J. Indian Inst. Sci., Nov.–Dec. 2006, **86**, 569–598
© Indian Institute of Science.

A survey of routing algorithms for wireless sensor networks

N. NARASIMHA DATTA* AND K. GOPINATH
Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India.
email: ndatta@gmail.com, gopi@csa.iisc.ernet.in

Received on May 14, 2005; Revised on May 5, 2006.

Abstract

Distributed wireless sensor networks consist of a large number of small, low-cost and low-power nodes (called motes) that coordinate with one another for environmental sensing. The sensor nodes are severely restricted in power, memory and computational resources. The nodes can be densely deployed in close proximity to the phenomenon to be observed. They can be deployed in hostile environments where the nodes may not be physically accessible and are subject to tampering. Nodes can be added to and deleted from the network at any time, resulting in unpredictable changes to the topology of the network. This presents new challenges in the design of routing protocols for sensor networks. In this paper, the constituent building blocks of sensor network routing protocols are identified and analyzed. The routing protocols are broadly classified into two categories: flat and hierarchical, and further into subcategories based on the centrality of their theme. Several routing algorithms belonging to each category that have been proposed in the literature are explored. The techniques used to achieve convergence and to eliminate routing loops are highlighted. Further the open problems in each algorithm are mentioned briefly. The paper concludes with a comprehensive comparison of the protocols based on several parameters.

Keywords: Power nodes, wireless sensor networks, routing protocols.

1. Introduction

Sensor networks are made up of a large number of sensor nodes which possess self-organizing capabilities. The core of a sensor node is a small, low-cost, low-power microprocessor. The microprocessor monitors one or more sensors and connects to the outside world with a radio link. Many popular radio transceivers allow a mote to transmit to a distance of a few hundred meters. The typical power consumption is about 10 milliamps when the mote is running, and about 10 microamps in sleep mode. Each sensor node is driven by one or two 1.5 V cells. The microprocessor, sensors, antenna and batteries are all packaged in small containers, typically a few millimeters thick [1–3].

Sensor networks may consist of many different types of sensors such as seismic, thermal, electrical, visual, acoustic, radar and so on. Sensor networks are finding a wide variety of applications in a number of domains. Some common applications of sensor networks are:

- Military applications such as battlefield surveillance, nuclear, biological and chemical (NBC) attack detection, and reconnaissance over enemy territory.

*Author for correspondence.

- Environmental applications such as wild animal tracking, air and water pollution level monitoring, forest fire detection and precision agriculture.
- Health applications such as heart rate monitoring, telemedicine and drug administration.
- Commercial applications such as highway traffic analysis, building security, structural fault detection, and power consumption measurement.

Sensor networks resemble embedded wireless networks in a variety of ways. Both types of networks consist of a collection of sensor nodes with limited computation and communication capabilities. Both have limited memory and communicate intermittently using radio links. Moreover, both have *data collection nodes* that sense and process data, and *control nodes* that monitor the network and transmit simple commands to the data collection nodes.

In spite of these similarities, sensor networks differ from traditional embedded wireless networks in many ways [4], some of them being:

- The scale of sensor networks is often orders of magnitude larger than that of traditional wireless networks. There may be tens of thousands of nodes in a sensor network, as compared to a few tens of nodes in a normal wireless network.
- Sensor networks are often densely and redundantly deployed, i.e. the number of nodes deployed per unit area is much greater than traditional wireless networks.
- Sensor networks are *dynamic* in the sense that nodes can get added to and deleted from the network without manual intervention, resulting in the expansion and contraction of the network after deployment.
- Sensor networks can be deployed in hostile territory, where they can be subject to communication surveillance and node capture and compromise by adversaries.
- Sensor nodes mainly use broadcast communication paradigms whereas traditional wireless networks mostly use point-to-point communication. The motivation for this paradigm shift is that in sensor networks, the focus is on the retrieval of data by attributes, and hence the individual nodes do not matter and are redundantly deployed.

Sensor network nodes may be deployed over a wide geographical region, say over a field of a few square kilometers. Typically there are one or more sink nodes or base stations which serve as collection points and connect the wireless nodes to a wired infrastructural network, for example, the Internet. Since the radio range of sensor nodes is of the order of a few hundred meters, the farthest nodes may not be able to reach the sink node in a single hop transmission. Moreover, the nodes may be deployed over uneven terrain in a non-uniform manner (as would be the case for example when several sensor nodes are air-dropped over a mountainous region). These factors combined with the resource limitations of sensor nodes make the problem of routing highly nontrivial. The obvious solution to this problem is to resort to *multi-hop routing*, wherein sensor nodes communicate with the sink node via multiple hops through other intermediate nodes. Each sensor node serves as a router in addition to sensing its environment. Conventional *link state* routing algorithms consume a lot of expensive memory space for maintaining their tables and are hence unsuitable for the sensor network scenario.

The lifetime of a fully active sensor node is of the order of a few days. The most energy-intensive operations for a node are those of radio transmission and reception. It is found that the energy consumed is proportional to the number of packets sent or received [5]. To maximize the network lifetime, therefore, the amount of network traffic should be minimized. One way of accomplishing this is for certain network nodes to collect raw sensor readings from a number of sensor nodes and combine them into a single composite signal which is then forwarded towards the sink node. This process is called *data aggregation*. Data aggregation can greatly reduce the number of packets transmitted, which can result in large energy savings.

The routing protocols that have been proposed for sensor networks can be broadly classified as *flat* and *hierarchical* protocols. Hierarchical protocols organize the network nodes into several logical levels. This is typically implemented by a process called *cluster formation*. A cluster consists of a set of geographically proximal sensor nodes; one of the nodes serves as a *cluster head*. The cluster heads can be organized into further hierarchical levels. The key advantage of hierarchical routing protocols is that the clusterheads can perform efficient in-network data aggregation. Routing proceeds by forwarding packets up the hierarchy until the sink node is reached. Flat routing protocols, on the other hand, attempt to find good-quality routes from source nodes to sink nodes by some form of *flooding*. Since flooding is a very costly operation in resource starved networks, smart routing algorithms restrict the flooding to localized regions. Some algorithms use probabilistic techniques based on certain heuristics to establish routing paths.

Flat routing protocols can be further classified according to the centrality of their theme. *Flooding-based* protocols rely primarily on flooding for route discovery. Many protocols couple query routing with data routing, i.e. source nodes transmit their observed data readings directly in response to queries from sink nodes. Such protocols can be classified as *query-driven* protocols. On the other hand, *data-driven* protocols assume that there is a separate query propagation phase by which some sensor nodes realize that their data should be sent to a sink. This phase is generally also responsible for setting up routes. Source nodes transmit their readings along these routes either periodically or whenever they observe some interesting events during the subsequent data transfer phase. *Multipath* routing protocols attempt to construct several completely or partially disjoint paths from the source to the sink. This increases the resilience of the network to node failures. *Geographic* routing algorithms route queries towards geographically defined regions. They are particularly suitable for sensor networks since user queries for physical phenomena such as movement are typically directed towards specific geographic regions. *Probabilistic* algorithms take packet-forwarding decisions probabilistically based on several parameters such as node reputation and link reliability. The classification of the surveyed routing algorithms is presented in Table I.

In the following sections, several routing algorithms that have been proposed over the last few years are surveyed. The operation of each algorithm is explained in a simplified manner in Section 2. The merits and demerits of each algorithm are discussed. The paper concludes with a summary of the routing protocols in Section 3.

Table I
Classification of sensor network routing protocols. The first seven categories are specific instances of flat routing protocols

Routing protocol category	Example routing protocols
Flooding-based	TinyOS Beaconing, Pulse
Query-driven	Directed Discussion, Rumor Routing, Braided Path Routing, GEAR
Data-driven	SPIN, GRAB, INSENS, SAR, ARRIVE
Multipath	Braided Path Routing, GRAB, INSENS, SAR
Geographic	GEAR
Probabilistic	ARRIVE
Other flat	ASCENT, Deng *et al.* [20], TBF, Data Mules
Hierarchical/ Cluster-based	SWE/MWE, LEACH, SRPSN

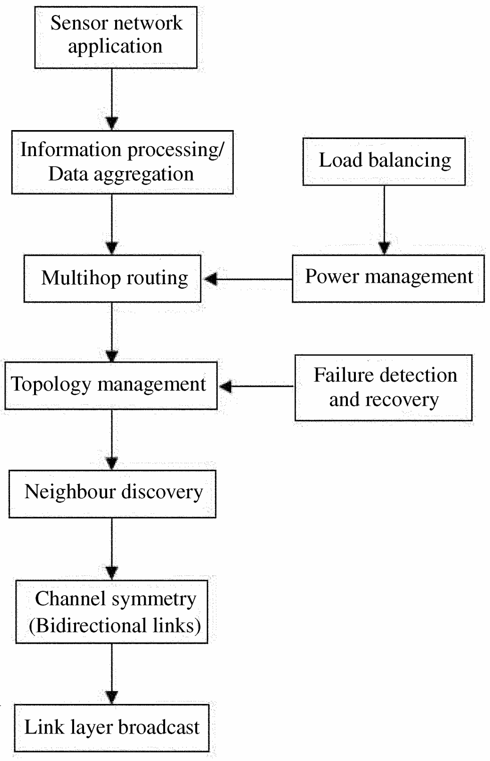
Sensor network application
Information processing/ Data aggregation
Load balancing
Multihop routing
Power management
Topology management
Failure detection and recovery
Neighbour discovery
Channel symmetry (Bidirectional links)
Link layer broadcast

FIG. 1. Depends-on hierarchy for flat sensor network routing protocols. The function at the tail of an arrow depends on the function at the head of the arrow.

2. Routing protocols

A number of routing protocols for sensor networks have been proposed in the literature over the last few years. Many of the protocols draw inspiration from similar protocols for wireless ad-hoc networks. Since the challenges for sensor networks are different from those of ad-hoc networks, several interesting variations are introduced. In addition, many novel routing mechanisms have been proposed specially for sensor networks. The following subsections survey many of the sensor network routing algorithms.

2.1. *Flat routing protocols*

Flat routing protocols are similar to the conventional multihop ad-hoc routing protocols. Each sensor node determines its parent node(s) to forward data packets. The nodes are not organized into hierarchical clusters as is done in the hierarchical protocols. The advantage of this approach is that all the nodes can reach the base station irrespective of their position.

Each of the flat routing protocols can be decomposed into several constituent blocks as depicted in Fig. 1. The arrows in the figure depict the *depends-on* relation between functions. Multihop routing is an essential prerequisite for data aggregation; this is because there is no scope for aggregation if each node transmits directly to the base station. Similarly, reliable neighbour discovery depends on channel symmetry. If the radio links are not bidirectional (for example, as a consequence of the hidden terminal problem), then reliable communication is not possible. Link layer broadcast is a fundamental requirement for sensor network routing, since radio channels are inherently broadcast in nature.

Multihop routing makes it possible to achieve load balancing by restricting the power level at which sensor nodes communicate. Since the sensor nodes have severely restricted power resources, this can greatly increase the lifetime of the network. Finally failure detection and recovery is possible if each node is aware of its surrounding network topology.

Most of the flat routing protocols that have been proposed for sensor networks incorporate distance vector routing algorithms. In distance vector routing [6], nodes maintain estimates of their distances from the destination nodes. Each node transmits its distance estimates to its neighbours. Each node updates its distance vector so as to minimize the distance to each destination by examining the cost to that destination reported by each of its neighbours and then adding its distance to that neighbour. The problem with the straightforward distance vector algorithm is that it takes a long time to converge after a topological change. Several techniques are used to detect the counting to infinity problem [6] and hasten convergence in practice. For instance, some protocols use a time-to-live (TTL) field in their packets. When the TTL drops to zero, the packet is discarded. Other protocols use randomization techniques to avoid routing loops. Some other ways in which convergence is achieved are backpropagating learned costs to destinations, making route changes only at periodic intervals, eavesdropping on the broadcast medium, running centralized shortest path algorithms and so on. Table II summarizes these convergence techniques.

The following subsections describe the routing protocols that can be classified as flat routing protocols.

2.1.1. *TinyOS beaconing*

The TinyOS embedded sensor network platform [7] employs a very simple ad-hoc routing protocol. The base station periodically broadcasts a route update *beacon* message to the network. The beacon message is received by a few nodes that are in the vicinity of the base station. These nodes mark the base station as their parent and rebroadcast the beacon to their neighbours. The algorithm proceeds recursively with nodes progressively propagating the beacon to their neighbours; each node marks the first node that it hears from as its parent. The beacon is thus flooded throughout the network, setting up a *breadth-first spanning tree* rooted at the base station. This process is repeated at periodic intervals known as *epochs*.

Each network node periodically reads its sensor data and transmits the data packet to its parent in the spanning tree. The parent node in turn forwards the packet to its parent and so on. This process is repeated until the data finally reaches the base station.

The attractive feature of TinyOS beaconing is its simplicity–nodes do not have to maintain large routing tables or other complicated data structures. Each node needs to remember only its parent node in the path to the base station. By combining the beaconing with a MAC layer scheduling scheme such as TDMA, the nodes can conserve power by keeping their radio off most of the time. In spite of its attractive features, the beaconing protocol suffers from one main disadvantage: it is not resilient to node failures. If a parent node

Table II
Techniques used to achieve convergence and loop elimination in distance vector-based flat sensor network routing protocols

Protocol	Techniques used to achieve convergence and loop elimination
Directed diffusion	Each node maintains an interest cache and a data cache. If a node receives a data message from its neighbour for which it does not have an entry in its interest cache, the packet is silently dropped. Similarly, if the data corresponding to an interest is already present in the data cache, the packet is silently dropped. This prevents the formation of routing loops.
Rumor routing	Query packets maintain a list of recently traversed nodes, using which they avoid revisiting any node. Each node maintains a list of recently encountered queries; if a node receives the same query again, it forwards it in a random direction. Maintaining a TTL in each query also helps in eliminating routing loops.
Braided multipath routing	Multiple partially disjoint paths are constructed and maintained which increases the probability of packets reaching their destination.
GRAB	Only nodes that have lesser cost than the sender are allowed to forward packets. Routing loops are avoided since packets flow towards nodes whose costs decrease monotonically. The algorithm also employs event-driven refreshing of the cost field that can ensure that the information about link failures spreads throughout the network and thereby avoids the counting to infinity problem.
INSENS	The base station periodically collects global information from the network nodes and runs a centralized Dijkstra’s algorithm to compute optimal routes.
SAR	The counting to infinity is hastened by detecting rapid increase in the path metric using a threshold method.
GEAR	Each time a packet is successfully delivered to the destination, the correct value of the learned cost is propagated one hop backwards. Thus the learned cost will converge at all the nodes along the path in a finite amount of time.
Woo *et al.* [18]	Route changes are made only at periodic intervals or when a cycle is detected. This can ensure that the information about link failures spreads throughout the network. Moreover nodes eavesdrop on the broadcast medium and quickly learn if a node does not have a route to the destination. These mechanisms avoid the counting to infinity problem.
Deng *et al.* [20]	The hop count (to the base station) of the nodes in a loop keeps increasing continually. After some limit, neighbouring nodes that have real paths to the base station attract some of the loop nodes and thus break the loop.

fails, then its entire subtree is cut off from the base station during the current epoch. Moreover, the protocol results in uneven power consumption across network nodes. The nodes nearer to the base station consume a lot of power in forwarding packets from all the nodes in their subtree, whereas the leaf nodes in the spanning tree do not have to perform any forwarding at all and consume the least power.

2.1.2. *Pulse*

The Pulse protocol [8] addresses the three topics of routing, energy consumption and time synchronization in sensor networks. It uses a periodic pulse signal generated and flooded by a pulse source to provide routing paths and synchronization to the network. As the pulse propagates through the network nodes, a spanning tree rooted at the pulse source is constructed. Node traffic follows the paths along this spanning tree. A node that wants to communicate packets sends a *reservation packet* to the pulse source. The reservation packet contains the address of the node sending the packet and is used to set up reverse routes for data packets. Thus, active nodes need to keep sending reservation packets in response to the periodic pulse signals to keep the routes fresh. Idle nodes that do not have data to communicate and that are not needed for forwarding packets can switch off their radios till the next pulse signal arrives and thereby save energy.

To further reduce energy consumption, the Pulse protocol is modified to incorporate *intermediate wake-up periods*. The motivation behind this modification is that the routes in the network are established by the flooding of the pulse signal, which is an expensive process. Instead, nodes are permitted to send reservation packets during intermediate wake-up periods which can occur several times between two pulse floods. This enables faster path activations with lesser energy expenditure.

The Pulse protocol is similar to the beaconing protocol if the pulse source is considered to be the base station. Thus it has similar merits and demerits as the beaconing protocol. One area of improvement in the Pulse protocol is to provide a path deactivation feature. This feature would allow nodes to deactivate paths and conserve energy even if the intervals between wake-up periods are arbitrarily long. This would of course trade off the fast path activation for power efficiency.

2.1.3. *Directed diffusion*

A data-centric communication protocol for sensor networks has been proposed in [9]. All sensor data are characterized by attribute-value pairs. A node that requires data sends out *interests* for named data; interests are diffused through the network towards the nodes that are capable of responding. Data are in turn drawn towards the requesting node via *gradients* established along the reverse path of interest propagation. This style of data-centric communication is fundamentally different from the node-centric end-to-end communication mechanism of traditional IP networks.

An interest for data may contain several fields such as type, interval, duration, time stamp and the coordinates of the target region. The duration refers to the time period for which data is desired, and the interval refers to the data rate. The sink broadcasts interests to its neighbours; due to the unreliable nature of broadcast networks, interests are refreshed periodically with updated timestamp values. The initial interest specifies a large interval value; when the path to the event source is established, a higher data rate is requested. Each node maintains an *interest cache* that contains several fields. One of the fields is called a *gradient* that specifies the node's downstream neighbour. The gradients in each node are used to set up the reverse path for information flow from the source to the sink. A gradient also specifies the data rate requested by the neighbouring node.

Whenever an interest is received, the node looks up its interest cache. If there is no matching entry in the cache, a new interest entry is created. If a matching entry exists already, its timestamp is updated. The node further broadcasts the interest to its neighbours, and thus the interest is flooded throughout the network, ultimately reaching the source.

When the source node detects an event, it searches its interest cache for matching event entries. If a matching interest entry is found, the node starts relaying its readings at the highest requested data rate among all its outgoing gradients. Intermediate nodes that receive a data message from their neighbours also check their interest caches for matching entries. If no matching entry is found, the data packet is silently discarded. Otherwise, the node searches its *data cache* associated with the matching interest entry. If there is no recently seen data item corresponding to the interest, a new entry is created and the data is forwarded to the neighbouring nodes; if the data is already present in the cache, the data packet is silently dropped. This mechanism helps in preventing the formation of loops in data dissemination.

The sink may finally receive low-rate event data from several paths. It *reinforces* one of its neighbours to draw high-rate events. Reinforcement is done by sending out an interest with a higher data rate (smaller interval). The same procedure is adopted by all the upstream nodes to reinforce one or more paths that deliver high-quality event data. This finally results in an empirically low-delay path between the source and the sink. In case multiple paths are created and some paths are found to perform consistently better, an option is available to *negatively reinforce* the other paths. The reinforcement rules can also be applied by intermediate nodes along previously reinforced paths to enable local repair of failed or degraded paths. Figure 2 illustrates the working of the directed diffusion algorithm.

Directed diffusion has been a pioneering work in the area of data-centric routing in sensor networks. It has introduced several new features such as path reinforcement, caching and in-network data aggregation. There is adequate scope for further research in each of these areas. Several routing protocols such as rumor routing [10] and highly resilient multipath routing [11] have drawn inspiration from directed diffusion.

2.1.4. *Rumor routing algorithm*

Braginsky and Estrin [10] propose an algorithm to route user queries to nodes that have observed certain events. Events are assumed to be localized phenomena, occurring in a fixed region of space. Queries can be requests for information or commands to initiate collection of more data. If the number of observed interesting events is high and the number of queries for the events is low, it is better to flood the queries through the network. On the other hand, if the number of user queries is very high compared to the number of interesting events, it makes sense to flood event information. The rumor routing algorithm tries to fit in between query flooding and event flooding.

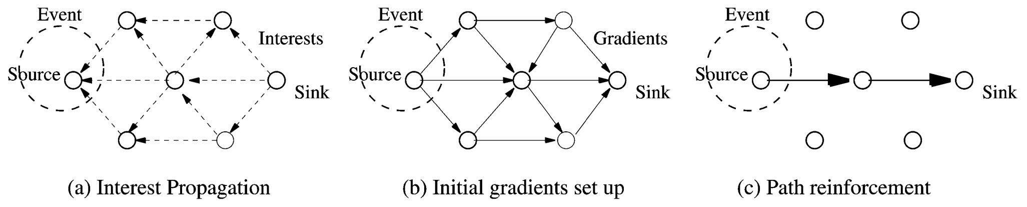

FIG. 2. Directed diffusion.

Rumor routing aims to create paths leading to events; whenever a query is issued for an event, it is sent on a random walk through the network until it intersects one of the event paths. If the random query path fails to intersect any event path, the application resubmits the query, or in the worst case, floods the query throughout the network.

Each node maintains a *neighbour table* and an *event table*. The event table contains a list of events that the node has observed. The neighbour table can be maintained by actively initiating hello messages or passively eavesdropping on network broadcasts.

The algorithm employs a set of long-lived packets called *agents* that traverse the network, record interesting events that they observe and disseminate this event information to network nodes. Agents are generated by nodes randomly with a tunable probability based on whether the nodes have observed an event in the recent past. Agents also contain event tables similar to nodes, which include the number of hops to each event. When an agent crosses a node that has information about some event that the agent has not yet seen, it updates its event table to include the event. Agents travel for a specified number of hops and then die. Nodes can update their routing tables when they encounter agents that have cheaper paths to certain events.

Any network node may generate a query; if a node has a route to the desired event, it is forwarded along the route. If there is no route, the query is forwarded in a random direction. The query packet maintains a list of recently traversed nodes, and avoids revisiting any of them. Similarly each node maintains a list of recently encountered queries; when it receives the same query again, it forwards the query in a random direction instead of along the route to the event. This mechanism along with maintaining a TTL in each query can eliminate routing loops.

There are several problems that need further exploration in rumor routing. The path followed by agents is determined randomly in the algorithm. Instead it may be possible to find interesting events faster by exploiting localization information. Another alternative could be to resort to limited flooding hoping that paths may be found quickly. The probabilistic parameters can be tuned for optimal performance either by nodes using local information or at the system level by observing the event patterns over a period of time.

2.1.5. *SPIN*

Classic flooding-based routing protocols suffer from three basic deficiencies:

1. *Implosion:* Flooding delivers packets to nodes regardless of whether they have already received a copy of the packets from another location. This is the case in diamond-shaped topologies as illustrated in Fig. 3.
2. *Overlap:* The region sensed by two neighbouring nodes may overlap in area. This is particularly true in densely deployed sensor networks where nodes are located close to one another. Thus an upstream node may receive packets representing identical information from two or more different sensors.
3. *Resource blindness:* Flooding-based protocols do not distinguish between energy-rich and energy-depleted nodes.

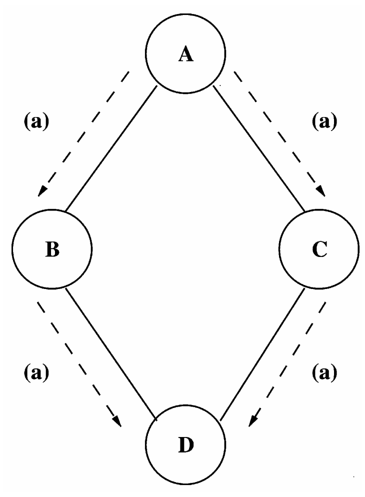

FIG. 3. Implosion of packets in classical flooding. Node A floods the packet marked (a) to its neighbours B and C. Thus node D receives two copies of the same packet, one from each of its neighbours B and C.

To rectify these deficiencies of classic flooding, a set of protocols called SPIN (sensor protocols for in formation via negotiation) [12] have been proposed. Nodes name their data using high-level descriptors called *metadata*. The problems of implosion and overlap are eliminated by avoiding transmitting redundant copies of the same data using the metadata descriptors. Since metadata is so crucial, it must be the case that the metadata representing a piece of data should be considerably shorter than the actual data itself. Moreover, each node has a component called the *resource manager* that tracks energy consumption and is consulted whenever applications transmit or process data. This eliminates the problem of resource blindness.

SPIN comprises a family of protocols: SPIN-PP and SPIN-EC for point-to-point communication networks and SPIN-BC and SPIN-RL for broadcast networks. SPIN-PP assumes that each node can communicate with another node without interfering with any other nodes. When a node has data to communicate, it begins by sending out a new data advertisement (ADV) message to its neighbours. This message includes the metadata corresponding to its data. Each neighbour first checks to see if it has already received or requested for the advertised data. If not, it responds with a data request (REQ) message to the sender. Finally, the sender transmits the requested data in a DATA message. SPIN-EC is the energy conserving version of the basic SPIN-PP protocol. It uses the same ADV, REQ and DATA messages as SPIN-PP. However, if a node observes that its energy reserves are approaching a low threshold, it stops participating in the protocol unless it determines that it can complete all the three protocol steps without running out of energy.

The broadcast protocols SPIN-BC and SPIN-RL exploit the one-to-many transmission phenomenon of broadcast networks, i.e. a broadcast message sent by one node is received by all its neighbours. Thus the multiple unicast messages addressed separately to individual neighbours in SPIN-PP are combined into a single broadcast ADV message that reaches all neighbours. If all neighbours respond immediately to the data advertisement broadcast, there will be a lot of MAC layer collisions. To circumvent this, nodes wait for random intervals before transmitting their REQ messages, with the original advertiser specified in the header of the REQ message. As soon as the first REQ message is heard, the other neighbours cancel their (now redundant) REQ messages. Moreover, the advertiser responds to the REQ with a single broadcast DATA message that reaches all the neighbours simultaneously. SPIN-RL is a reliable version of SPIN-BC in which nodes keep track of which node sent the ADV message. If they do not get a DATA message within a timeout period, they resend their requests. Also the rate at which DATA messages are sent is limited—a node waits for a specified amount of time before responding to a request for the same piece of data more than once.

The SPIN family of protocols effectively solves the problems inherent to classic flooding with lesser energy consumption using metadata descriptors. Moreover, the protocols maintain only local information about their neighbouring nodes which is an attractive feature for mobile networks. Further work remains to be done in the case of networks that use both unicast and broadcast paradigms.

2.1.6. *Highly resilient energy-efficient multipath routing*

Ganesan *et al.* [11] present a multipath routing technique to improve the resilience of a sensor network to node failure. Constructing k disjoint paths from the source to the sink ensures that the network does not get disconnected even if k nodes fail. However, finding completely disjoint paths tends to be very energy-inefficient. To overcome this problem, an algorithm to construct partially disjoint paths (called braided paths) with some common nodes between paths is presented.

The algorithm aims to extend the concept of directed diffusion [8] to eliminate the energy-intensive flooding used to discover alternate paths. The basic idea of the algorithm is to set up multiple paths along which data is disseminated at low data rates at the same time when the primary path is established. If there is any node or link failure on the primary path, nodes can quickly reinforce one of the alternate paths without resorting to expensive flooding. Nodes have to fall back on flooding only if all the multiple paths fail simultaneously.

A mechanism to discover strictly disjoint multipaths is presented first. Following the directed diffusion algorithm, the sink reinforces the link with its most preferred neighbour. At the same time, it sends an alternate reinforcement message to its next most preferred neighbour, say A. A propagates this reinforcement to its most preferred neighbour, say B, in the direction of the source. If B already happens to lie on the primary path between the source and the sink, it sends a *negative reinforcement* message back to A; A then tries its next most preferred neighbour and so on. Otherwise B continues to propagate the alternate reinforcement to its neighbours. This procedure can be extended to discover k disjoint multipaths between the source and the sink.

The problem of finding braided multipaths can be defined as finding the best path from the source to the sink that does not contain one of the nodes on the primary path. This results in finding an alternate path that is expected to be physically close to the primary path, and hence dissipates energy proportional to the primary path. Braided paths are constructed using a procedure similar to that of the disjoint multipaths. Each node on the primary path sends reinforcement messages to its first and second most preferred neighbours, thus trying to route around its immediate neighbour on the primary path. A node not on the primary path that receives a reinforcement message from a primary node propagates the message to its most preferred neighbour. If this neighbour happens to lie on the primary path, then the reinforcement is not propagated any further (since a braided path has already been found).

Constructing strictly disjoint multipaths ensures that any number of failures on the primary path does not affect any of the alternate paths. In contrast, in the case of the braided multipaths, failure of a certain set of nodes on the primary path can disrupt all the multipaths. However, the advantage of braided multipaths stems from the fact that the total number of *distinct* alternate paths through a braid is much higher than the number of nodes along the primary path, thus greatly increasing the resilience of the braid. It will be interesting to study the extension of the braided multipath algorithm to multiple sources and sinks with respect to complexity, resilience to isolated and patterned failures, and maintenance overhead.

2.1.7. *GRAB*

GRAdient Broadcast (GRAB), a robust data delivery protocol for large-scale sensor networks has been proposed by Ye *et al.* [13]. This algorithm constructs and maintains a *cost field* that specifies the cost incurred by a packet at each node to reach the sink. Sensor data is forwarded along an *interleaved mesh* from the source to the sink. Nodes do not maintain state about their next hop parents, instead, all data packets are routed by broadcasting. Only lower cost nodes are permitted to forward the packets. The resulting multiple interleaved paths greatly improve network reliability. The width of the mesh can be controlled by a *credit* carried in each data message to achieve any desired trade-off between robustness and energy consumption.

The protocol begins with the sink broadcasting advertisement (ADV) packets with a cost of zero to initiate the construction of the cost field. Each sensor node starts out with a cost of infinity. When a new ADV packet is received from a neighbour, a node updates its cost to the minimum of its original cost and the sum of the advertised cost and the cost of transmitting a packet to the neighbour. The ADV packet is rebroadcast if it reduces the node's cost. Thus the algorithm greedily builds gradients directed towards the sink, nodes nearer the sink have a low cost, whereas those farther away have a higher cost. Packets flow along the gradients from higher cost nodes towards lower ones and ultimately reach the sink. The cost field can be reconstructed as and when the sink observes significant variations in the packet reception characteristics.

The physical or environmental phenomenon to be sensed is observed by many nodes in the vicinity. The nodes exchange their received signal strength among themselves. The node that has the largest signal strength is elected as the *Center of Stimulus* (CoS) and is responsible for transmitting the sensor reading to the sink. At each hop, only the nodes that have lesser cost than the sending node are allowed to forward the packet further. This mechanism implicitly avoids the creation of routing loops, since packets flow towards nodes whose costs decrease monotonically. The routing mesh is illustrated in Fig. 4.

In the above scheme, packets tend to take diverging paths many of which lead away from the direction of the sink. Considerable amount of energy is spent before all the diverging

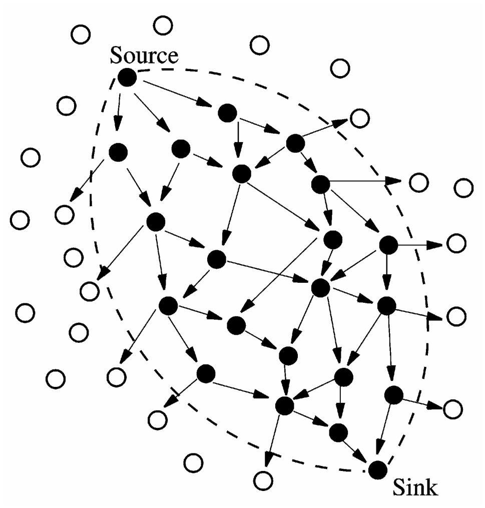

FIG. 4. Routing of packets along the GRAB forwarding mesh. Some of the unshaded nodes receive packets but do not forward them since they have a higher cost to the sink.

packets are discarded. To control the width of the forwarding mesh, the source node includes a parameter called the *credit* in each data packet. The credit represents the amount of extra budget that can be used to forward packets along multiple interleaved paths. The difference between the costs of the minimum cost path and any other path should not exceed the credit specified by the source. Each data packet includes the credit, the cost of the source, the cost of sending the node and the cumulative amount of power consumed for forwarding the packet from the source to the current node. Using these fields, the receiving node can determine whether it has sufficient credit; if so, it can increase its transmission power to forward the packet to many nearer neighbours. Otherwise, the node does not have adequate credit and broadcasts the packet at a power level that is just sufficient to reach its next hop neighbour on the minimum cost path. Nodes wait for random amounts of time before transmitting to minimize the chances of collisions.

The credit assignment in the GRAB packets can be made adaptive to the local network conditions. This can be driven by the sink node which can indicate the quality of recently received packets. Nodes facing higher losses and failures in their neighbourhood can use greater shares of their credits. Another problem that needs to be addressed relates to multiple sources and sinks. This may require building multiple cost fields which may turn out to be prohibitively expensive. Similar issues may be encountered in the case of a mobile sink wherein the cost field needs to be reconstructed periodically as and when the sink moves.

2.1.8. *INSENS*

INSENS (INtrusion-tolerant routing protocol for SEnsor NetworkS) [14] is a routing protocol that ensures that a single compromised node can only affect a limited portion of the network without disrupting the functioning of the rest of the network. This is accomplished by maintaining multiple disjoint paths from sensor nodes to the base station. It forbids individual sensor nodes from broadcasting to the entire network. This succeeds in preventing denial of service type of attacks. Control information such as routing protocol messages is necessarily authenticated so that the base station always receives a correct, albeit possibly incomplete, picture of the network topology. Since sensor nodes are severely limited in terms of computation power, memory and energy levels, the expensive job of computing and distributing routing tables is delegated to the resource-rich base station. Moreover, symmetric key cryptography is used for confidentiality and authentication operations rather than resource-intensive public key cryptography.

The routing protocol involves two phases: the *route discovery phase* and the *data forwarding phase*. The route discovery phase is divided into three rounds: route request, route feedback and computation and propagation of multipath routing tables.

The route request round is initiated by the base station which broadcasts a *request* message to its immediate neighbours. Each of the immediate neighbours of the base station in turn broadcasts a message to their neighbours. In general, the message broadcasts by any node includes a path from the base station to that node. The process of recursive broadcasts continues until all the nodes receive the route request.

The messages from the base station are authenticated by the sensor nodes using a one-way cryptographic function F as follows. Each node is preprogrammed with a secret key shared with the base station along with an initial sequence number K_0. The base station generates the sequence of numbers $K_0, K_1, \ldots, K_n$, such that $K_i = F(K_i + 1)$, $0 \leq i < n$. During each route discovery phase, the base station includes the next number from this sequence in the request message that it sends. Sensor nodes can authenticate each message by verifying the relation between the current and previous sequence number. Since the sequence numbers get revealed in the reverse order from which they were generated, it is impossible for a malicious node to guess the next sequence number and thereby spoof the base station.

In the second round, each node sends a list of its neighbouring nodes in the form of a feedback message to the base station. This feedback message follows the reverse route as the one followed during the route request phase. A MAC of the message contents generated using the secret key shared with the base station is appended to the message and used for node authentication. To prevent DoS attacks, the outgoing rate of messages at each node is limited below a fixed maximum value irrespective of the incoming message rate. In the third round, the base station waits for a timeout period before it receives all the feedback messages. Then it constructs multiple redundant paths between sensor nodes and the base station as follows. For a given node A, the shortest path to the base station is determined using Dijkstra's algorithm. To construct a node-disjoint secondary path, the nodes along the primary path along with their one- and two-hop neighbours are removed and a shortest path from A to the base station is found. If such a path does not exist, the procedure is successively repeated by removing the primary path nodes and their one-hop neighbours, and finally only the primary path nodes. If a shortest path from A to the base station is not found after any of these steps, only a single shortest path to the base station is maintained. The computed routing tables are disseminated among all the sensor nodes hierarchically by the base station starting from its immediate neighbours.

Once the route discovery phase completes, sensor data can be forwarded by nodes to the base station at periodic intervals. Each sensor node maintains a *forwarding table* that has several entries of the form <*destination*, *source*, *immediate sender*>, where *destination* and *source* are the destination and source nodes of a packet and *immediate sender* is the node that just forwarded the packet. When a node receives a packet from its neighbour that matches an entry in its forwarding table, it sets the immediate sender field of the packet to itself and forwards (broadcasts) the packet.

INSENS is a simple routing protocol in that the routing computations are performed by the central base station rather than the resource-constrained sensor nodes. While this may be energy efficient for the sensor nodes, the dependence on the base station makes it unsuitable for highly dynamic sensor networks. For instance, topology changes keep happening frequently in mobile or lossy networks. To maintain network connectivity, INSENS would have to run the route discovery phase frequently which may not be feasible in practice.

2.1.9. *SAR*

A table-driven multipath routing algorithm, *Sequential Assignment Routing* (SAR) [15] is proposed to improve the resilience of the network to node failures. Finding strictly disjoint multiple paths complicates localized recovery schemes. Generating a k-disjoint structure requires approximately k times the overhead complexity of a shortest path algorithm. Hence the disjointness property is not strictly required for nodes away from a 1-hop neighbourhood of the sink.

To create multiple paths to the sink, multiple routing trees rooted at different 1-hop neighbours of the sink are constructed. The trees proceed to grow away from the sink by successive branching, avoiding nodes with low QoS and energy reserves. Most of the sensor nodes in the network end up belonging to more than one routing tree. Packets are also assigned priorities, and higher-priority packets are provided higher QoS. The objective of the SAR algorithm is to optimize the average weighted QoS metric in the network.

Failure recovery is enforced by a local handshaking procedure that ensures routing table consistency between the upstream and downstream neighbours of the failed nodes. This is accomplished by triggering a local route recomputation procedure whenever a node fails. The counting to infinity problem is avoided by hastening the convergence to infinity whenever the path metric reaches an upper threshold.

2.1.10. *GEAR*

Yu *et al.* [16] present GEAR (Geographical and Energy Aware Routing), a geographic routing algorithm, i.e. an algorithm that routes queries towards a certain geographically defined region. User queries for physical phenomena such as temperature or movement in the real world are typically directed towards specific geographic areas. This makes geographic algorithms very attractive for sensor network routing. The GEAR algorithm uses heuristics based on energy levels to route packets, instead of flooding. Once the query reaches the boundary of the desired region, it is disseminated within the region using a recursive forwarding technique. It is assumed that each node can determine its own position using some localization system such as GPS. Further, nodes exchange their position and remaining energy level information with their neighbours using a simple hello protocol.

Each query packet specifies the target region in some way, for example, by specifying the coordinates of a rectangular region. Each node maintains a *learned cost* to target regions. The learned costs of nodes are exchanged with their neighbours occasionally. In case the learned cost of a neighbour is not available, the node computes an *estimated cost* for that neighbour. The estimated cost of a neighbour is calculated as an exponential mean of the distance of the neighbour from the target region (determined using the position information) and its remaining energy level. The exponential mean value can be controlled by a tunable parameter which can alter between pure geographic mode and pure energy conserving mode. To route queries to the target region in an energy-efficient manner, a node greedily forwards the query to the neighbour that has the minimum learned cost. Moreover, the node updates its learned cost for the target region to be the sum of the learned cost of the next-hop neighbour and the cost of transmitting a packet to the next-hop neighbour.

The proposed algorithm works even if a node is in a *hole*, i.e. all its neighbours are further away from the target region. This is because the query packet is routed around the hole by picking the lowest cost neighbour as the next-hop at each step. Each time a packet is successfully delivered to the destination, the correct value of the learned cost is propagated one hop backwards. Thus the learned cost will converge at all the nodes along the path in a finite amount of time.

Routing continues as described in the previous paragraphs until the packet arrives at the target region. A naive way of distributing the packet in the target region is to flood it within the entire region. However, flooding is an expensive operation especially in densely deployed sensor networks. Instead a *recursive geographic forwarding* algorithm is followed wherein the first node within the target region that receives an incoming packet divides the region into four rectangular sub-regions and transmits four copies of the packet to each of the four subregions. This process of subdividing and forwarding packets continues until either there is only one node in a subregion or the packet reaches an empty subregion at which point it is dropped. A node determines that it is the only one in a subregion if its transmission range encompasses the subregion but none of its neighbours lies within the boundaries of the subregion. It can determine if a subregion is empty in a similar manner. The process of recursive geographic forwarding is illustrated in Fig. 5.

The recursive geographic forwarding algorithm suffers from two pathological conditions. Firstly, it unicasts packets four times to the subregions which results in large energy consumption. Secondly, it may not terminate if the farthest point of an empty subregion is outside the radio range of a node. This typically happens when the network density is low. Thus the node does not know that the target subregion is empty and the packet keeps trying to find routes to the empty subregion (until it times out). Both of the above conditions can be avoided by resorting to restricted flooding within the target region when the network density is below a threshold value (as indicated by the node degree of the boundary node in the target region).

2.1.11. *ARRIVE*

Karlof *et al.* [17] propose ARRIVE (Algorithm for Robust Routing In Volatile Environments), a probabilistic algorithm that avoids packet loss by sending multiple packets corre-

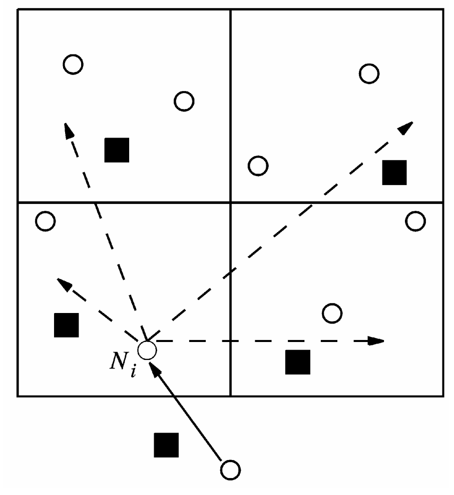

FIG. 5. Recursive geographic forwarding. Circles depict nodes and rectangles depict packets. N_i is the first node in the target region that receives the incoming packet. It divides the region into four rectangular subregions and forwards four copies of the packet to each of the subregions. The process continues recursively until all nodes receive the packet.

sponding to a single sensor reading. It ensures that different links are chosen for each of the packets by selecting outgoing links probabilistically based on link reliability and node reputation. To mitigate the effects of malicious nodes, nodes listen in promiscuous mode to the transmission of their neighbours; if a node E detects that a packet sent from node A to node B is not being forwarded by B, then E itself forwards the packet to the next-hop neighbour of B. This is called *passive participation*. The existence of unidirectional links in the network as well as the hidden terminal problem may result in increased traffic since the listening node may not be able to perceive actually forwarded packets from B.

Each node in the network is assigned a level during an initial breadth first search beaconing procedure. Nodes closer to the root (which could be the base station) have lower-numbered levels, whereas those farther have higher-numbered levels. The algorithm decides whether to forward an incoming packet to a parent or a peer neighbour based on a *forwarding probability*. The probability with which packets are forwarded to parents (as opposed to peer neighbours) increases as the nodes get closer to the root of the tree.

As a security measure, a kind of peer-ranking system is followed, wherein each node maintains a reputation history for each of its neighbours. The *reputation* of a node P with respect to a node N is defined to be the ratio of the number of packets sent by N to P to the number of packets actually forwarded by P. The reputation is periodically updated using an exponential average function, with recent information getting more weightage. When a node receives a packet, it first checks if the packet is addressed to itself. If so, it filters out all the nodes whose reputation falls below a certain threshold value from being considered for the next hop.

To promote diversity of the paths followed by packets, each node maintains a *convergence history* of all events. The convergence history maintained by a node for an event is defined to be the set of all the nodes to which the node has forwarded packets representing the event. When choosing the next hop for a packet addressed to itself, a node excludes all the members of the convergence history corresponding to the event represented by the packet. This ensures that packets corresponding to the same event follow different paths and thereby increase the chance of arriving at their final destination.

The ARRIVE algorithm does not require sensor nodes to maintain large routing tables nor does it assume specialized localization mechanisms. Its limitation however is that there may not be multiple alternate routes between the source and the sink in sparse sensor networks, and hence path diversity may not be feasible. Also the paths followed by packets are not optimal due to the probabilistic nature of the algorithm. The algorithm could also benefit from MAC layer scheduling schemes such as TDMA.

2.1.12. *Taming the underlying challenges of reliable multihop routing*

Woo *et al.* [18] propose that the link characteristics should be taken into consideration by the routing protocol. Radio links are highly lossy and unreliable; the loss rate may change dynamically due to environmental factors and high contention for the broadcast medium. The routing protocol should take such statistics into account for reliable routing.

The authors test a number of link estimation techniques and conclude that the WMEWMA (Window Mean with Exponentially Weighted Moving Average) estimator performs best with constant memory overhead and small settling time. This estimator measures the ratio of packets received to the number of expected packets over a time interval. These periodic ratios are smoothened using an exponential mean with a tunable parameter.

A neighbourhood management scheme that attempts to retain the maximum number of good neighbours within each node using constant memory is also proposed. Neighbours are inserted in the neighbourhood table of a node until it becomes full, at which point existing neighbour entries have to be evicted from the table. Several candidate eviction policies such as round robin, FIFO (First In First Out), LRH (Least Recently Heard) and CLOCK approximation to LRH are considered. Upon hearing from a neighbour that already exists in the neighbourhood table, its entry is reinforced. A frequency count is maintained for each neighbour; if a neighbour has to be reinforced, its count is incremented by one. The eviction policy selects one of the entries whose count has dropped to zero.

A distributed distance vector routing algorithm is implemented using the neighbourhood tables. A parent selection process runs periodically and identifies the next hop neighbour using a cost metric that may be the number of hops, the expected number of transmissions or some other parameter that represents the energy required to reach the sink. The link estimation process allows the selection of new parents over time. To minimize the cascading effect of changing parents, route changes are made only at periodic intervals. The counting to infinity problem associated with distance vector routing is solved by having nodes eavesdrop on the broadcast medium and quickly learn if a node does not have a route to the destination.

2.1.13. *ASCENT*

The ASCENT (Adaptive Self-Configuring sEnsor Networks Topologies) system [19] builds on the observation that only a subset of the nodes is actually required to establish connectivity in a dense sensor network. Each node determines its connectivity and decides whether or not to participate in the routing mechanism. ASCENT is not a routing protocol per se; it operates in between the MAC and network layers and only determines which nodes join the routing infrastructure. It does not utilize or modify state maintained by the routing protocol.

Nodes in ASCENT can be in one of four states: *active*, *passive*, *test* or *sleep*. Active and test nodes are involved in the forwarding of data and routing control messages. Sleeping nodes keep their radios turned off to save power, whereas passive nodes only listen to the network traffic in promiscuous mode. Initially nodes start in the test state which is used to determine if the addition of a new node is likely to improve the connectivity of the network. When a node enters the test state, it initializes a timer T_t and transmits *neighbour announcement* messages to other nodes. The node upgrades itself to the active state as soon as the timer expires. Before the timer goes off, however, if the number of active neighbours is above a certain threshold or the data loss rate is higher than before, the node falls back to the passive state. A passive timer T_p is started when the node enters the passive state. As soon as this timer expires, the node turns its radio off and enters the sleep state. Before this timer goes off, however, if the number of active neighbours is less than a threshold, or the node hears a help message (described below) from an active neighbour, it transitions back to the test state. Nodes in the sleep state wake up and move to the passive state after a time-out interval, T_s. Figure 6 shows the various state transitions.

The source transmits packets towards the sink via the active nodes. Since there are only a few active nodes to start with, many losses are encountered. This prompts the active nodes to broadcast *help* messages, asking for more active nodes to join the network. Some of the nodes in the passive state react to the help messages and become active nodes. This process continues until a sufficient number of active nodes is available for reliable data transmission.

The ASCENT system follows a reactive algorithm that responds to changes in the network characteristics. This results in some latency before the network stabilizes. The performance of the system needs to be carefully studied in highly dynamic sensor networks with constantly changing topologies. The algorithm does not detect network partitions nor does it attempt to repair them. This is another issue that may be worth exploring. Finally it may be interesting to compare the behaviour of ASCENT using different routing strategies and MAC layer scheduling algorithms.

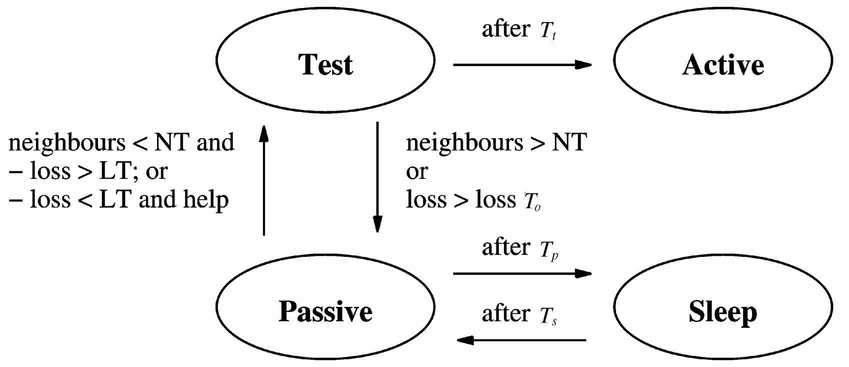

FIG. 6. ASCENT state transitions.

2.1.14. *A robust and light-weight routing mechanism*

Deng *et al.* [20] describe a mechanism to discover alternate routes from a node to the base station when a single node or a set of nodes along the original path to the base station fails. It assumes that the initial route from each node to the base station has already been set up using some other routing protocol such as TinyOS beaconing [6].

The path repair algorithm works in four stages. In first stage called *failure detection*, a node tries to detect if its parent has failed by sending out probe messages. If the parent is alive, it replies with a message that indicates whether it is connected to the base station and if so, the number of hops to the base station. In the second stage called *failure information propagation*, the node informs its children (when they issue probe messages) if its parent is connected to the base station or not. If there is no response from the parent within a timeout period, the node sends a RQST message to its neighbours to adopt a new parent. A neighbour that is connected to the base station replies back with a RPLY message that includes its parent node ID and hop count to the base station. This step is called *new parent detection*. When the disconnected node receives replies from all its neighbours, it picks the one that has the smallest hop count to the base station as its new parent. This is the final step and is known as *new parent selection*. The metric used for new parent selection can be based on other parameters; for example, if many nodes have failed in a certain area, then a node may not be willing to choose a parent nearer to the area of failure. In this case, the location of nodes (determined using localization systems such as GPS or directional antennae) also influences the choice of parent.

The algorithm works when nodes fail randomly in the network as well as when a certain portion of the network fails entirely (*area failure*). When nodes fail in a network, it takes some time for the failures to be detected by all the active nodes. This results in routing inconsistencies such as loops. The path-repair algorithm is capable of eliminating loops resulting from such inconsistencies. The central idea in loop elimination is that the hop count (to the base station) of the nodes in the loop keeps increasing continually. After some limit, neighbouring nodes that have real paths to the base station will attract some of the loop nodes and thus break the loop.

The failure detection stage requires each node to keep its radio receiver on to be able to respond to the keep alive probe messages from its children. This results in unnecessary expenditure of energy. This can be alleviated in conjunction with TDMA-style MAC layer scheduling strategies. The performance of the path repair algorithm needs to be studied with different routing protocols by means of simulations.

2.1.15. *Energy-efficient routing*

Schurgers *et al.* [21] propose a set of techniques to improve the routing in sensor networks. They argue that uniform utilization of resources such as power can be obtained by shaping the traffic flow. For instance, the routing paths for several data streams are likely to share many common nodes. These common nodes burn out faster owing to the heavy load and thereby limit the system lifetime. A traffic flow that spreads the energy utilization over all the nodes uniformly is highly desirable to maximize the lifetime of the network.

Three techniques for spreading the network traffic uniformly are proposed. In the first scheme, a *stochastic model* is used by any node to select the next hop. The randomly chosen links tend to distribute the traffic load across the network. In the *energy-based scheme*, a node that has depleted its energy reserves below a certain threshold discourages neighbouring nodes from forwarding packets to it. This is done by appropriate advertisements to neighbours. In the *stream-based scheme*, a node on the path of one data stream tries to divert traffic from other streams away from it.

2.1.16. *Trajectory-based forwarding*

Conventional routing algorithms treat a route as a discrete set of points bounded by the source and the destination. However, Trajectory-Based Forwarding (TBF) [22] views a route as a continuous function. This approach is motivated by the high density of nodes in a sensor network. The essential idea is to come up with a route trajectory that closely approximates the shape of the physical phenomenon being measured. The trajectory is embedded in each outbound packet that is routed by intermediate nodes along a path that follows the trajectory. The chief advantage of this method is that the actual intermediate nodes are not explicitly named by the path. The redundant distribution of sensor nodes makes it possible for the route trajectory to be followed even when several nodes move or fail.

Trajectory-based forwarding is basically a greedy algorithm wherein each intermediate node attempts to forward packets along an optimal path with respect to the intended trajectory. It is assumed that the nodes are positioned relative to an absolute or relative coordinate system. Nodes determine their position using specialized equipment such as GPS receivers or approximate positioning algorithms. Trajectories are commonly expressed in parametric form. For instance, a straight line with slope α and passing through the point with coordinates (x_1, y_1) is represented using the parameter t as $X(t) = x_1 + t\cos\alpha$, and $Y(t) = y_1 + t\sin\alpha$.

There can be several possible metrics for choosing the next hop along a path. An implementation may select the node that is closest to the trajectory or the node that makes the most progress along the path. Other possibilities include choosing the node at the centroid of the feasible set, or one with the greatest remaining battery power.

The TBF paradigm has been applied as a low-cost solution to many applications including unicast routing, resource and topology discovery, broadcasting, multicasting and multipath routing. It is shown to behave quite well in adverse scenarios such as sparse networks and imprecise positioning systems. There are several open problems with trajectory-based forwarding. The main issue is with specifying and modifying the trajectory, whether to use curve fitting techniques or simply a list of points. Generally the source node determines and specifies the trajectory. However, it may be possible for the destination node to discover better trajectories in certain scenarios. Finally a number of interesting problems arise if the target node becomes mobile.

2.1.17. *Data mules*

Wireless sensor nodes are severely constrained in terms of available memory and power resources. The Data Mules [23] approach aims to address this problem by exploiting mobile agents, termed MULEs (Mobile Ubiquitous LAN Extensions) in the environment. For instance, in a vehicular traffic monitoring application, the vehicles can serve as mobile agents, whereas in a wildlife tracking application, the animals can be used as mobile agents. The MULEs are fitted with transceivers that are capable of short-range wireless communication. They can exchange data with sensors and access points when they move into their vicinity. The advantage of such a scheme in sparse networks is immediately obvious—the sensor nodes do not have to expend large amounts of energy in transmitting their readings to distant base stations. Instead they can do short-range transmissions to nearby agents.

The MULE architecture is organized in three tiers. The bottom tier comprises the fixed wireless sensor nodes, the top tier is made up of wired access points and the intermediate tier consists of MULEs. The MULEs are characterized by large memory reserves and renewable battery power, both of which are not feasible for sensor nodes. Further, MULEs can communicate with one another to improve network connectivity. A sensor node is not dependent on a single MULE, and so the network remains connected even if some MULEs fail. Since the sensor nodes do not participate in multihop routing, there is minimal overhead for the sensors; this feature can be contrasted with traditional ad-hoc networks, wherein each node expends substantial energy in routing packets to and from other nodes.

The main disadvantage of the Data Mules scheme is its high latency. Each sensor node needs to wait for a MULE to come within its transmission radius before it can transfer its readings. Further, it has to reach the access point before the node readings can be delivered. This can be mitigated to some extent by equipping MULEs with long-haul communication equipment such as satellite phones. Another disadvantage is that the system assumes the existence of mobile agents in the target environment, which may not always be present.

Improving the reliability of the system by incorporating acknowledgments in lossy environments remains an important research issue. Two levels of acknowledgments have to be taken into consideration: MULE to sensor node, and access point to MULE. The sensor nodes need to keep their radio receivers on continuously to be able to communicate with MULEs. It is well known that a radio in listening mode consumes a similar amount of energy as when it is actually receiving data. Thus another area of research is to come up with a mechanism in which the nodes can keep their radios off for long periods of time, and only awaken when MULEs approach them.

2.1.18. *Summary of flat routing protocols*

The preceding sections have described several flat routing protocols for sensor networks. Since the routing algorithms have to operate based only on local knowledge, some form of distance vector routing is adopted. However, as discussed previously, distance vector routing protocols converge very slowly. Table I summarizes the different methods adopted by flat sensor network routing protocols to achieve convergence and avoid routing loops.

Other flat routing protocols use some form of *flooding* to construct routes and achieve convergence. For instance, TinyOS beaconing and the Pulse protocol use flooding to build a spanning tree rooted at the base station. In the SPIN family of protocols, each node broadcasts an advertisement message containing metadata to all its neighbours. However, flooding is an expensive operation that is normally avoided by sensor network routing protocols.

Some routing protocols such as ARRIVE use probabilistic forwarding that ensures better traffic distribution. Geographic routing protocols such as GEAR route packets towards geographically defined target regions to model typical user queries.

2.2. *Hierarchical and cluster-based routing protocols*

Hiearchical routing protocols organize the network into groups called *clusters*. Each cluster selects a node that serves as the *cluster-head*. The cluster-head is responsible for collecting the sensor data from all the cluster members, aggregating them and transmitting a summary to the base station. This results in eliminating a large number of redundant messages from the nodes, thereby reducing the overall power consumption in the network. It also avoids many MAC layer collisions that waste the available bandwidth. This enables the sensor network to scale to a large number of nodes.

The disadvantage of cluster-based algorithms is that the base station should be reachable from all the cluster-heads. This drains the power reserves of the cluster-heads quickly, thereby disconnecting the corresponding clusters from the network. It is possible to avoid this problem by periodically rotating the cluster heads among the nodes to ensure uniform energy consumption.

Hierarchical routing protocols can be decomposed into several constituent blocks as depicted in Fig. 7. The dependencies are essentially similar to those for flat routing protocols with a few additions. Since hierarchical routing protocols depend on the formation of clusters, a new Cluster formation block is introduced. Cluster formation involves not only the organization of nodes into groups, but also the election of cluster-heads. Clustering facilitates MAC layer scheduling of transmissions. The cluster-head computes and distributes the MAC schedule among its cluster nodes. Each node transmits only during its time slot; it can switch its radio off during all the other slots thereby conserving energy. Cluster maintenance depends on failure detection and recovery to determine if the cluster-head is alive or not. If the cluster-head has failed, the cluster formation process can be reinitiated. Failure detection in turn can be implemented by techniques like hierarchical heartbeat that are well suited for cluster-based topologies.

The following subsections describe routing protocols that can be classified as hierarchical or cluster-based routing protocols.

2.2.1. *Protocols for self-organization of a wireless sensor network*

Sohrabi *et al.* [15] propose a number of protocols for the self-organization, mobility management, multihop routing and local cooperative information processing in sensor networks. The efficiency of a routing protocol can be improved in three different areas: route setup, route maintenance and service. Algorithms that find good routing paths usually involve complex computations for their maintenance, which implies a heavy energy cost. For a robust, long-lasting, multihop network, it is worthwhile to expend energy initially to establish good paths, since it will pay off during network operation. On the other hand, reducing the initial route setup delay assumes more importance in networks with light data traffic.

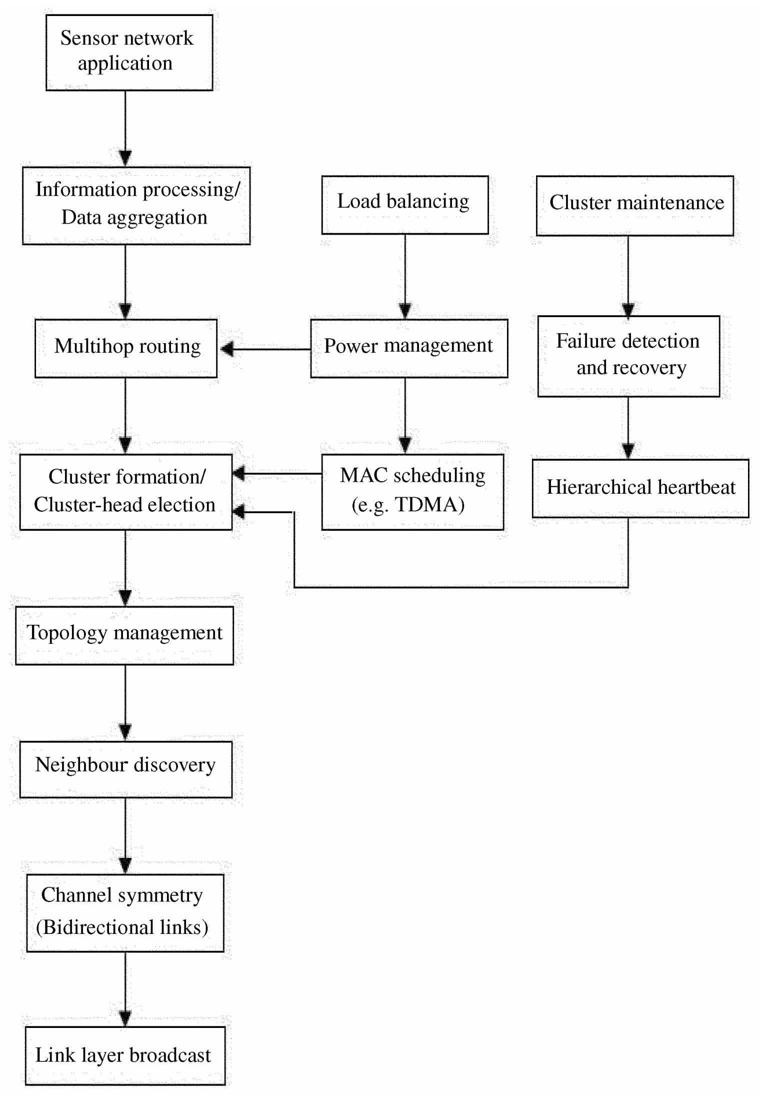
Sensor network application
Information processing/ Data aggregation
Load balancing
Cluster maintenance
Multihop routing
Power management
Failure detection and recovery
Cluster formation/ Cluster-head election
MAC scheduling (e.g. TDMA)
Hierarchical heartbeat
Topology management
Neighbour discovery
Channel symmetry (Bidirectional links)
Link layer broadcast

FIG. 7. Depends-on hierarchy for cluster-based sensor network routing protocols. The function at the tail of an arrow depends on the function at the head of the arrow.

Cooperative signal processing techniques are classified into two categories: *noncoherent* and *coherent* processing. In noncoherent processing, data is preprocessed at each node before sending a summary to a central node for further processing. Thus the communication traffic in such networks is not very heavy. In coherent processing, however, raw sensor data with minimal preprocessing is uploaded to the central node for more intensive computations. This results in heavy network traffic which increases the significance of selecting optimal routing paths.

A procedure for adaptive local routing for cooperative signal processing is presented. This applies to the sensor nodes that are in the vicinity of the sensed phenomenon. In a non-coherent network, a single central node is elected using the *Single Winner Election* (SWE) and *Spanning Tree* algorithms. Nodes that wish to become central nodes broadcast *Elect* messages that contain a set of parameters such as available energy and computational power. The nodes that receive Elect messages compare them with the messages they have already received. If the current Elect message received by a node presents a better candidate, it is added to the node registry and rebroadcast to its neighbours; otherwise the message is discarded. Finally the information about the winning candidate is diffused to all the nodes in the vicinity. Simultaneously a minimum-hop spanning tree rooted at the winning central node is constructed. If each candidate delays its Elect message depending on its likelihood of winning (i.e. the candidate that is most likely to win sends its Elect message first), then a lot of localized message exchange can be avoided.

In coherent networks, where the overhead of sending all the raw data to a central node is high, a simple variation of the SWE algorithm called the *Multi-Winner Election* (MWE) process is employed. Each node now records the first n winners instead of a single winner. The winners here represent the subset of source nodes that will provide the sensor data. The SWE algorithm can then be run on these source nodes to elect a central node that does coherent cooperative function processing. The election of a central node for coherent networks involves higher delay, higher overhead and lower scalability than for noncoherent networks.

2.2.2. *LEACH*

Heinzelman *et al.* [24] describe LEACH (Low Energy Adaptive Clustering Hierarchy), a cluster-based routing protocol. LEACH aims to uniformly distribute the energy consumed by sensor nodes across the network to extend system lifetime. This is accomplished by periodically rotating the cluster head nodes. The cluster heads collect the sensor readings from the other nodes in the cluster, perform local compression or aggregation on the data to reduce global communication and transmit a summary of the readings back to a central base station. Thus the cluster heads are the most critical nodes in the network since the entire cluster would be disconnected if the corresponding cluster-head were to run out of energy. A fundamental assumption of the LEACH algorithm is that nodes can adjust their transmission power to transmit signals to varying distances.

The LEACH algorithm runs in rounds, with each round beginning with a *setup phase* in which the cluster-heads are selected and the clusters are formed, and the *steady-state phase*, in which the sensor data transfer takes place. Each node determines by itself whether to serve as a cluster head or not during the current round, based on its remaining energy level and a predetermined desired percentage of cluster-heads in the network. The algorithm guarantees that each node will become a cluster-head eventually, after some fixed number of rounds. This contributes towards uniform energy dissipation of the nodes.

Once a node decides to act as a cluster-head for the current round, it broadcasts an *advertisement message* to the rest of the nodes. Each of the non-cluster-head nodes affiliate themselves with the cluster-head from which they receive the advertisement message with the highest signal strength, with ties being broken randomly. The cluster-head is informed about this affiliation by a message from each of the affiliating nodes. This process organizes the entire network into clusters, with a single cluster-head for each cluster.

After a cluster-head receives affiliation messages from all the nodes in its cluster, it creates a TDMA schedule and broadcasts it to the nodes. The TDMA schedule divides time into a set of slots, the number of slots being equal to the number of nodes in the cluster. Each node is assigned a unique time slot during which it can transmit its readings to the cluster-head. The advantage of this approach is that a node can turn off its radio transceiver during all of the other time slots, leading to large energy savings. When the cluster-head receives the sensor readings from all of its cluster nodes, it compresses and aggregates them into a composite signal and transmits it to the base station. This transmission potentially requires high energy since the cluster-head may be very distant from the base station. An additional disadvantage of this scheme is that if there is any physical obstruction (such as a tree, a hill or a building) between the cluster-head and the base station, the entire cluster is cut off from the base station.

LEACH is one of the earliest cluster-based routing protocols for sensor networks. Our simulation experiments showed that its probabilistic cluster formation algorithm does not result in entirely uniform cluster sizes. Some clusters end up with only one or two nodes whereas others have a large number of nodes. Further, LEACH does not guarantee base station reachability for all the network nodes. If some cluster-heads are farther from the base station than their maximum transmission range, then all the nodes in their cluster get cut off from the base station for the current round. A possible solution to this problem is to extend the LEACH cluster formation algorithm to multiple hierarchical levels, and ensure that the cluster-heads at the topmost level are always reachable from the base station.

2.2.3. *A secure hierarchical model*

Tubaishat *et al.* [25] propose an energy-efficient level-based hierarchical routing protocol. Additionally, it designs a Secure Routing Protocol for Sensor Networks (SRPSN) that provides protection against different kinds of attacks and guarantees that packets reach the sink from the source even in the presence of intermediate adversaries.

The protocol organizes the sensor nodes hierarchically into levels. This classification into levels is based on the number of neighbours of each node. If a node has a larger number of neighbours, then it will be assigned a higher level. A node discovers the identity and number of its neighbours (NBR) by broadcasting a hello message. This is followed by a round of exchange of node IDs and NBRs to build connected groups or clusters. Nodes which have the highest NBRs become cluster-heads. The cluster-heads are responsible for aggregating and forwarding sensor data. A sensor that is connected to two or more cluster-heads increases its level further and becomes the *root*. The higher level nodes can get depleted of their energy fast since all the traffic is routed towards higher-level nodes. However, root nodes receive messages only from the few cluster-heads and hence increase their power efficiency. The cluster-heads filter and aggregate the sensor node data and forward the summary to the root nodes. Hence, the information available at the higher-level nodes not only encompasses a wide area but is also more meaningful than lower-level nodes.

It is assumed that sensor nodes are equipped with a position-determining system such as GPS. However, to conserve power, only nodes at higher levels activate their GPS, determine their location and broadcast it to their children. This serves as a reasonable approximation for the locations of the children nodes since they must be located physically close to the cluster-head.

The routing table in each node consists of entries of the form <ID, Level, NBR>, where ID, Level and NBR are, respectively, the identifier, level and number of neighbours of each potential next hop neighbour. When a node receives a packet to forward, it chooses the neighbour that is closest to the location of the targeted sink or source node as the next hop node. The level and the number of neighbours (NBR) of each neighbour also help in determining the next hop.

In the SRPSN protocol, each sensor node shares a secret key with the sink node. The sink node maintains a table of (id, key) pairs for all nodes. The protocol runs in two phases: *secure route discovery* and *secure data forwarding*. The secure route discovery phase is initiated by the source node, which sends out a route request (RREQ) message to its neighbours. The RREQ message includes the *ID*s of the source and the sink, an encrypted nonce and a MAC generated using the key shared with the sink node. An intermediate node forwards the RREQ after including its *ID* and its previous hop *ID*. When the sink node receives the RREQ, it verifies the MAC using the shared key, constructs a route reply (RREP) message and broadcasts it. The RREP message consists of the *ID*s of source node, sink node, current node and predecessor node protected by a MAC generated using the shared key. As the RREP message is propagated towards the source node, the routing tables of all the intermediate nodes are updated. The source node verifies that the RREP message originated from the sink node using the MAC. If the source node does not receive the RREP message (possibly due to malicious intermediate nodes), it triggers another route discovery.

Once the route discovery phase completes, the sink node transmits encrypted cluster group keys to nodes using the shared secret keys. The sensor nodes send encrypted data along with an MAC to their cluster-heads using the cluster group key. Secure inter-cluster communication works similar to the route discovery phase.

The routing protocol uses a naive clustering algorithm based on the number of neighbours as the metric. This can lead to nonuniform clusters depending on the distribution of the sensor nodes. Thus the nodes having the largest number of neighbours can burn out faster. A better clustering algorithm that can yield uniform clusters will be a useful improvement to the protocol.

2.2.4. *Summary of hierarchical and cluster-based routing protocols*

Hierarchical routing protocols greatly increase the scalability of a sensor network. The overall energy consumption of the nodes is reduced, leading to prolonged network lifetime. The organization of the network into clusters lends itself to efficient data aggregation which in turn results in better utilization of the channel bandwidth. Cluster-based routing holds great promise for many-to-one and one-to-many communication paradigms that are prevalent in sensor networks.

Table III
Comparison of sensor network routing protocols

	GPS required	Multi-path routing	MAC scheduling (TDMA)	Mobility aware	Event driven	Energy distribution	Flooding involved	Intrusion tolerant	Failure recovery
TinyOS beaconing	No	No	No	No	No	Non-uniform	Yes	No	No
Pulse	No	No	Yes	No	No	Non-uniform	Yes	No	Yes
Directed diffusion	Yes	No	No	Yes	No	Non-uniform	Yes	No	Yes
Rumor Routing	No	No	No	Yes	Yes	Non-uniform	Partly	No	Yes
SPIN	No	No	Yes	No	Yes	Uniform	Partly	No	Yes
Braided multipath routing	No	Yes	No	Yes	No	Non-uniform	No	No	Yes
GRAB	No	Yes	Yes	Yes	Yes	Non-uniform	No	No	Yes
INSENS	No	Yes	Yes	No	No	Non-uniform	Yes	Yes	No
SAR	No	Yes	Yes	Yes	No	Uniform	Yes	No	Yes
GEAR	Yes	No	No	No	No	Uniform	Partly	No	Yes
ARRIVE	No	Yes	Yes	No	No	Non-uniform	Partly	Partly	Yes
ASCENT	No	No	Yes	No	No	Non-uniform	No	No	Yes
Robust routing	Optional	No	No	No	No	Non-uniform	No	No	Yes
TBF	Yes	Yes	No	Yes	No	Non-uniform	Partly in broadcasting	No	Yes
Data MULEs	No	No	No	Yes	No	Uniform	No	No	Yes
SWE/MWE	No	No	Yes	Yes	No	Uniform	Yes	No	No
LEACH	No	No	Yes	No	No	Uniform	No	No	Yes
SRPSN	Yes	No	No	No	No	Uniform	Yes	Yes	Yes

3. Conclusions

Wireless sensor networks are increasingly being used in military, environmental, health and commercial applications. The problem of relaying data from remote sensor nodes to a central base station is of paramount importance in such applications. Severe resource constraints in the form of limited computation, memory and power make the problem of routing interesting and challenging. Sensor networks are inherently different from traditional wired networks as well as wireless ad-hoc networks. However, routing algorithms for sensor networks have borrowed liberally from the existing algorithms for ad-hoc networks. In this paper we have tried to explore the space of sensor network routing. An attempt has been made to summarize many of the proposed routing algorithms. The routing protocols have been classified into distinct categories and their advantages and disadvantages have been discussed. The routing protocols are compared in Table II and classified in Table III. This survey will hopefully motivate future researchers to come up with smarter and more robust routing techniques.

References

1. J. M. Kahn, R. H. Katz, and K. S. J. Pister, Next century challenges: Mobile networking for smart dust, *Proc. Fifth Annual ACM/IEEE Int. Conf. Mobile Computing and Networking* (*Mobicom 99*), 1999, pp. 271–278.
2. J. Rabaey, J. Ammer, J. L. da Silva Jr, and D. Patel, Pico-radio: ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes, *Proc. VLSI* (WVLSI '00), Orlando, 2000.
3. Anish Arora *et al.*, ExScal: Elements of an extreme scale wireless sensor network, *Proc. 11 IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications* (RTCSA '05), 2005, pp. 102–108.
4. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless sensor networks: a survey, *Computer Networks*, **38**, 393–422 (2002).
5. Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, Analyzing the energy consumption of security protocols, *Proc 2003 Int. Symp. on Low Power Electronics and Design* (ISLPED), 2003, pp. 30–35.
6. Radia Perlman, *Interconnections: Bridges*, *Routers*, *Switches*, and *Internetworking protocols*, Second edition, Addison-Wesley (2000).
7. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, System architecture directions for networked sensors, *Proc. ACM Architectural Support for Programming Languages and Operating Systems* (ASPLOS IX), 2000.
8. Baruch Awerbuch, David Holmer, Herbert Rubens, Kirk Chang, and I. J. Wang, The Pulse protocol: sensor network routing and power saving, *Military Communications Conf.* (MIL-COM 2004), 2004.
9. Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, *Proc. Sixth Annual Int. Conf. Mobile Computing and Networking* (Mobicom '00), Boston, 2000.
10. David Braginsky, and Deborah Estrin, Rumor routing algorithm for sensor networks, *First ACM Int. Workshop on Wireless Sensor Networks and Applications* (WSNA), Atlanta, Sept. 28, 2002, pp. 22–31.
11. Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin, Highly-resilient, energy-efficient multipath routing in wireless sensor networks, *ACM SIGMOBILE*, *Mobile Computing Commun. Rev.*, **15**, 11–25 (2001).
12. Joanna Kulik, Wendi Heinzelman, and Hari Balakrishnan, Negotiation-based protocols for disseminating information in wireless sensor networks, *Wireless Networks*, **8**, 169–185 (2002).
13. Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang, GRAdient broadcast: A robust data delivery protocol for large scale sensor networks, *ACM Wireless Networks J.* (*WINET*), **11**, 285–298 (2005).
14. Jing Deng, Richard Han, and Shivakant Mishra, INSENS: Intrusion-tolerant routing in wireless sensor networks, Poster paper, *23rd IEEE Int. Conf. on Distributed Computing Systems* (*ICDCS 2003*), Rhode Island, May 2003.
15. Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J. Pottie, Protocols for self-organization of a wireless sensor network, *IEEE Personal Commun.*, **7**, 16–27 (2000).
16. Yan Yu, Ramesh Govindan, and Deborah Estrin, *Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks*, Technical Report UCLA/CSD-TR-01-0023, Computer Science Department, University of California at Los Angeles (2001).
17. Chris Karlof, Yaping Li, and Joseph Polastre, *ARRIVE: Algorithm for robust routing in volatile environments*, Technical Report, UCB/CSD-03-1233, Computer Science Department, University of California at Berkeley (2002).
18. Alec Woo, Terence Tong, and David Culler, Taming the underlying challenges of reliable multihop routing in sensor networks, *ACM Conf. on Embedded Networked Sensor Systems* (*SenSys '03*), 2003.
19. Alberto Cerpa, and Deborah Estrin, ASCENT: Adaptive self-configuring sEnsor networks topologies, *IEEE Trans. on Mobile Computing, Special Issue on Mission-Oriented Sensor Networks*, **3**, 272–285 (2004).
20. Jing Deng, Richard Han, and Shivakant Mishra, A robust and light-weight routing mechanism for wireless sensor networks, *Proc. 1st Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks* (*DIWANS 2004*), 2004.

21. Curt Schurgers, and Mani B. Srivastava, Energy efficient routing in wireless sensor networks, *Military Commun. Conf.* (*MILCOM 2001*), 2001.

22. Dragos Niculescu, and Badri Nath, Trajectory based forwarding and its applications, *Proc. Ninth Annual Int. Conf. on Mobile Computing and Networking* (*Mobicom '03*), 2003.

23. Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette, Data MULEs: Modeling a three-tier architecture for sparse sensor networks, *Proc. First IEEE Int. Workshop on Sensor Network Protocols and Applications*, 2003.

24. Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, *Proc. 33rd Hawaii Int. Conf. on System Sciences*, 2000.

25. Malik Tubaishat, Jian Yin, Biswajit Panja, and Sanjay Madria, A secure hierarchical model for sensor network, *ACM SIGMOD Record*, **33**, 7–13 (2004).

26. Chris Karlof, and David Wagner, Secure routing in wireless sensor networks: Attacks and countermeasures, *Ad Hoc Networks*, **1**, 293–315 (2003).

27. Jamal N. Al-Karaki, and Ahmed E. Kamal, Routing techniques in wireless sensor networks: a survey, *IEEE Wireless Commun.*, **11**, 6–28 (2004).