffic, passive clustering allows only one gateway for each pair of clusters and enough distributed gateway nodes.

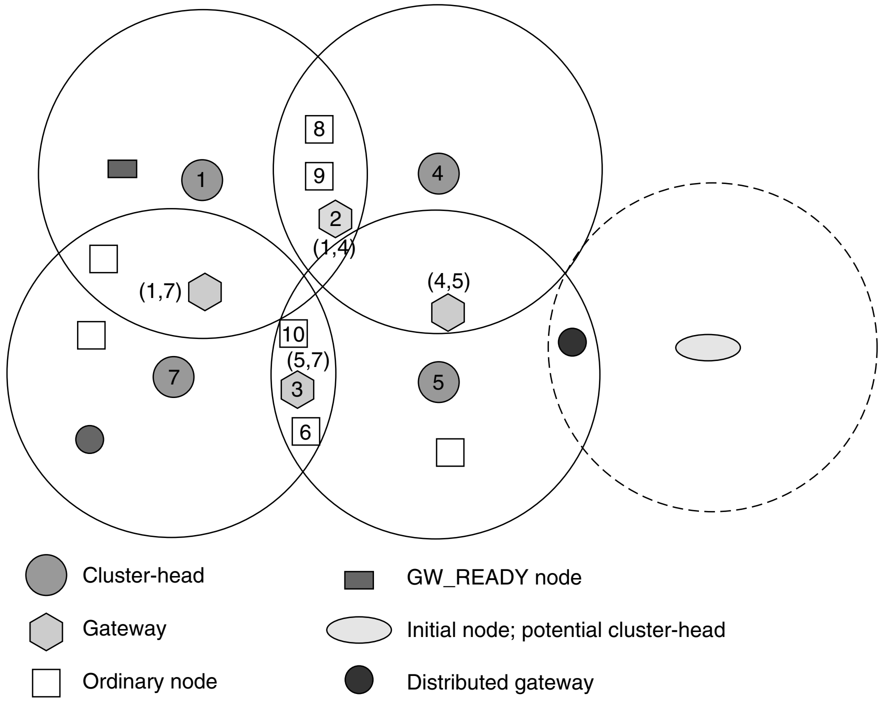

Figure 6.16 An example of a gateway selection heuristic. There is at most one gateway between any pair of two cluster-heads. A gateway can survive only when this node is the only gateway for an announced pair of cluster-heads or this node has the lowest ID among contention gateways (who announced the same pair of cluster heads).

The overhead and flooding efficiency of passive clustering needs to be analyzed. For the message overhead, passive clustering adds 8 bytes or 16 bytes to each outgoing packet. In analysis control, message overhead is considered, as the number of messages is more important than the size of each packet in dedicated networks using IEEE 802.11 DCF protocol.

Passive clustering mechanisms are more efficient than distributed tree algorithms in respect of processing overhead. The computational overhead of passive clustering is $O(Avg\_Neighbor)$ where $Avg\_Neighbor$ denotes the number of active neighbors. Upon receiving a packet, each node updates its neighbor table and changes its state if necessary. A cluster-head only updates its neighbor table. A member node, in addition, adjusts its state based on gateway selection heuristic. Each node computes with $O(Avg\_Neighbor)$ computational complexity upon receiving a packet. With an outgoing packet, each node simply piggybacks cluster-related information. The complexity is $O(1)$.

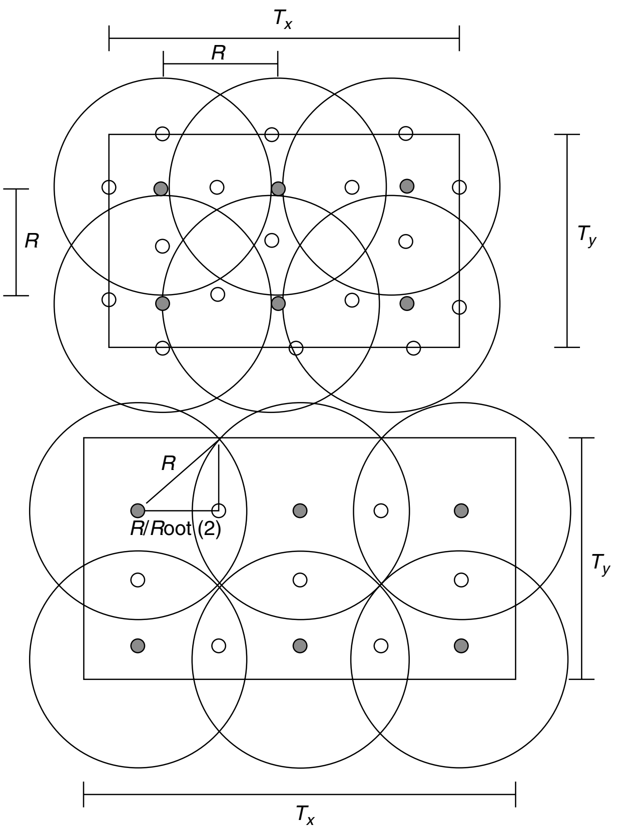

Figure 6.17 The average and most dense case of cluster architecture.

Passive clustering divides nodes into several groups based on the transmission range of the representative node (cluster-head). Thus, the number of forwarding nodes is stable regardless of the geographical density of the network. The reduction rate improves in proportion to the geographical density.

Figure 6.17 illustrates the most dense and average case of cluster construction with the assumption that there are infinite number of nodes placed randomly, and the network size is ($T_x \times T_y$) where T_x is the horizontal size and T_y is the vertical size of the network area.

6.7. SUMMARY

Routing and data dissemination in sensor networks requires a simple and scalable solution.

The topology discovery algorithm for wireless-sensor networks selects a set of distinguished nodes, and constructs a reachability map based on their information. The topology discovery algorithm logically organizes the network in the form of clusters and forms a tree of clusters rooted at the monitoring node. We discussed the applications of tree of clusters for efficient data dissemination and aggregation, duty-cycle assignments and network-state retrieval. The topology discovery algorithm is completely distributed, uses only local information, and is highly scalable.

To achieve optimal performance in a wireless sensor network, it is important to consider the interactions among the algorithms operating at the different layers of the protocol stack. While there has been much research on partitioning a MANET into clusters, most of this work has focused on doing so for routing and resource allocation purposes. For sensor networks, a key addition is how the self-organization of the network into clusters affects the sensing performance.

Distributed microsensor networks hold great promise in applications ranging from medical monitoring and diagnosis to target detection, home automation, hazard detection, and automotive and industrial control. However, even within a single application, the tremendous operational and environmental diversity inherent to the microsensor network demand an ability to make trade-offs between quality and energy dissipation. Hooks for energy–quality scalability are necessary not only at the component level, but also throughout the node's algorithms and the network's communication protocols. Distributed sensor networks designed with built-in power awareness and scalable energy consumption will achieve maximal system lifetime in the most challenging and diverse environments.

Passive clustering can reduce redundant flooding with negligible extra protocol overhead. Moreover, passive clustering can be applied to reactive, on-demand routing protocols with substantial performance gains.

Performance of blind flooding is severely impaired especially in large and dense networks.

PROBLEMS

Learning Objectives

After completing this chapter you should be able to:

- demonstrate understanding of the clustering techniques in wireless sensor networks;