LECTURE NOTES

ON

ADHOC SENSOR NETWORKS

IV YEAR B. TECH II SEMESTER

UNIT-1

Mobile Ad hoc Networking

1. Introduction

Simply stating, a Mobile Ad hoc NETwork (MANET) is one that comes together as needed, not necessarily with any support from the existing Internet infrastructure or any other kind of fixed stations. We can formalize this statement by defining an ad hoc network as an autonomous system of mobile hosts (also serving as routers) connected by wireless links, the union of which forms a communication network modeled in the form of an arbitrary graph. This is in contrast to the well-known single hop cellular network model that supports the needs of wireless communication by installing base stations as access points. In these cellular networks, communications between two mobile nodes completely rely on the wired backbone and the fixed base stations. In a MANET, no such infrastructure exists and the network topology may dynamically change in an unpredictable manner since nodes are free to move.

As for the mode of operation, ad hoc networks are basically peer-to-peer multi-hop mobile wireless networks where information packets are transmitted in a store-and-forward manner from a source to an arbitrary destination, via intermediate nodes as shown in Figure 1. As the nodes move, the resulting change in network topology must be made known to the other nodes so that outdated topology information can be updated or removed. For example, as MH2 in Figure 1 changes its point of attachment from MH3 to MH4 other nodes part of the network should use this new route to forward packets to MH2.

Note that in Figure 1, and throughout this text, we assume that it is not possible to have all nodes within range of each other. In case all nodes are close-by within radio range, there are no routing issues to be addressed. In real situations, the power needed to obtain complete connectivity may be, at least, infeasible, not to mention issues such as battery life. Therefore, we are interested in scenarios where only few nodes are within radio range of each other.

Figure 1 raises another issue of symmetric (bi-directional) and asymmetric (unidirectional) links. As we shall see later on, some of the protocols we discuss consider symmetric links with associative radio range, i.e., if (in Figure 1) MH1 is within radio range of MH3, then MH3 is also within radio range of MH1. This is to say that the communication links are symmetric. Although this assumption is not always valid, it is usually made because routing in asymmetric networks is a relatively hard task [Prakash 1999]. In certain cases, it is possible to find routes that could avoid asymmetric links, since it is quite likely that these links imminently fail. Unless stated otherwise, throughout this text we consider symmetric links, with all nodes having identical capabilities and responsibilities.

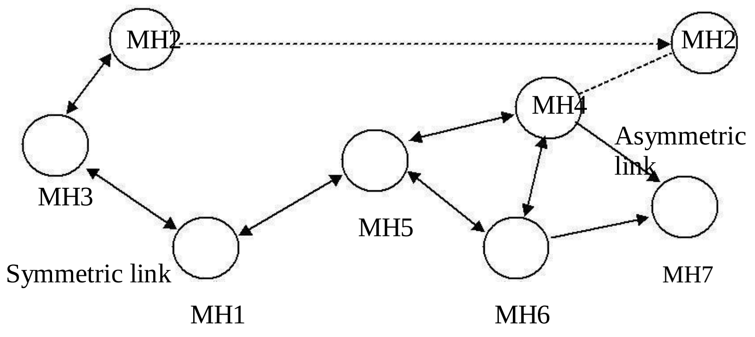


Figure 1 – A Mobile Ad hoc network

The issue of symmetric and asymmetric links is one among the several challenges encountered in a MANET. Another important issue is that different nodes often have different mobility patterns. Some nodes are highly mobile, while others are primarily stationary. It is difficult to predict a node’s movement and pattern of movement. Table 1 summarizes some of the main characteristics [Duggirala 2000] and challenges faced in a MANET.

Wireless Sensor Networks [Estrin 1999, Kahn 1999] is an emerging application area for ad hoc networks which has been receiving a large attention. The idea is that a collection of cheap to manufacture, stationary, tiny sensors would be able to sense, coordinate activities and transmit some physical characteristics about the surrounding environment to an associated base station. Once placed in a given environment, these sensors remain stationary. Furthermore, it is expected that power will be a major driving issue behind protocols tailored to these networks, since the lifetime of the battery usually defines the sensor’s lifetime. One of the most cited examples is the battlefield surveillance of enemy’s territory wherein a large number of sensors are dropped from an airplane so that activities on the ground could be detected and communicated. Other potential commercial fields include machinery prognosis, bio sensing and environmental monitoring.

This rest of this text is organized as follows. We initially provide necessary background on ad hoc networking by illustrating its diverse applications. Next, we cover the routing aspect in a MANET, considering both unicast and multicast communication. MAC issues related to a MANET are then illustrated. Following, sensor networks, its diverse applications, and associated routing protocols are discussed. Finally, we conclude this text by discussing the current standard activities at both IETF and the Bluetooth SIG, and also bringing up some open problems that have not received much attention so far and still need to be addressed.

Applications of MANETs

There are many applications to ad hoc networks. As a matter of fact, any day-to-day application such as electronic email and file transfer can be considered to be easily deployable within an ad hoc network environment. Web services are also possible in case any node in the network can serve as a gateway to the outside world. In this discussion, we need not emphasize the wide range of military applications possible with ad hoc networks. Not to mention, the technology was initially developed keeping in mind the military applications, such as battlefield in an unknown territory where an infrastructured network is almost impossible to have or maintain. In such situations, the ad hoc networks having self-organizing capability can be effectively used where other technologies either fail or cannot be deployed effectively. Advanced features of wireless mobile systems, including data rates compatible with multimedia applications, global roaming capability, and coordination with other network structures, are enabling new applications. Some well-known ad hoc network

applications are:

- Collaborative Work – For some business environments, the need for collaborative computing might be more important outside office environments than inside. After all, it is often the case where people do need to have outside meetings to cooperate and exchange information on a given project.

- Crisis-management Applications – These arise, for example, as a result of natural disasters where the entire communications infrastructure is in disarray. Restoring communications quickly is essential. By using ad hoc networks, an infrastructure could be set up in hours instead of days/weeks required for wire-line communications.
- Personal Area Networking and Bluetooth – A personal area network (PAN) is a short-range, localized network where nodes are usually associated with a given person. These nodes could be attached to someone's pulse watch, belt, and so on.

  In these scenarios, mobility is only a major consideration when interaction among several PANs is necessary, illustrating the case where, for instance, people meet in real life. Bluetooth [Haarsten 1998], is a technology aimed at, among other things, supporting PANs by eliminating the need of wires between devices such as printers, PDAs, notebook computers, digital cameras, and so on, and is discussed later.

Routing in a MANET

It has become clear that routing in a MANET is intrinsically different from traditional routing found on infra structured networks. Routing in a MANET depends on many factors including topology, selection of routers, and initiation of request, and specific underlying characteristic that could serve as a heuristic in finding the path quickly and efficiently. The low resource availability in these networks demands efficient utilization and hence the motivation for optimal routing in ad hoc networks. Also, the highly dynamic nature of these networks imposes severe restrictions on routing protocols specifically designed for them, thus motivating the study of protocols which aim at achieving routing stability.

One of the major challenges in designing a routing protocol [Jubin 1987] for ad hoc networks stems from the fact that, on one hand, a node needs to know at least the reachability information to its neighbors for determining a packet route and, on the other hand, the network topology can change quite often in an ad hoc network. Furthermore, as the number of network nodes can be large, finding route to the destinations also requires large and frequent exchange of routing control information among the nodes. Thus, the amount of update traffic can be quite high, and it is even higher when high mobility nodes are present. High mobility nodes can impact route maintenance overhead of routing protocols in such a way that no bandwidth might remain leftover for the transmission of data packets [Corson 1996].

1.1 Proactive and Reactive Routing Protocols

Ad hoc routing protocols can be broadly classified as being Proactive (or table-driven) or Reactive (on-demand). Proactive protocols mandates that nodes in a MANET should keep track of routes to all possible destinations so that when a packet needs to be forwarded, the route is already known and can be immediately used. On the other hand, reactive protocols employ a lazy approach whereby nodes only discover routes to destinations on demand, i.e., a node does not need a route to a destination until that destination is to be the sink of data packets sent by the node.

Proactive protocols have the advantage that a node experiences minimal delay whenever a route is needed as a route is immediately selected from the routing table. However, proactive protocols may not always be appropriate as they continuously use a substantial fraction of the network capacity to maintain the routing information current. To cope up with this shortcoming, reactive protocols adopt the inverse approach by finding a route to a destination only when needed. Reactive protocols often consume much less bandwidth than proactive protocols, but the delay to determine a route can be significantly high and they will typically experience a long delay for discovering a route to a destination prior to the actual communication. In brief, we can conclude that no protocol is suited for all possible environments, while some proposals using a hybrid approach have been suggested.

Unicast Routing Protocols

1.1 Proactive Routing Approach

In this section, we consider some of the important proactive routing protocols.

1.1.1 Destination-Sequenced Distance-Vector Protocol

The destination-sequenced distance-vector (DSDV) [Perkins 1994] is a proactive hop-by-hop distance vector routing protocol, requiring each node to periodically broadcast routing updates. Here, every mobile node in the network maintains a routing table for all possible destinations within the network and the number of hops to each destination. Each entry is marked with a sequence number assigned by the destination node. The sequence numbers enable the mobile nodes to distinguish stale routes from new ones, thereby avoiding the formation of routing loops. Routing table updates are periodically transmitted throughout the network in order to maintain consistency in the table.

To alleviate the potentially large amount of network update traffic, route updates can employ two possible types of packets: full dumps or small increment packets. A full dump type of packet carries all available routing information and can require multiple network protocol data units (NPDUs). These packets are transmitted infrequently during periods of occasional movement. Smaller incremental packets are used to relay only the information that has changed since the last full dump. Each of these broadcasts should fit into a standard-size NPDU, thereby decreasing the amount of traffic generated. The mobile nodes maintain an additional table where they store the data sent in the incremental routing information packets. New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the smaller metric is used in order to optimize (shorten) the path. Mobiles also keep track of settling time of the routes, or the weighted average time that routes to a destination could fluctuate before the route with the best metric is received. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce

network traffic and optimize routes by eliminating those broadcasts that would occur if a better route could be discovered in the very near future.

Note that each node in the network advertises a monotonically increasing sequence number for itself. The consequence of doing it so is that when a node B decides that its route to a destination D is broken, it advertises the route to D with an infinite metric and a sequence number one greater than its sequence number for the route that has broken (making an odd sequence number). This causes any node A routing packets through B to incorporate the infinite-metric route into its routing table until node A hears a route to D with a higher sequence number.

1.1.2 The Wireless Routing Protocol

The Wireless Routing Protocol (WRP) [Murthy 1996] described is a table-based protocol with the goal of maintaining routing information among all nodes in the network. Each node in the network is responsible for maintaining four tables: Distance table, Routing table, Link-cost table, and the Message Retransmission List (MRL) table. Each entry of the MRL contains the sequence number of the update message, a re-transmission counter, an acknowledgment-required flag vector with one entry per neighbor, and a list of updates sent in the update message. The MRL records which updates in an update message need to be retransmitted and neighbors should acknowledge the retransmission.

Mobiles inform each other of link changes through the use of update messages. An update message is sent only between neighboring nodes and contains a list of updates (the destination, the distance to the destination, and the predecessor of the destination), as well as a list of responses indicating which mobiles should acknowledge (ACK) the update. After processing updates from neighbors or detecting a change in a link, mobiles send update messages to a neighbor. In the event of the loss of a link between two nodes, the nodes send update messages to their neighbors. The neighbors then modify their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

Nodes learn about the existence of their neighbors from the receipt of acknowledgments and other messages. If a node is not sending messages, it must send a hello message within a specified time period to ensure connectivity. Otherwise, the lack of messages from the node indicates the failure of that link; this may cause a false alarm. When a mobile receives a hello message from a new node, that new node is added to the mobile’s routing table, and the mobile sends the new node a copy of its routing table information.

Part of the novelty of WRP stems from the way in which it achieves freedom from loops. In WRP, routing nodes communicate the distance and second-to-last hop information for each destination in the wireless networks. WRP belongs to the class of path-finding algorithms with an important exception. It avoids the “count-to-infinity” problem by forcing each node to perform consistency checks of predecessor information reported by all its neighbors. This ultimately (although not instantaneously) eliminates looping situations and provides faster route convergence when a link failure occurs.

1.2Reactive Routing Approach

In this section, we describe some of the most cited reactive routing protocols.

1.2.1 Dynamic Source Routing

The Dynamic Source Routing (DSR) [Johnson 1996] algorithm is an innovative approach to routing in a MANET in which nodes communicate along paths stored in source routes carried by the data packets. It is referred as one of the purest examples of an on-demand protocol [Perkins 2001]. In DSR, mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are continually updated as new routes are learned. The protocol consists of two major phases: route discovery and route maintenance. When a mobile node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it will use this route to send the packet. On the other hand, if the node does not have such a route, it initiates route discovery by broadcasting a route request packet. This route request contains the address of the destination, along with the source node’s address and a unique identification number. Each node receiving the packet checks whether it knows of a route to the destination. If it does not, it adds its own address to the route record of the packet and then forwards the packet along its outgoing links.

To limit the number of route requests propagated on the outgoing links of a node, a mobile node only forwards the route request if the request has not yet been seen by the mobile and if the mobile’s address does not already appear in the route record.

A route reply is generated when the route request reaches either the destination itself, or an intermediate node that contains in its route cache an unexpired route to the destination. By the time the packet reaches either the destination or such an intermediate node, it contains a route record yielding the sequence of hops taken. Figure 2(a) illustrates the formation of the route record as the route request propagates through the network. If the node generating the route reply is the destination, it places the route record contained in the route request into the route reply. If the responding node is an intermediate node, it appends its cached route to the route record and then generates the route reply. To return the route reply, the responding node must have a route to the initiator. If it has a route to the initiator in its route cache, it may use that route. Otherwise, if symmetric links are supported, the node may reverse the route in the route record. If symmetric links are not supported, the node may initiate its own route discovery and piggyback the route reply on the new route request. Figure 2(b) shows the transmission of route record back to the source node.

Route maintenance is accomplished through the use of route error packets and acknowledgments. Route error packets are generated at a node when the data link layer encounters a fatal transmission problem. When a route error packet is received, the hop in error is removed from the node's route cache and all routes containing the hop are truncated at that point. In addition to route error messages, acknowledgments are used to verify the correct operation of the route links. These include passive acknowledgments, where a mobile is able to hear the next hop forwarding the packet along the route.

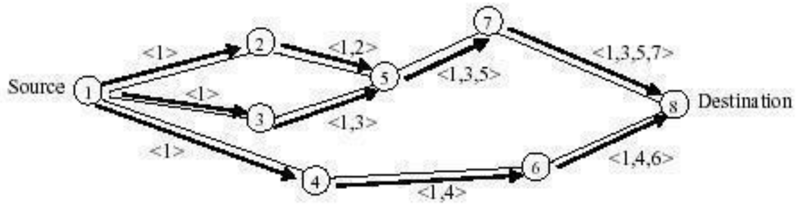


Figure 2(a) – Route discovery in DSR

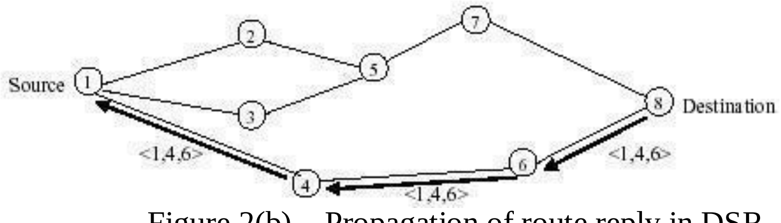


Figure 2(b) – Propagation of route reply in DSR

3.2.2 The Ad Hoc On-Demand Distance Vector Protocol

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol [Perkins 1999] is basically a combination of DSDV and DSR. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV. AODV minimizes the number of required broadcasts by creating routes on an on-demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm. Authors of AODV classify it as a pure on-demand route acquisition system since nodes that are not on a selected path, do not maintain routing information or participate in routing table exchanges. It supports only symmetric links with two different phases:

- Route Discovery, Route Maintenance; and
- Data forwarding.

When a source node desires to send a message and does not already have a valid route to the destination, it initiates a path discovery process to locate the corresponding node. It broadcasts a route request (RREQ) packet to its neighbors, which then forwards the request to their neighbors, and so on, until either the destination or an intermediate node with a “fresh enough” route to the destination is located. Figure 3(a) illustrates the propagation of the broadcast RREQs across the network. AODV utilizes destination sequence numbers to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number, as well as a broadcast ID. The broadcast ID is incremented for every RREQ the node initiates, and together with the node’s IP address, uniquely identifies an RREQ. Along with the node’s sequence number and the broadcast ID, the RREQ includes the most recent sequence number it has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ.

During the process of forwarding the RREQ, intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received, thereby establishing a reverse path. If additional copies of the same RREQ are later received, these packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by unicasting a route reply (RREP) packet back to the neighbor from which it first received the RREQ (Figure 3(b)). As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables that point to the node from which the RREP came. These forward route entries indicate the active forward route. Associated with each route entry is a route timer which causes the deletion of the entry if it is not used within the specified lifetime. Because the RREP is forwarded along the path established by the RREQ, AODV only supports the use of symmetric links.

Routes are maintained as follows. If a source node moves, it is able to reinitiate the route discovery protocol to find a new route to the destination. If a node along the route moves, its upstream neighbor notices the move and propagates a link failure notification message (an RREP with infinite metric) to each of its active upstream neighbors to inform them of the breakage of that part of the route. These nodes in turn propagate the link failure notification to their upstream neighbors, and so on until the source node is reached. The source node may then choose to re-initiate route discovery for that destination if a route is still desired. An additional aspect of the protocol is the use of hello messages, periodic local broadcasts by a node to inform each mobile node of other nodes in its neighborhood. Hello messages can be used to maintain the local connectivity of a node. However, the use of hello messages may not be required at all times. Nodes listen for re-transmission of data packets to ensure that the next hop is still within reach. If such a re-transmission is not heard, the node may use one of a number of techniques, including the use of hello messages themselves, to determine whether the next hop is within its communication range. The hello messages may also list other nodes from which a mobile node has recently heard, thereby yielding greater knowledge of network connectivity.

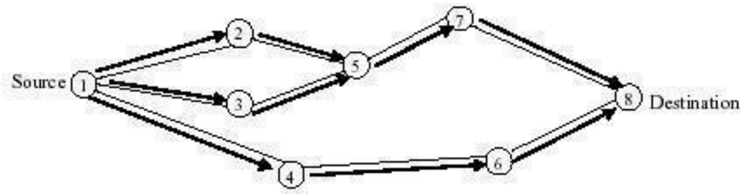
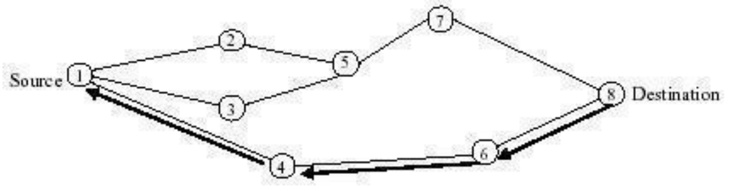


Figure 3(a) – Propagation of RREQ in AODV



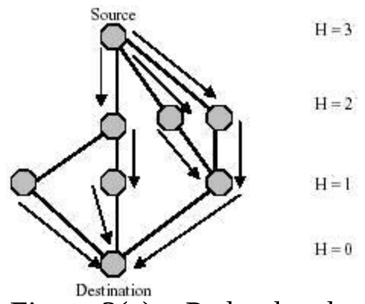


Figure 3(a) – Path taken by the RREP in AODV

3.2.3 Link Reversal Routing and TORA

The Temporally Ordered Routing Algorithm (TORA) [Park 1997] is a highly adaptive loop-free distributed routing algorithm based on the concept of link reversal. It is designed to minimize reaction to topological changes. A key design concept in TORA is that it decouples the generation of potentially far-reaching control messages from the rate of topological changes. Such messaging is typically localized to a very small set of nodes near the change without having to resort to a dynamic, hierarchical routing solution with its added complexity. Route optimality (shortest-path) is considered of secondary importance, and longer routes are often used if discovery of newer routes could be avoided. TORA is also characterized by a multipath routing capability.

The actions taken by TORA can be described in terms of water flowing downhill towards a destination node through a network of tubes that models the routing state of the real network. The tubes represent links between nodes in the network, the junctions of tubes represent the nodes, and the water in the tubes represents the packets flowing towards the destination. Each node has a height with respect to the destination that is computed by the routing protocol. If a tube between nodes A and B becomes blocked such that water can no longer flow through it, the height of A is set to a height greater than that of any of its remaining neighbors, such that water will now flow back out of A (and towards the other nodes that had been routing packets to the destination via A). Figure 4 illustrates the use of the height metric. It is simply the distance from the destination node.

TORA is proposed to operate in a highly dynamic mobile networking environment. It is source initiated and provides multiple routes for any desired source/destination pair. To accomplish this, nodes need to maintain routing information about adjacent (one-hop) nodes. The protocol performs three basic functions:

- Route creation,
- Route maintenance, and
- Route erasure.

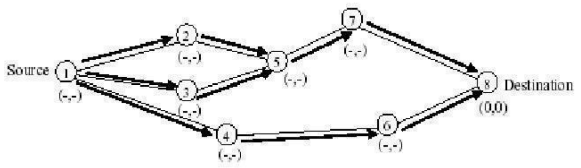
For each node in the network, a separate directed acyclic graph (DAG) is maintained for each destination. When a node needs a route to a particular destination, it broadcasts a QUERY packet containing the address of the destination for which it requires a route. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination. The recipient of the QUERY then broadcasts an UPDATE packet listing its height with respect to the destination. As this packet propagates through the network, each node that receives the UPDATE sets its height to a value greater than the height of the neighbor from which the UPDATE has been received. This has the effect of creating a series of directed links from the original sender of the QUERY to the node that initially generated the UPDATE. When a node discovers that a route to a destination is no longer valid, it adjusts its height so that it is a local maximum with respect to its neighbors and transmits an UPDATE packet. If the node has no neighbors of finite height with respect to this destination, then the node instead attempts to discover a new route as described above. When a node detects a network partition, it generates a CLEAR packet that resets routing state and removes invalid routes from the network.

TORA is layered on top of IMEP, the Internet MANET Encapsulation Protocol [Corson 1997], which is required to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, plus notification to the routing protocol whenever a link to one of its neighbors is created or broken. To reduce overhead, IMEP attempts to aggregate many TORA and IMEP control messages (which IMEP refers to as objects) together into a single packet (as an object block) before transmission. Each block carries a sequence number and a response list of other nodes from which an ACK has not yet been received, and only those nodes acknowledge the block when receiving it; IMEP retransmits each block with some period, and continues to retransmit it if needed for some maximum total period, after which TORA is notified of each broken link to unacknowledged nodes. For link status sensing and maintaining a list of a node's neighbors, each IMEP node periodically transmits a BEACON (or "BEACON-equivalent") packet, which is answered by each node hearing it with a HELLO (or "HELLO-equivalent") packet.

As we mentioned earlier, during the route creation and maintenance phases, nodes use the “height” metric to establish a DAG rooted at the destination. Thereafter, links are assigned a direction (upstream or downstream) based on the relative height metric of neighboring nodes as shown in Figure 5(a). In times of node mobility the DAG route is broken, and route maintenance is necessary to reestablish a DAG rooted at the same destination. As shown in Figure 5(b), upon failure of the last downstream link, a node generates a new reference level that effectively coordinates a structured reaction to the failure. Links are reversed to reflect the change in adapting to the new reference level. This has the same effect as reversing the direction of one or more links when a node has no downstream links.

Timing is an important factor for TORA because the “height” metric is dependent on the logical time of a link failure; TORA assumes that all nodes have synchronized clocks (accomplished via an external time source such as the Global Positioning System). TORA’s metric is a quintuple comprising five elements, namely:

- Logical time of a link failure,
- The unique ID of the node that defined the new reference level,
- A reflection indicator bit,
- A propagation ordering parameter,
- The unique ID of the node.



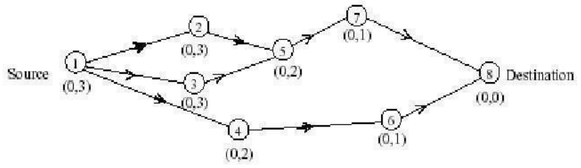


Figure 5(a) – Propagation of the query message

Figure 5(b) – Node’s height updated as a result of the update message

The first three elements collectively represent the reference level. A new reference level is defined each time a node loses its last downstream link due to a link failure. TORA’s route erasure phase essentially involves flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes. In TORA, there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing routes, and building new routes based on each other (Figure 6). Because TORA uses inter-nodal coordination, its instability is similar to the “count-to-infinity” problem, except that such oscillations are temporary and route convergence ultimately occurs. Note that TORA is partially proactive and partially reactive. It is reactive in the sense that route creation is initiated on demand. However, route maintenance is done on a proactive basis such that multiple routing options are available in case of link failures.

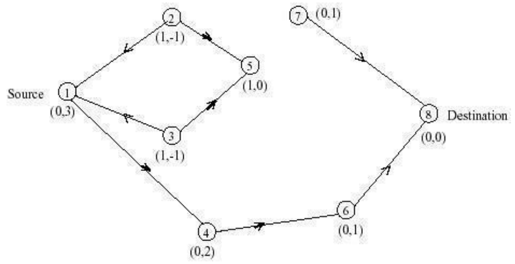


Figure 6 – Route maintenance in TORA

3.2.4 Routing Using Location Information

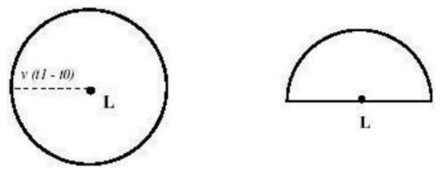
In this section we discuss some ad hoc routing protocols that take advantage of some sort of location information in the routing process.

3.2.4.1 Location-Aided Routing

The Location-Aided Routing (LAR) [Ko 1998] protocol exploits location information to limit the scope of route request flood employed in protocols such as AODV and DSR. Such location information can be obtained through GPS (Global Positioning System). LAR limits the search for a route to the so-called request zone, determined based on the expected location of the destination node at the time of route discovery. Two concepts are important to understand how LAR works: Expected Zone and Request Zone.

Let us first discuss what is an Expected Zone. Consider a node S that needs to find a route to node D. Assume that node S knows that node D was at location L at time t0, and that the current time is t1. Then, the ―expected zone‖ of node D, from the viewpoint of node S at time t1, is the region expected to contain node D. Node S can determine the expected zone based on the knowledge that node D was at location L at time t0. For instance, if node S knows that node D travels with average speed v, then S may assume that the expected zone is the circular region of radius v(t1 - t0), centered at location L (see Figure 7(a)). If actual speed happens to be larger than the average, then the destination may actually be outside the expected zone at time t1. Thus, expected zone is only an estimate made by node S to determine a region that potentially contains D at time t1.

If node S does not know a previous location of node D, then node S cannot reasonably determine the expected zone (the entire region that may potentially be occupied by the ad hoc network is assumed to be the expected zone). In this case, LAR reduces to the basic flooding algorithm. In general, having more information regarding mobility of a destination node can result in a smaller expected zone. For instance, if S knows that destination D is moving north, then the circular expected zone in Figure 7(a) can be reduced to the semi-circle of Figure 7(b).



(a) (b) Figure 7 – Examples of expected zone

Based on the expected zone, we can define the request zone. Again, consider node S that needs to determine a route to node D. The proposed LAR algorithms use flooding with one modification. Node S defines (implicitly or explicitly) a request zone for the route request. A node forwards a route request only if it belongs to the request zone (unlike the flooding algorithm in AODV and DSR). To increase the probability that the route request will reach node D, the request zone should include the expected zone (described above). Additionally, the request zone may also include other regions around the request zone.

Based on this information, the source node S can thus determine the four corners of the expected zone. S includes their coordinates with the route request message transmitted when initiating route discovery. When a node receives a route request, it discards the request if the node is not within the rectangle specified by the four corners included in the route request. For instance, in Figure 8, if node I receives the route request from another node, node I forwards the request to its neighbors, because I determines that it is within the rectangular request zone. However, when node J receives the route request, node J discards the request, as node J is not within the request zone (see Figure 8).

The algorithm just described in the called LAR scheme 1. The LAR scheme 2 is a slight modification to include two pieces of information within the route request packet: assume that node S knows the location (Xd; Yd) of node D at some time t0 – the time at which route discovery is initiated by node S is t1, where t1 ≥ t0. Node S calculates its distance from location (Xd; Yd), denoted as DISTS, and includes this distance with the route request message. The coordinates (Xd; Yd) are also included in the route request packet. With this information, a given node J forwards a route request forwarded by I (originated by node S), if J is within an expected distance from (Xd; Yd) than node I.

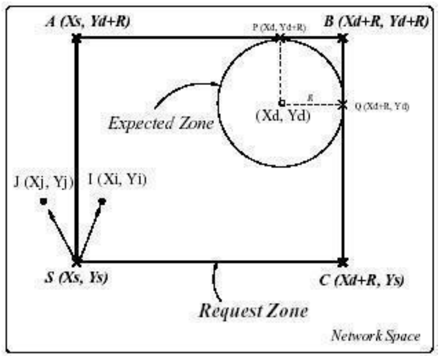


Figure 8 – LAR scheme

3.2.4.2 Distance Routing Effect Algorithm for Mobility

DREAM (Distance Routing Effect Algorithm for Mobility) [Basagni 1998] is a routing protocol for ad hoc networks built around two novel observations. One, called the distance effect, uses the fact that the greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Accordingly, the location information in routing tables can be updated as a function of the distance separating nodes without compromising the routing accuracy. The second idea is that of triggering the sending of location updates by the moving nodes autonomously, based only on a node's mobility rate.Intuitively, it is clear that in a directional routing algorithm, routing information about the slower moving nodes needs to be updated less frequently than that about highly mobile nodes. In this way each node can optimize the frequency at which it sends updates to the networks and correspondingly reduce the bandwidth and energy used, leading to a fully distributed and self-optimizing system. Based on these routing tables, the proposed directional algorithm sends messages in the “recorded direction” of the destination node, guaranteeing delivery by following the direction with a given probability.

3.2.4.3 Relative Distance Micro-Discovery Ad Hoc Routing

The RDMAR (Relative Distance Micro-discovery Ad Hoc Routing) routing protocol [Aggelou 1999] is a highly adaptive, efficient and scalable routing protocol. It is well-suited in large mobile networks whose rate of topological changes is moderate. A key concept in its design is that protocol reaction to link failures is typically localized to a very small region of the network near the change. This desirable behavior is achieved through the use of a novel mechanism for route discovery, called Relative Distance Micro-discovery (RDM). The concept behind RDM is that a query flood can be localized by knowing the relative distance (RD) between two terminals. To accomplish this, every time a route search between the two terminals is triggered, an iterative algorithm calculates an estimate of their RD, given an average nodal mobility and information about the elapsed time since they last communicated and their previous RD. Based on the newly calculated RD, the query flood is then localized to a limited region of the network centered at the source node of the route discovery and with maximum propagation radius that equals to the estimated relative distance. This ability to localize query flooding into a limited area of the network serves to minimize routing overhead and overall network congestion.

In RDMAR, calls are routed between the stations of the network by using routing tables which are stored at each station of the network; each node is treated as a host as well as a store-and-forward node. Each routing table lists all reachable destinations, wherein for each destination i, additional routing information is also maintained. This includes: the “Default Router” field that indicates the next hop node through which the current node can reach i, the “RD” field which shows an estimate of the relative distance (in hops) between the node and i, the ‖Time_Last_Update‖ (TLU)

field that indicates the time since the node last received routing information for i, a "RT_Timeout" field which records the remaining amount of time before the route is considered invalid, and a "Route Flag" field which declares whether the route to i is active.

RDMAR comprises of two main algorithms:

- Route Discovery – When an incoming call arrives at node i for destination node j and there is no route available, i initiates a route discovery phase. Here, i has two options; either to flood the network with a route query in which case the route query packets are broadcast into the whole network, or instead, to limit the discovery in a smaller region of the network, if some kind of location prediction model for j can be established. The former case is straightforward. In the latter case, the source of the route discovery, i, refers to its routing table in order to retrieve information on its previous relative distance with j and the time elapsed since i last received routing information for j. Let us designate this time as tmotion. Based on this information and assuming a moderate velocity, Micro_Velocity, and a moderate transmission range, Micro_Range, node i is then able to estimate its new relative distance to destination node j in terms of actual number of hops. To accomplish this, node i calculates the distance offset of DST (DST_Offset) during tmotion, and "adjusts" the result onto their previous relative distance (RDM_Radius).
- Route Maintenance – An intermediate node i, upon reception of a data packet, first processes the routing header and then forwards the packet to the next hop. In addition, node i sends an explicit message to examine whether a bi-directional link can be established with the previous node. RDMAR, therefore, does not assume bi-directional links but in contrast nodes exercise the possibility of having bi-directional links. In this way, nodes that forward a data packet will always have routing information to send the future acknowledgement back to the source. If node i is unable to forward the packet because there is no route available or a forwarding error occurs along the data path as a result of a link or node failure, i may attempt a number of additional re-transmissions of the same data packet, up to a maximum number of retries. However, if the failure persists, node *i* initiates a Route Discovery procedure.

3.3 Hybrid Routing Protocols

3.3.1 Zone Routing Protocol

Zone Routing Protocol (ZRP) [Haas 1998] is a hybrid example of reactive and proactive schemes. It limits the scope of the proactive procedure only to the node’s local neighborhood, while the search throughout the network, although it is global, can be performed efficiently by querying selected nodes in the network, as opposed to querying all the network nodes. In ZRP, a node proactively maintains routes to destinations within a local neighborhood, which is referred to as a routing zone and is defined as a collection of nodes whose minimum distance in hops from the node in question is no greater than a parameter referred to as zone radius. Each node maintains its zone radius and there is an overlap of neighboring zones.

The construction of a routing zone requires a node to first know who its neighbors are. A neighbor is defined as a node that can communicate directly with the node in question and is discovered through a MAC level Neighbor discovery protocol (NDP). The ZRP maintains routing zones through a proactive component called the Intrazone routing protocol (IARP) which is implemented as a modified distance vector scheme. On the other hand, the Interzone routing protocol (IERP) is responsible for acquiring routes to destinations that are located beyond the routing zone. The IERP uses a query-response mechanism to discover routes on demand. The IERP is distinguished from the standard flooding algorithm by exploiting the structure of the routing zone, through a process known as bordercasting. The ZRP provides this service through a component called Border resolution protocol (BRP).

The network layer triggers an IERP route query when a data packet is to be sent to a destination that does not lie within its routing zone. The source generates a route query packet, which is uniquely identified by a combination of the source node’s ID and request number. The query is then broadcast to all the source’s peripheral nodes. Upon receipt of a route query packet, a node adds its ID to the query. The sequence of recorded node Ids specifies an accumulated route from the source to the current routing zone.If the destination does not appear in the node’s routing zone, the node border casts the query to its peripheral nodes. If the destination is a member of the routing zone, a route reply is sent back to the source, along the path specified by reversing the accumulated route. A node will discard any route query packet for a query that it has previously encountered. An important feature of this route discovery process is that a single route query can return multiple route replies. The quality of these returned routes can be determined based on some metric. The best route can be selected based on the relative quality of the route.

3.3.2 Fisheye State Routing (FSR)

The Fisheye State Routing (FSR) protocol [Iwata 1999] introduces the notion of multi-level fisheye scope to reduce routing update overhead in large networks. Nodes exchange link state entries with their neighbors with a frequency which depends on distance to destination. From link state entries, nodes construct the topology map of the entire network and compute optimal routes. FSR tries to improve the scalability of a routing protocol by putting most effort into gathering data on the topology information that is most likely to be needed soon. Assuming that nearby changes to the network topology are those most likely to matter, FSR tries to focus its view of the network so that nearby changes are seen with the highest resolution in time and changes at distant nodes are observed with a lower resolution and less frequently. It is possible to think the FSR as blurring the sharp boundary defined in the network model used by ZRP.

3.3.3 Landmark Routing (LANMAR) for MANET with Group Mobility

Landmark Ad Hoc Routing (LANMAR) [Pei 2000] combines the features of FSR and Landmark routing. The key novelty is the use of landmarks for each set of nodes which move as a group (viz., a group of soldiers in a battlefield) in order to reduce routing update overhead. Like in FSR, nodes exchange link state only with their neighbors. Routes within Fisheye scope are accurate, while routes to remote groups of nodes are “summarized” by the corresponding landmarks. A packet directed to a remote destination initially aims at the Landmark; as it gets closer to destination it eventually switches to the accurate route provided by Fisheye. In the original wired landmark scheme [Tsuchiya 1988], the predefined hierarchical address of each node reflects its position within the hierarchy and helps find a route to it. Each node knows the routes to all the nodes within it hierarchical partition. Moreover, each node knows the routes to various “landmarks” at different hierarchical levels. Packet forwarding is consistent with the landmark hierarchy and the path is gradually refined from top-level hierarchy to lower levels as a packet approaches the destination.

LANMAR borrows from [Tsuchiya 1988] the notion of landmarks to keep track of logical subnets. A subnet consists of members which have a commonality of interests and are likely to move as a “group” (viz., soldiers in the battlefield, or a group of students from the same class). A “landmark” node is elected in each subnet. The routing scheme itself is modified version of FSR. The main difference is that the FSR routing table contains “all” nodes in the network, while the LANMAR routing table includes only the nodes within the scope and the landmark nodes. This feature greatly improves scalability by reducing routing table size and update traffic overhead. When a node needs to relay a packet, if the destination is within its neighbor scope, the address is found in the routing table and the packet is forwarded directly. Otherwise, the logical subnet field of the destination is searched and the packet is routed towards the landmark for that logical subnet. The packet however does not need to pass through the landmark. Rather, once the packet gets within the scope of the destination, it is routed to it directly.

The routing update exchange in LANMAR routing is similar to FSR. Each node periodically exchanges topology information with its immediate neighbors. In each update, the node sends entries within its fisheye scope. It also piggy-backs a distance vector with size equal to the number of logical subnets and thus landmark nodes. Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers.

3.4 Other Routing Protocols

There is plenty of routing protocol proposals for mobile ad hoc networks. Our discussion here is far from being exhaustive. Below we will describe some other routing protocols which employ different optimization criteria as the ones we have previously described.

3.4.1 Signal Stability Routing

Another on-demand protocol is the Signal Stability-Based Adaptive Routing protocol (SSR) [Dube 1997]. Unlike the algorithms described so far, SSR selects routes based on the signal strength (weak or strong) between nodes and a node’s location stability. The signal strengths of neighboring nodes are obtained by periodic beacons from the link layer of each neighboring node. This route selection criterion of SSR has the effect of choosing routes that have “stronger” connectivity [Chlamtac 1986].

3.4.2 Power Aware Routing

In this protocol, power-aware metrics [Singh 1998, Jin 2000] are used for determining routes in wireless ad hoc networks. It has been shown that using these metrics in a shortest-cost routing algorithm reduces the cost/packet of routing packets by 5 - 30 percent over shortest-hop routing (this cost reduction is on top of a 40-70 percent reduction in energy consumption over the MAC layer protocol used). Furthermore, using these new metrics ensures that mean time to node failure is increased significantly, but packet delays do not increase. A recent work [Lee 2000a] concentrates on selecting a route based the traffic and congestion characteristics in the network.

3.4.3 Associativity-Based Routing

This is a totally different approach in mobile routing. The Associativity-Based Routing (ABR) [Toh 1997] protocol is free from loops, deadlock, and packet duplicates, and defines a new routing metric for ad hoc mobile networks. In ABR, a route is selected based on a metric that is known as the degree of association stability. Each node periodically generates a beacon to signify its existence. When received by neighboring nodes, this beacon causes their associability tables to be updated. For each beacon received, the associatively tick of the current node with respect to the beaconing node is incremented. Association stability is defined by connection stability of one node with respect to another node over time and space. A high (low) degree of association stability may indicate a low (high) state of node mobility. Associability ticks are reset when the neighbors of a node or the node itself move out of proximity. A fundamental objective of ABR is to derive longer-lived routes for ad hoc networks. The three phases of ABR are:

- Route discovery,
- Route reconstruction (RRC),
- Route deletion.

The route discovery phase is accomplished by a broadcast query and await-reply

(BQ-REPLY) cycle. A node desiring a route broadcasts a BQ message in search of mobiles that have a route to the destination. All nodes receiving the query (that are not the destination) append their addresses and their associativity ticks with their neighbors along with QoS information to the query packet. A successor node erases its upstream node neighbors' associativity tick entries and retains only the entry concerned with itself and its upstream node. In this way, each resultant packet arriving at the destination contains the associativity ticks of the nodes along the route to the destination. The destination is then able to select the best route by examining the associativity ticks along each of the paths. When multiple paths have the same overall degree of association stability, the route with the minimum number of hops is selected. The destination then sends a REPLY packet back to the source along this path. Nodes propagating the REPLY mark their routes as valid. All other routes remain inactive, and the possibility of duplicate packets arriving at the destination is avoided.

RRC may consist of partial route discovery, invalid route erasure, valid route updates, and new route discovery, depending on which node(s) along the route move. Movement by the source results in a new BQ-REPLY process. The RN message is a route notification used to erase the route entries associated with downstream nodes. When the destination moves, the immediate upstream node erases its route and determines if the node is still reachable by a localized query (LQ[H]) process, where H refers to the hop count from the upstream node to the destination. If the destination receives the LQ packet, it REPLYs with the best partial route; otherwise, the initiating node times out and the process backtracks to the next upstream node. Here, a RN [0] message is sent to the next upstream node to erase the invalid route and inform this node that it should invoke the LQ[H] process. If this process results in

backtracking more than halfway to the source, the LQ process is discontinued and a new BQ process is initiated at the source.

3.5 Comparison Table

Table 2 summarizes the main characteristics of the most cited protocols discussed so far.

Table 2 – Protocol characteristics

Routing Protocol	**Route Acquisition**	**Flood for Route Discovery**	**Delay for Route Discovery**	**Multipath Capability**	**Upon Route Failure**
DSDV	Computed a priori	No	No	No	Floods route updates throughout the network
WRP	Computed a priori	No	No	No	Ultimately, updates the routing tables of all nodes by exchanging MRL between neighbors
DSR	On-demand, only when needed	Yes. Aggressive use of caching often reduces flood scope	Yes	Not explicitly. The technique of salvaging may quickly restore a Route	Route error propagated up to the source to erase invalid path
AODV	On-demand, only when needed	Yes. Conservative use of cache to reduce flood scope	Yes	No, although recent research indicate viability	Route error broadcasted to erase invalid path
TORA	On-demand, only when needed	Usually, only one flood for initial DAG construction	Yes. Once the DAG is constructed, multiple paths are found	Yes	Error is recovered locally, and only when alternative routes are not available
LAR	On-demand, only when needed	Localized flood by using location information	Yes	No	Route error propagated up to the source
ZRP	Hybrid	Only outside a source’s zone	Only if the destination is outside the source’s zone	No	Hybrid of updating nodes’ tables within a zone and propagating route error to the source

Route
RREQ
RREP
MACT
Group member
Tree member
Non-member

4. Multicast Routing Protocols

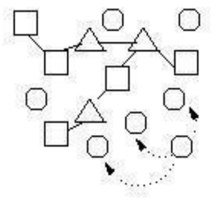
Multicasting is the process of sending packets from a transmitter to multiple destinations identified by a single address. As with their wired counterparts, multicasting in MANET is also a hard task to accomplish, and it is even harder in the MANET case since the physical topology changes quite frequently. Therefore, multicast protocols designed for a MANET have to take topological changes into consideration. In this section, we discuss two multicast routing protocols, namely AODV Multicasting and ODMRP, proposed within the MANET [MANET] working group at the IETF [IETF].

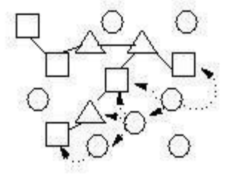
4.1 AODV Multicasting (MAODV)

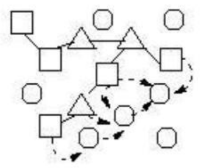
The AODV multicast algorithm [Royer 1999] uses similar RREQ and RREP messages as in unicast operation. The nodes join the multicast group on-demand, and a multicast tree is created in the process. The tree consists of the group members and nodes connected to the group members. This enables a recipient host to join a multicast group even if it is more than one hop away from a multicast group member. The unicast operation of the protocol also benefits from the information that is gathered while discovering routes for multicast traffic. This cuts down the signaling traffic in the network.

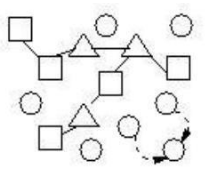
4.1.1 Route Discovery

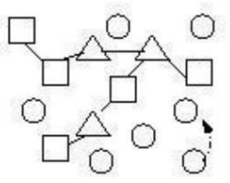
When a node wishes to find a route for a multicast group, it sends an RREQ message. The destination address in the RREQ message is set to the address of the multicast group. If the node wants to join the group in question, the J_flag in the message is set. Any node may respond to a RREQ merely looking for a route, but only a router in the desired multicast tree may respond to a join RREQ. The corresponding RREP message may travel through nodes that are not members of the multicast group. This means that the eventual route may also include hops through non-member nodes (see Figure 9).

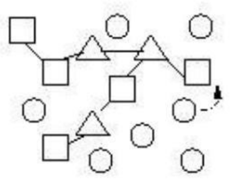

(a) A node sends RREQ to join multicast group


(b) Neighboring nodes rebroadcast the request


(c) Tree members and Group members send RREPs


(d) Duplicate RREPs are dropped


(e) MACT is sent to the node that has the shortest path


(f) MACT is forwarded to a group member

The multicast RREP message is slightly different from the unicast RREP. The address of the multicast group leader is stored in a field called Group_Leader_Addr. In addition, there is a field called Mgroup_hop. This field is initialized to zero and it is incremented at each hop along the route. Mgroup_hop contains the distance in hops of the source node to the nearest member of the multicast tree.

Because the protocol relies on a group-wide destination sequence number (DSN) to ensure fresh routes, the group leader broadcasts periodical Group Hello messages. The Group Hello is an unsolicited RREP message that has a TTL greater than the diameter of the network. The message contains extensions that indicate the multicast group addresses and corresponding sequence numbers of all the groups for which the node is the group leader. The sequence number for each group is incremented each time the Group Hello is broadcast. The Hop_Cnt field in the message is initialized as zero and incremented by the intermediate nodes.

The nodes receiving the Group Hello use the information contained therein to update their request tables. If a node does not have an entry for the advertised multicast group, one is inserted. The hop counts are used to determine the current distance from the group leader.

4.1.2 Multicast Tree Maintenance

In a network consisting of mobile nodes, link breakages are bound to happen. The breakages should be repaired promptly to ensure maximal connectivity of the multicast group. Multicast tree maintenance has three different scenarios: activating a link when a new node joins the group, pruning the tree when a node leaves the group, and repairing a broken link. Repairing consists of re-establishing the branches when a link goes down and reconnecting the tree after a possible partition in the network.

Route activation. If a node receives more than one RREP to a RREQ it has sent, it must pick only one RREP as the next hop. This avoids adding any extra branches to the tree and loops are eliminated. The source node waits for a specified Route Discovery timeout after it has sent an RREQ, and then selected the received route that has the greatest DSN and the fewest hops to the nearest member of the multicast tree. The node then unicasts a multicast-specific message called Multicast Activation (MACT) to the selected next hop. The MACT message carries the source address, SSN, destination address and flags P_flag and GL_flag. P_flag is used in case of pruning and GL_flag is set when selecting the multicast group leader, as explained later.

When a group member receives a MACT message, it enables the entry for the source node in its own multicast routing table. If a node that is not a group member receives a MACT message, it acts in a similar way as the source node. A new MACT is sent to the best next hop towards the multicast group. Those nodes that have generated or forwarded RREP messages delete the corresponding route entries from their routing tables if they do not receive a MACT within a specified Multicast Tree Build timeout period.

The multicast tree can never have multiple paths to any tree node because the MACT messages are only propagated through one path. Hence, the tree is indeed a tree. Because the nodes forward data only along the activated routes in their routing tables, the data can never be forwarded to the source node by multiple intermediate nodes.

Pruning. Multicast group members sometimes remove themselves from the group. If a non-leaf node decides to leave the group it must continue as a router for the multicast tree. Leaf nodes may prune themselves by sending a MACT message with the P_flag (prune) set. The Dest_addr of the message points to the multicast group in question.

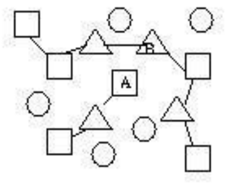
Because a leaf node can only have one next hop node for the multicast group, the MACT can be unicasted to that node. After the MACT has been sent, information about the group can be removed from the route table. When the recipient of the MACT notices the P_flag, it deletes the entry for the sender of the MACT from its own route table. If the recipient is not a member of the multicast group and it would become a leaf node after this operation, it can prune itself from the multicast tree using the same method. The pruning process terminates when it reaches either a multicast group member or a non-leaf node.

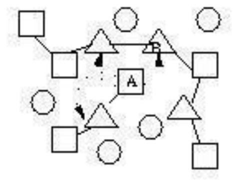
4.1.3 Triggered Repair of Broken Links

Timeouts and node mobility cause links to break within the multicast tree. In multicast operation, routes must be reconstructed when a link goes down since current group members must stay connected.

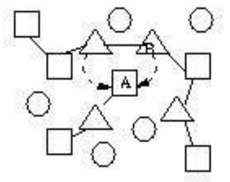
If a node does not receive Group Hello messages from a neighbor within an acceptable timeout period, the link between the two nodes is considered broken (see Figure 10). The node that is further away from the multicast group leader tries to rejoin the group as explained earlier. The RREQ carries a Multicast Group Leader Extension with a Hop Count that equals the distance to the group leader. However, the RREQ message is first broadcast with a restricted TTL value (two more than the recorded Multicast Group Hop Count). Because only one node starts the repair process, loops and duplicate work can be avoided. The use of a small TTL is an effort to keep the effects of the link breakage local. Only if no RREP is received within a timeout period, the RREQ is rebroadcast over the whole network.

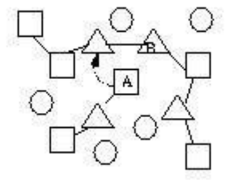
A node that is nearer to the group leader than the RREQ hop count indicates is allowed to respond to the request. Duplicate RREPs are discarded. When a RREP has been received, the route must be activated. If the node was not a leaf node, it must also inform the downstream nodes of possible changes in distance to the group leader. In this case, a MACT message with an U_flag (Update) is broadcast. Those nodes that are not downstream neighbors of the sender ignore the update messages.



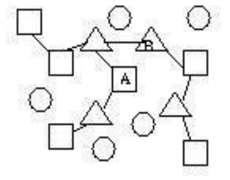


(a) A stops receiving Group Hello messages from B (b) A sends a RREQ with Group Leader extension





(c) Tree members and Group members send RREPs (d) A RREP that has the smallest hop count is selected



(e) MACT is sent to the sender of the chosen RREP

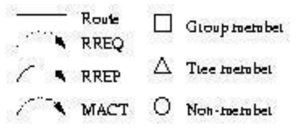
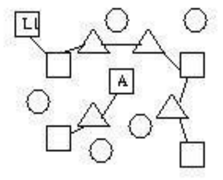


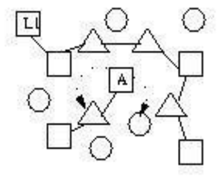
Figure 10 – AODV multicast tree repair

If the node that started the repair process never gets a RREP after a set number of retries, a partition is assumed to be present in the network (see Figure 11). In this case a new group leader must be elected. If the said node was a part of the multicast group, it becomes the new group leader. Otherwise it sends a pruning MACT message to its next hop, if there is only one of them. When the next hop node receives the MACT with P_flag set, it has the same options as the originating node. This is repeated until a new group leader is found.

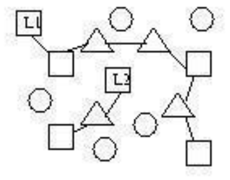
If there is more than one next hop downstream, a node cannot prune itself. Instead it sends a MACT with GL_flag (Group Leader) set to the first of its next hops. This process is repeated until a multicast group member receives the MACT. This node then becomes the new group leader and it broadcasts a Group Hello message with U_flag set. The nodes that receive the Hello then update their route tables accordingly. Meanwhile, if the node upstream of the breakage became a non-member leaf node, it waits for a MACT from the downstream node. If it is not received within a set timeout period, the node prunes itself from the tree.

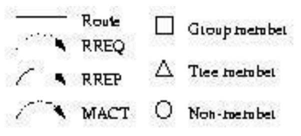
Reconnecting two partitioned trees. After a network partition, there are two group leaders (see Figure 12). A node can detect reconnection of the partitions if it receives a Group Hello with conflicting information about the group leader and hop count. The group leader that has the lower IP address (L2) initiates the repair process by sending a RREQ with R_flag (Repair) and J_flag set to the other group leader with the higher IP address (L1). The RREQ is sent via the next hop from which the Hello message was received. The current DSN for L2 is included in the message. The RREQ is only propagated upstream towards L1 so that no loops can appear. When L1 receives the RREQ, it creates a new DSN that is higher than its own or the one carried by the RREQ. It sends back a RREP message with the R_flag set and becomes the new leader.





(a) A stops receiving Group Hello messages from L (b) A broadcasts a RREQ with Group Leader extension





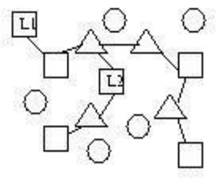
(c) After a set number of retries No RREP is received. A becomes a new Group Leader (L2)

Figure 11 – Detecting a network partition

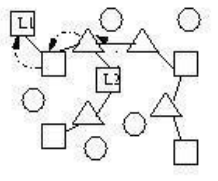
The nodes that used to belong to the group lead by L2 propagate the RREP towards L2. The node where the RREP was received from is marked as the next hop upstream and the next hop towards L2 is marked as being downstream. When L2 finally receives the message, it acknowledges L1 as the new leader, thus completing the reconnection of the tree. The next time that L1 sends a Hello message it sets the U_flag so that all the nodes will update their routing information.

4.2 On-Demand Multicast Routing Protocol (ODMRP)

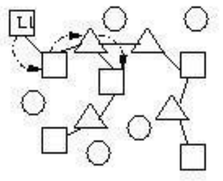
ODMRP (On-Demand Multicast Routing Protocol) [Bae 2000] is a mesh-based, instead of tree-based, multicast protocol that provides richer connectivity among multicast members. By building a mesh and supplying multiple routes, multicast packets can be delivered to destinations in the face of node movements and topology changes. In addition, the drawbacks of multicast trees in mobile wireless networks (e.g., intermittent connectivity, frequent tree reconfiguration, traffic concentration, etc.) are avoided. To establish a mesh for each multicast group, ODMRP uses the concept of forwarding group [Chiang 1998]. The forwarding group is a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs. ODMRP also applies on-demand routing techniques to avoid channel overhead and improve scalability. No explicit control message is required to leave a group.

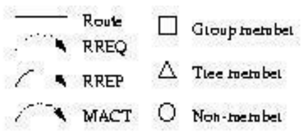


(a) L2 starts receiving Group Hello messages That
point to L1 as the leader



(b) L2 unicasts a RREQ to L1 with *J_Flag* and *R_Flag*
Set





(c) L1 becomes the leader of the reconnected tree

Figure 12 – Reconnecting partitioned subtrees

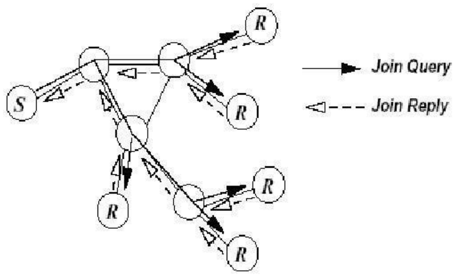
4.2.1 Multicast Route and Mesh Creation

In ODMRP, group membership and multicast routes are established and updated by the source on demand. Similar to on-demand unicast routing protocols, a request phase and a reply phase constitute the protocol (see Figure 13). While a multicast source has packets to send, it periodically broadcasts to the entire network a member advertising packet, called a JOIN QUERY, which refreshes the membership information and updates the route as follows. When a node receives a non-duplicate JOIN QUERY, it stores the upstream node ID (i.e., backward learning) and rebroadcasts the packet. When the JOIN QUERY packet reaches a multicast receiver, the receiver creates or updates the source entry in its Member Table. While valid entries exist in the Member Table, JOIN REPLIES are broadcasted periodically to the neighbors. When a node receives a JOIN QUERY, it checks if the next node ID of one of the entries matches its own ID. If it does, the node realizes that it is on the path to the source and this is part of the forwarding group. It then sets the flag FG_Flag and broadcasts its own JOIN RELY built upon matched entries. The JOIN REPLY is this propagated by each forwarding group member until it reaches the multicast source via the shortest path. This process constructs (or updates) the routes from sources to receivers and builds a mesh of nodes, the forwarding group.

Figure 14 visualizes the forwarding group concept. The forwarding group is a set of nodes in charge of forwarding multicast packets. It supports shortest paths between any member pairs. All nodes inside the bubble (multicast members and forwarding group nodes) forward multicast data packets. Note that a multicast receiver can also be a forwarding group node if it is on the path between a multicast source and another receiver. The mesh provides richer connectivity among multicast members compared to trees. Flooding redundancy among forwarding group helps overcome node displacements and channel fading. Hence, unlike trees, frequent reconfigurations are not required.

Figure 15 is an example to show the robustness of a mesh configuration. Three sources (S1, S2, and S3) send multicast data packets to three receivers (R1, R2, and R3) via three forwarding group nodes (A, B, and C). Suppose the route from S1 to R2 is < S1-A-B- R2>. In a tree configuration, if the link between nodes A and B breaks or fails, R2 cannot receive any packets from S1 until the tree is reconfigured. ODMRP, on the other hand, already has a redundant route <S1-A-C-B-R2> to deliver packets without going through the broken link between nodes A and B.

After group establishment and route construction process, a multicast source can transmit packets to receivers via selected routes and forwarding groups. Periodic control packets are sent only when outgoing data packets are still present. When receiving a multicast data packet, a node forwards it only if it is not a duplicate and the setting of the FG_Flag for the multicast group has not expired. This procedure minimizes traffic overhead and prevents sending packets through stale routes.



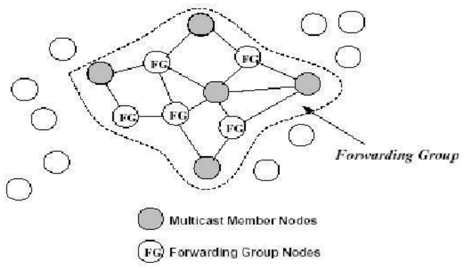


Figure 13 – On-demand procedure
for membership
setup and maintenance

Figure 14 – The forwarding group concept

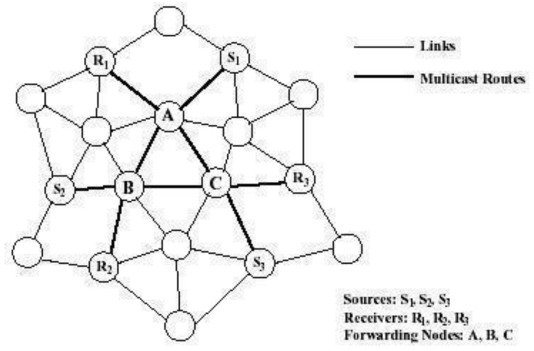

R_1
S_1
A
S_2
B
C
R_3
R_2
S_3
Links
Multicast Routes
Sources: S_1, S_2, S_3
Receivers: R_1, R_2, R_3
Forwarding Nodes: A, B, C

Figure 15 – The robustness of a mesh organization

4.3 Other Protocols

There are many more protocols with different characteristics for multicast in a MANET. They include: AMRoute [Bommaiah 1998], AMRIS [Wu 1998], CAMP [Garcia-Luna-Aceves 1999a], and flooding. A comparison of these protocols is performed in [Lee 2000b].

5. Medium Access Control (MAC) Protocols Issues

Maybe not as much as ad hoc routing protocols, but MAC protocols have also been receiving attention from the research community. There are still many issues that need to be addressed in order to design an efficient MAC protocol to be used in a wireless ad hoc network environment [Royer 2000]. There are several MAC protocols which can be employed for multi-hop ad hoc networking including IEEE 802.11 [Crow 1997], Bluetooth [Bluetooth] and HiperLAN [HiperLAN 1995]. Usually, the IEEE 802.11 standard is the platform employed to experiment multi-hop networking. However, it does not support multi-hop as is. In this section, we discuss some fundamental issues MAC protocols for wireless multi-hop ad hoc networks have to cope up with, along with their proposed solutions.

5.1 Hidden Terminal Problem

In CSMA, every station senses the carrier before transmitting, and if it detects carrier then the transmission is deferred. Carrier sense attempts to avoid collisions by testing the signal strength in the vicinity of the transmitter. However, collisions occur at the receiver, not the transmitter; i.e., it is the presence of two or more interfering signals at the receiver that constitutes a collision. Since the receiver and the sender are typically not co-located, carrier sense does not provide the appropriate information for collision avoidance. Two examples illustrate this point in more detail. Consider the configuration depicted in Figure 16. Station A can hear B but not C, and station C can hear station B but not A (and, by symmetry, we know that station B can hear both A and C). First, assume A is sending to B. When C is ready to transmit (perhaps to B or perhaps to some other station), it does not detect carrier and thus commences transmission; this produces a collision at B. Station C's carrier sense did not provide the necessary information since station A was "hidden" from it. This is the classic "hidden terminal" scenario.

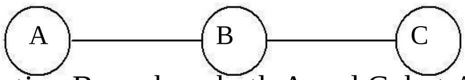


Figure 16 – Station B can hear both A and C, but A and C cannot hear each other.

An “exposed” terminal scenario results if now we assume that B is sending to A rather than A sending to B. Then, when C is ready to transmit, it does detect carrier and therefore defers transmission. However, there is no reason to defer transmission to a station other than B since station A is out of C’s range. Station C’s carrier sense did not provide the necessary information since it was exposed to station B even though it would not collide or interfere with B’s transmission. The point to note here is that carrier sense provides information about potential collisions at the sender, but not at the receiver. This information can be misleading when the configuration is distributed so that not all stations are within range of each other.

The solution to the hidden terminal problem was proposed in [Karn 1990]. It consists of transmitting RTS (Request-to-Send) and CTS (Clear-to-Send) packets between nodes that wish to communicate. Among other things, these RTS and CTS packets carry the duration of the data transfer of the communicating parties. Stations in the neighborhood that do not participate in the communication but overhear either the RTS or CTS keep quiet for the duration of the transfer. Returning to our example of Figure 16, when node A wants to send a packet to node B, node A first sends a RTS packet to B. On receiving the RTS packet, node B responds by sending a CTS packet (provided node A is able to receive the packet). As a result of that, when node C overhears the CTS sent by B it keeps quiet for the duration of the transfer contained in the CTS packet. As for the exposed terminal problem, while in IEEE 802.11 MAC layer there is almost no scheme to deal with it, MACAW [Bharghavan 1994] solves this problem by having the source transmit a data sending (DS) control packet to alert exposed nodes of the impending arrival of an ACK packet.

5.2 Reliability

Wireless links are prone to errors. Actually, packet error rates of wireless mediums are much higher than that of their wired counterparts. As a result, some protocols – which were originally designed to work in wired world – suffer performance degradation when operating in a wireless environment. A classic example of this problem is TCP (which has been designed and fine-tuned for wired networks) that assumes transmission timer expiration as an indication of network congestion. This event triggers the execution of TCP congestion control mechanisms which ultimately decreases the transmission rate aiming at reducing the network congestion. As a matter of fact, this is often true in wired environments since wired media are usually very reliable. However, in wireless environment this is often not the case. Due to effects such as multipath fading, interference, shadowing, distance between transmitter and receiver, etc., packet losses occur every now and then. As a result, when a packet loss takes place in a communication using TCP, it erroneously assumes that the loss was due to congestion and enters its congestion control mechanisms. There have been some proposals to cope up with this TCP behavior in a MANET [Holland 1999, Liu 2001, Chandran 2001].

As for the MAC protocol issues, a common approach to reduce packet loss rates experienced by upper layers is to introduce acknowledgment (ACK) packets. Returning to our earlier example of Figure 16, whenever node B received a packet from node A, node B sends an ACK packet to A. In case node A fails to receive the ACK from B, it will retransmit the packet. This approached is adopted in many protocols [Bharghavan 1994]. As an example, the IEEE 802.11 DCF (Distributed Coordination Function) [Crow 1997] uses RTS-CTS to avoid the hidden terminal problem and ACK to achieve reliability.

5.3 Collision Avoidance

The radios used in the wireless mobile nodes employed in wireless communications are half-duplex. This is to say that these radios are not able to transmit and receive at the same and, thus, collision detection is not possible. To minimize collisions, wireless MAC protocols, such as CSMA/CA, often use collision avoidance techniques in conjunction with a carrier sense (be it physical or virtual) mechanism. Carrier sense is the mechanism whereby nodes wishing to transmit data first have to check whether the medium is idle or busy. The idea is that a station cannot transmit until the channel is idle. Collision avoidance is implemented by mandating that, when the channel is sensed idle, the node has to wait for a randomly chosen duration before attempting to transmit. This mechanism drastically decreases the probability that more that one node attempts to transmit at the same time, hence, avoiding collision. Of course there will be cases where more than one node initiate their transmission at the same time. In these cases, transmissions are corrupted and the corresponding nodes retry later on.

5.4 Congestion Avoidance

Congestion avoidance is another aspect of fundamental importance in wireless MAC protocols. In IEEE 802.11 DCF, when a node detects the medium to be idle, it chooses a backoff interval between [0, CW], where CW is called contention window which usually has a minimum (CW_min) and maximum value (CW_max). The idea is that the node will count down the backoff interval and when it reaches zero the node can transmit the RTS. In case the medium becomes busy while the node is still counting down the backoff interval, the countdown process is suspended.

To illustrate how DCF works, let us consider the example in Figure 17. In this figure, BO1 and BO2 are the backoff intervals of nodes 1 and 2, and we assume for this example that CW = 31. As we can see from Figure 17, node 1 and node 2 have chosen a backoff interval of 25 and 20, respectively. Obviously, node 2 will reach zero before five units of time earlier than node 1. When this happens, node 1 will notice that the medium became busy and freezes its backoff interval currently at 5. As soon as the medium becomes idle again, node 1 resumes its backoff countdown and transmits its data once the backoff interval reaches zero. Similarly, upon node's 1 transmission, node 2 will freeze its backoff countdown process and resume it as soon as node 1 finishes its transmission. To a certain extent, collision can be avoided by carrying out this procedure provided we choose a suitable value for the CW parameter. Choosing a large CW leads to large backoff intervals and can result in larger overhead, since nodes have to carry out the countdown procedure. On the other hand, choosing a small CW leads to a larger number of collisions, that is, it is more likely that two nodes will count to zero simultaneously.

5.5 Congestion Control

As we have mentioned earlier, the number of nodes attempting to transmit simultaneously may change with time. Therefore, some mechanism to manage congestion is needed. In IEEE 802.11 DCF, congestion control is achieved by dynamically choosing the contention window CW. When a node fails to receive a CTS in response to its RTS, it assumes that congestion has built up and, as a consequence, doubles its contention window up to CW_max. When a node successfully completes its transmission, it restores its contention window to CW_min. This mechanism of dynamically controlling the contention window is called Binary Exponential Backoff, since the contention window increases exponentially with a failed CTS.

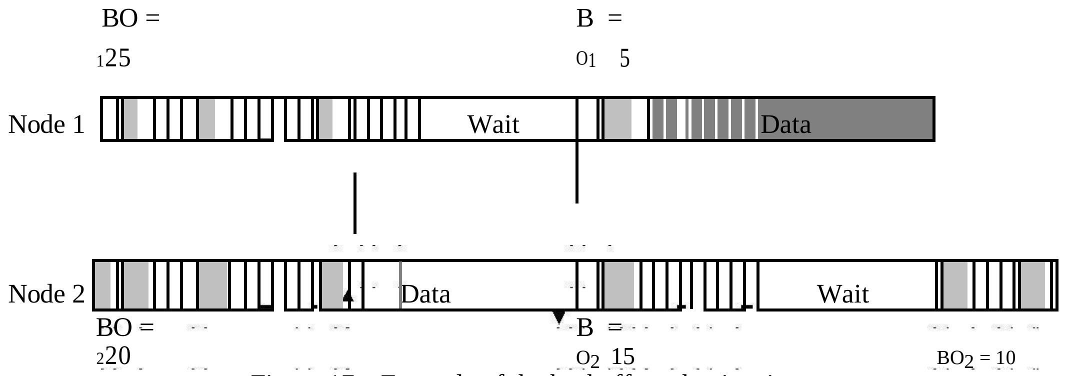


Figure 17 – Example of the backoff mechanism in DCF

5.6 Energy Efficiency

Since many mobile hosts are operated by batteries, there is an increasing interest for MAC protocols that conserve energy. The current proposals in this area usually suggest turning the radio off when it is not needed. IEEE 802.11 has a Power Saving (PS) mode whereby the Access Point (AP) periodically transmits a beacon indicating which nodes have packets waiting for them. Each PS node wakes up periodically to receive the beacon transmitted by the AP. In case a node has a packet waiting for it, it sends a PS-POLL packet to the AP after waiting for a backoff interval in [0, CW_min]. Upon receipt of the PS-POLL packet, the AP transmits the data to the requesting node. Using this procedure, it is possible to extend the battery life of mobile nodes for a longer period of time.

5.7 Other MAC Issues

Similarly to ad hoc routing protocols discussed earlier, our discussion on MAC protocol issues is also far from being exhaustive. There are many other issues to be considered such as fairness. Fairness has many meanings and one of them might say that stations should receive equal bandwidth. With the basic approach of IEEE 802.11, this fairness is not easy to accomplish since unfairness will eventually occur when one node backs off much more than some other node. MACAW's solution to this problem [Bharghavan 1994] is to append the contention window value (CW) to packets a node transmits, so that all nodes hearing that CW use it for their future transmissions. Since CW is an indication of the level of congestion in the vicinity of a specific receiver node, MACAW proposes maintaining a CW independently for each receiver. There are also other proposals such as Distributed Fair Scheduling [Vaidya 2000] and Balanced MAC [Ozugur 1998].

A final comment must be made on receiver-related issues in wireless MAC protocols. All protocols discussed so far are sender-initiated protocols. In other words, a sender always initiates a packet transfer to a receiver. The receiver might take a more active role in the process by assisting the transmitter in certain issues such as collision avoidance [Garcia-Luna-Aceves 1999b], and some sort of adaptive rate control [Holland 2001].

6. Wireless Sensor Networks

Wireless Sensor Networks (WSN) [Estrin 1999, Kahn 1999] are a recent application of ad hoc networks that is expected to find increasing deployment in coming years, as they enable reliable monitoring and analysis of unknown and untested environments. These networks are "data centric" i.e., unlike traditional networks where data is requested from a specific node, data is requested based on certain attributes such as, “which area has temperature 100ºF”. The routing protocols proposed for all the traditional networks are point-to-point and so these protocols are not well suited for wireless sensor networks.

Sensor networks consist of thousands of tiny disposable, low power devices equipped with programmable computing, multiple sensing and communication capability. They operate and respond in a very dynamic environment and their design must be application specific. Judging by the interest shown by the military, academia, and the media, innumerable applications exist for sensor networks. Examples include weather monitoring, security and tactical surveillance, distributed computing, fault detection and diagnosis in machinery, detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects, etc. One example of a wireless sensor network is illustrated in Figure 18, with sensor nodes being deployed from a low-flying airplane. The example in this figure is the most quoted one, whereby a large number of sensors are dropped on the enemy’s territory from an airplane so that activities on the ground can be detected and communicated.

A sensor node is basically a device that converts a sensed attribute (such as temperature, vibrations) into a form understandable by the users. Each of such devices may include a sensing module, a communicating module (display or a media to transmit data to the user), memory (to hold data till it can be used) and a power supply for the sensor.Wired sensor networks have been used for years for a number of applications. Some examples include distribution of thousands of sensors and wires over strategic locations in a structure such as an automobile or an airplane, so that conditions can be constantly monitored both from the inside and the outside and a real-time warning can be issued whenever a major problem is forthcoming in the monitored structure. These wired sensors are made large (and expensive) to cover as much area as desirable. Each of these has a continuous power supply and communicate their data to the end-user using a wired network. The organization of such a network should be pre-planned to find strategic position to place these nodes and then should be installed appropriately. The failure of a single node might bring down the whole network or leave that region completely un-monitored. These networks are usually unattended and some degree of fault-tolerance needs to be incorporated so that maintenance is minimized. This is especially desirable in those applications where the sensors may be embedded in the structure or places in an inhospitable terrain and are inaccessible for any service.

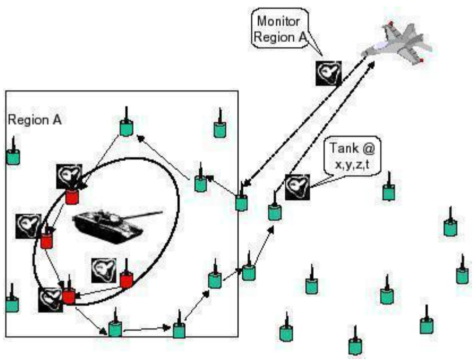


Figure 18 – A wireless sensor network

The advancement in technology has made it possible to have extremely small, low powered devices equipped with programmable computing, multiple parameter sensing and wireless communication capability. Also, the low cost of sensors makes it possible to have a network of hundreds or thousands of these wireless sensors, thereby enhancing the reliability, accuracy of data and the area coverage. Moreover, it is necessary that the sensors be easy to deploy (i.e., require very low or no installation cost etc). In short, the advantages of wireless sensor networks over wired ones are as follows:

- Ease of deployment – These wireless sensors can be deployed (dropped from plane or placed in factory) at the site of interest without any pre-organization, thus reducing the installation cost and increasing the flexibility of arrangement.

- Extended range – One huge wired sensor (Macro-sensor) can be replaced by many smaller wireless sensors for the same cost. One macro-sensor can sense only a limited region whereas a network of smaller sensors can be distributed over a wider region.

- Fault tolerant – Since sensor networks are mostly unattended, they should possess fault tolerant capability. With macro-sensors, the failure of one node makes that area completely unmonitored till it is replaced. Whereas with wireless sensors, failure of one node does not affect the network operation considerably as there are other nodes collecting similar data. At most, the accuracy of data collected may be reduced.

- Mobility – Since these wireless sensors are equipped with battery, they can possess limited mobility (e.g., robots). Thus, if a region becomes unmonitored we can have the nodes rearrange themselves to distribute evenly, i.e., these nodes can be made to move towards area of interest. Note, however, that these nodes have lower mobility as compared to ad hoc networks.

There are a few inherent limitations of wireless medium such as low bandwidth, error prone transmissions, need for collision free channel access, etc. It is clear due to the nature of observed phenomenon, the required bandwidth of sensor data will be low, on the order of 1-100 kb/s. Since the wireless nodes are mostly mobile and are not connected in any way to a constant power supply, they derive energy from a personal battery. This limits the amount of energy available to the nodes. In addition, since these sensor nodes are deployed in places where it is difficult to either replace the nodes or their batteries, it is desirable to increase the longevity of the network and, preferably, all the nodes should die together so that we can replace all the nodes simultaneously or put new nodes on the whole area. Finding individual dead nodes and then replacing those nodes selectively would require pre-planned deployment and eliminates some advantages of these networks. Thus, the protocols designed for these networks must strategically distribute the dissipation of energy, which also increases the average life of the overall system. In addition, environments in which these nodes operate and respond are very dynamic, with fast changing physical parameters.

Traditional routing protocols defined for wireless ad hoc networks are not well suited for wireless sensor networks due to the following reasons:

- Sensor networks are "data centric", i.e., unlike traditional networks where data is requested from a specific node, data is requested based on certain attributes such as "which area has temperature 50ºF".
- In traditional wired and wireless networks, each node is given a unique id, used for routing. This cannot be effectively used in sensor networks because these networks, being data centric, routing to and from specific nodes is not required.
- Adjacent nodes may have similar data. So, rather than sending data separately from each node to the requesting node, it is desirable to aggregate similar data before sending it.
- The requirements of the network change with the application and hence, it is application-specific. For example, in some applications the sensor nodes are fixed and not mobile while others may need data based only on one selected attribute (viz., attribute is fixed in this network).

Thus, sensor networks need protocols which are application specific, data centric, capable of aggregating data and minimizing energy consumption. An ideal sensor network should have the following additional features:

- Attribute based addressing is typically employed in sensor networks. The attribute-based addresses are composed of a series of attribute-value pairs which specify certain physical parameters to be sensed. For example, an attribute address may be (temperature > 40ºF, location = "Rio de Janeiro"). So, all nodes located in "Rio de Janeiro" which sense a temperature greater than 40ºF should respond.

- Location awareness is another important issue. Since most data collection is based on location, it is desirable that the nodes know their position whenever needed.

Another important requirement in some cases is that the sensors should react immediately to drastic changes in their environment, for example, in time-critical applications. The end user should be made aware of any drastic deviation in the situation with minimum delay, while making efficient use of the limited wireless channel bandwidth and sensor energy.

- Query Handling is another additional feature. Users, using hand held wireless devices, should be able to request data from the network. Since these hand-held devices are also energy constrained, the user should be able to query through the base station or through any of the nodes, whichever is closer. So, there should be a reliable mechanism to transmit the query to appropriate nodes which can respond to the query. The answer should then be re-routed back to the user as quickly as possible. Since efficient query handling is a highly desirable feature, we explore it in the next section.

In wireless sensor networks where efficient usage of energy is very critical, longer latency for non - critical data is preferable for longer node lifetime. However, queries for time critical data should not be delayed and should be handled immediately.Some protocols try to use the energy of the network very efficiently by reducing unnecessary data transmission for non-critical data but transmitting time-critical data immediately even if we have to keep the sensors on at all times. Periodic data is transmitted at longer intervals so that historical queries can be answered. All other data is retrieved from the system on-demand.

6.1 DARPA Efforts Towards Wireless Sensor Networks

The Defense Advanced Research Projects Agency (DARPA) has identified networked micro sensors technology as a key application for the future. There are many interesting projects and experiments going under the DARPA SensIT (Sensor Information Technology) program [SensIT]. The SensIT program aims to develop the software for distributed micro-sensors. On the battlefield of the future, a networked system of smart, inexpensive and plentiful microsensors, combining multiple sensor types, embedded processors, positioning ability and wireless communication, will pervade the environment and provide commanders and soldiers alike with heightened situation awareness. Therefore, software is needed to enable a variety of sensor nets, on the ground and in the air as well as on buildings and bodies, all functioning autonomously, operating with high reliability, and processing signals and information collaboratively in the network to provide useful information to the warfighter in a timely manner. In this text, we will focus on routing and MAC protocols targeted at supporting these wireless sensor networks.

6.2 Classification of Sensor Networks

Looking at the various ways in which one can employ the network resources, sensor networks can be classified on the basis of their mode of operation or functionality, and the type of target applications. Accordingly, sensor networks are classified into two types:

- **Proactive Networks** – The nodes in this network periodically switch on their sensors and transmitters, sense the environment and transmit the data of interest. Thus, they provide a snapshot of the relevant parameters at regular intervals and are well suited for applications requiring periodic data monitoring.

- **Reactive Networks** – In this scheme the nodes react immediately to sudden and drastic changes in the value of a sensed attribute. As such, these are well suited for time critical applications.

Once the type of network is decided, protocols that efficiently route data from the nodes to the users have to be designed, preferably using a suitable MAC sub-layer protocol to avoid collisions. Attempts should be made to distribute energy dissipation evenly among all nodes in the network as we do not have specialized high energy nodes in the network.

There are some basic functionalities and characteristics expected from a protocol for proactive networks. To illustrate this fact, let us take as an example a hierarchical clustering scheme whereby a group of nodes, called cluster members, synchronize and elect one of its members as the cluster-head (see Figure 19). At each cluster change time, once the cluster-heads are decided, the cluster-head broadcasts the following parameters:

- Report Time (TR): This is the time period between successive reports sent by a node.
- Attributes (A): This is a set of physical parameters which the user is interested in obtaining data about.

At every report time, the cluster members sense the parameters specified in the attributes and send the data to the cluster-head. The cluster-head aggregates this data and sends it to the base station or a higher level cluster-head. This ensures that the user has a complete picture of the entire area covered by the network. Important features of this scheme are as below:

- Since the nodes switch off their sensors and transmitters at all times except the report times, the energy of the network is conserved.
- At every cluster change time, TR and A are transmitted afresh and so, can be changed. Thus, by changing A and TR, the user can decide what parameters to sense and how

often to sense them. It is also possible that different clusters sense different attributes for different TR.

This scheme, however, has an important drawback. Because of the periodicity with which the data is sensed, it is possible that time critical data may reach the user only after the report time. Thus, this scheme may not be adequate for time-critical data sensing applications. In this text we will cover both proactive and reactive protocols, while highlighting that the protocol to be chosen is directly related to application requirements.

6.3 Fundamentals of MAC Protocol for Wireless Sensor Networks

Wireless medium is mostly a broadcast medium. All nodes within radio range of a node can hear its transmission. This can be used as a unicast medium by specifically addressing a particular node and all other nodes drop the packet they receive. There are two types of schemes available to allocate a single broadcast channel among competing nodes: Static Channel Allocation and Dynamic Channel Allocation.

- Static Channel Allocation: In this category of protocols, if there are N nodes, the bandwidth is divided into N equal portions either in frequency (FDMA: frequency division multiple access), in time (TDMA: time division multiple access), in code (CDMA: code division multiple access), in space (SDMA: space division multiple access) or OFDM (orthogonal frequency division multiplexing). Since each node is assigned a private portion, there is no interference between multiple users. These protocols works very well with efficient allocation mechanisms, when there are only a small and fixed number of users, each of which has buffered (heavy) load of data.
- Dynamic Channel Allocation: In this category of protocols, there is no fixed assignment of bandwidth. When the number of users changes dynamically and data is bursty at arbitrary nodes, it is most advisable to use dynamic channel allocation scheme. These are contention-based schemes, where nodes contend for the channel when they have data while minimizing collisions with other nodes' transmissions. When there is a collision, the nodes are forced to retransmit data,

  thus leading to increased wastage of energy of the nodes and unbounded delay. Example protocols are: CSMA (persistent and non-persistent) [Tanenbaum 1996], MACAW [Bharghavan 1994], IEEE 802.11 [Crow 1997], etc.

As we will see shortly, in a hierarchical clustering model, once clusters have been formed, the number of nodes in the cluster is fixed and due to hierarchical clustering, the number of nodes per cluster is also not large. So, with such a scenario, it is better to use one of the static channel allocation schemes. Studies [Heinzelman 2000a, Intanagonwiwat 2000] have pointed out the uses of TDMA for wireless sensor networks. In this scheme all the nodes transmit data in their slot to the cluster head and at all other times the radio can be switched off thereby saving valuable energy. When it is not possible to use TDMA, the nodes can use non-persistent CSMA since the data packets are of fixed size.

TDMA is suitable for either type of networks. In proactive networks since we have the nodes transmitting periodically, we can assign each node a slot and thus avoid collisions. In reactive networks, since adjacent nodes have similar data, when a sudden change takes place in some attribute being sensed, all the nodes will respond immediately. This will lead to collisions and it is possible that the data never reaches the user on time. For this reason, TDMA is employed so that each node is given a slot and they transmit only in that slot. Even though this increases the delay and some slots might be empty, it is better than the delay and energy consumption incurred due to dynamic channel allocation schemes.

CDMA is used to avoid inter cluster collisions. Though this means that more data needs to be transmitted per bit, it allows for multiple transmissions using the same frequency. A number of advantages have been pointed out for using TDMA/CDMA combination to avoid intra/inter cluster collision in ad hoc and sensor networks [Heinzelman 2000b].

6.4 Flat Routing in Sensor Networks

Routing in wireless sensor networks is very different from the traditional wired or wireless networks. Sensor networks are data centric, requesting information satisfying certain attributes and thus do not require routing of data between specific nodes. Also since adjacent nodes have almost similar data and might almost always satisfy the same attributes, rather than sending data separately from each node to the requesting node, it is desirable to aggregate similar data in a certain region before sending it. This aggregation is also known as “data fusion” [Brooks 1998, Varshney 1997]. Many protocols have been proposed that collect data based on the queries injected by the user or which always collect data so that the network is ready to answer any query the user asks. These protocols are based on the same concept as ad hoc networks where a route is set up only when needed (on-demand/reactive routing) or have a route from each node to every other node so that when it is needed, it is immediately available (proactive).We now look into protocols which collect data to answer queries injected by the user.

6.4.1 Directed Diffusion

Directed Diffusion [Intanagonwiwat 2000] is a data dissemination paradigm for sensor networks. It is a data-centric paradigm and is very useful to query dissemination and processing applications. The query is disseminated (flooded) throughout the network and gradients are setup to draw data satisfying the query towards the requesting node. Events (data) start flowing towards the requesting node from multiple paths. A small number of paths can be reinforced so as to prevent further flooding.

Such type of information retrieval is well suited only for persistent queries where requesting nodes are expecting data that satisfy a query for a duration of time. This makes it unsuitable for historical or one-time queries as it is not worth setting up gradients for queries which employ the path only once. Also this type of data collection does not fully exploit the feature of sensor networks, that adjacent nodes have similar data, as it uses a flat topology. At most, in this protocol data can be aggregated at the intermediate nodes.

6.4.2 Spin

In [Heinzelman 1999], a family of adaptive protocols called SPIN (Sensor Protocols for Information via Negotiation) has been proposed that disseminates all the information at each node to every node in the network. This enables a user to query any node and get the required information immediately. These protocols make use of the property that near-by nodes have similar data and thus distribute only the data which other nodes do not have. These protocols work pro-actively and distribute the information all over the network, even when a user does not request any data.

6.4.3 Cougar

Distributed Query processing results in several orders of magnitude times less message traffic and higher sensor lifetime than centralized query processing. In [Bonnet 2000, Bonnet 2001], it is discussed the application of distributed query execution techniques to improve communication efficiency in sensor and device networks. They discussed two approaches for processing sensor queries: warehousing and distributed. In the warehousing approach, data is extracted in a pre-defined manner and stored in a central database (BS). Subsequently, query processing takes place on the BS. In the distributed approach, only relevant data is extracted from the sensor network, when and where it is needed.

A model for sensor database systems known as COUGAR has been proposed, with appropriate user representation and internal representation of queries. The representation of sensor queries is also considered so that it is easier to aggregate the data and to combine two or more queries. Routing of queries is not being handled. Cougar has a three-tier architecture:

- The Query Proxy: A small database component running on the sensor nodes to interpret and execute queries.

- A Front end Component: A powerful query-proxy that allows the sensor network to connect to the outside world. Each front-end includes a full-fledged database server.
- A Graphical User Interface (GUI): Through the GUI, users can pose ad hoc and long running queries on the sensor network. A map component allows the user to query by region and visualize the topology of sensors in the network.

Queries are formulated regardless of the physical structure or the organization of the sensor network. Sensor data is different from the traditional relational data since it is not stored in a database server and it varies over time. Aggregate queries or correlation queries that give a bird eye's view of the environment also zoom on a particular region of interest. Each long running query defines a persistent view which it maintains during a given time interval. In addition, a sensor database should account for sensor and communication failures. It should consider sensor data as measurements with an associated uncertainty, and not as facts; the abstract data type represents data from physical sensors through representation by continuous distribution, thus capturing the uncertainty hidden in the sensor measurement. Finally, it should be able to establish and run a distributed query execution plan without assuming global knowledge of the sensor network.

In summary, the protocols we have seen so far use a flat topology which is not suitable for some applications of wireless sensor networks since through this topology one can not aggregate data from a number of near-by nodes and does not take full advantage of the specific features in sensor nodes. There are a number of clustering algorithms in literature and we discuss some of them in the next section. It is always important to keep in mind that different algorithms, whether viewing the topology as flat or hierarchical, are best suitable for different application environments.

6.5 Hierarchical Routing in Sensor Networks

Some authors suggest that a hierarchical clustering scheme is the most suitable for wireless sensor networks, as this model enables us to take advantage of all the features that are specific to sensor networks. The network is assumed to consist of a base station (BS), away from the nodes, through which the end user can access data from the sensor network. All the nodes in the network are homogeneous and begin with the same initial energy. The BS however has a constant power supply and so, has no energy constraints. Hence, it can be used to perform functions that are energy intensive. It can transmit with high power to all the nodes. Thus, there is no need for routing from the BS to any specific node. However, the nodes cannot always reply to the BS directly due to their power constraints, resulting in asymmetric communication. BS can also be used as a database to hold past data.

This model uses a hierarchical clustering scheme. Consider the partial network structure shown in Figure 19. Each cluster has a cluster head (CH) which collects data from its cluster members, aggregates it and sends it to the BS or an upper level cluster head. For example, nodes 1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.1.5 and 1.1 form a cluster with node 1.1 as the cluster head. Similarly, there exist other cluster heads such as 1.2, etc. These cluster-heads, in turn, form a cluster with node 1 as their cluster-head. So, node 1 becomes a second level cluster head as well. This pattern is repeated to form a hierarchy of clusters with the uppermost level cluster nodes reporting directly to the BS. The BS forms the root of this hierarchy and supervises the entire network. The main features of such architecture are:

- All the nodes transmit only to their immediate cluster-head, thus saving energy.
- Only the cluster head needs to perform additional computations on the data such as aggregation etc. So, energy is again conserved.
- The cluster members of a cluster are mostly adjacent to each other and have similar data. Since the cluster-heads aggregate similar data, aggregation can said to be more effective
- Cluster-heads at increasing levels in the hierarchy need to transmit data over relatively longer distances. As they need to perform extra computations, they end up consuming energy faster than the other lower level nodes. In order to evenly distribute this consumption, all the nodes take turns, becoming the cluster head for a time interval T, called the cluster period.
- Now since only the cluster-heads need to know how to route the data towards its own cluster-head or BS, it reduces complexity in data routing.

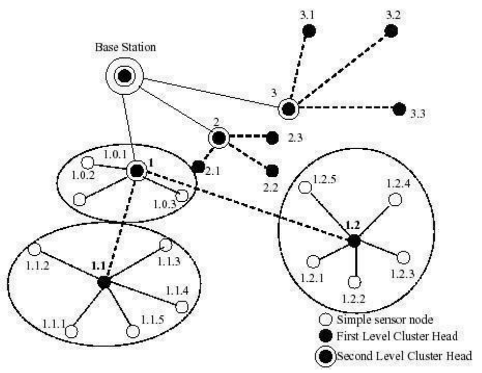


Figure 19 – Hierarchical Clustering

For applications which need to collect data for analysis of the situation/circumstances, it is adequate if we get data when the sensors are able to send it. But in applications that get data when something critical happens, such as “temperature going beyond 100ºF”, “more than 20 tanks passing by a region”, etc., but do not really care what happens in the network at other times, it is not desirable to waste sensors’ energy transmitting all the data they have collected. Ideally, it would be better if we could have flexibility in the network so that the user could decide how the network should behave based on the requirements and/or expectations.

6.5.1 Cluster Based Routing Protocol

A cluster based routing protocol (CBRP) has been proposed in [Jiang 1998] for sensor networks. It divides the network nodes into a number of overlapping or disjoint two-hop-diameter clusters in a distributed manner. Here, the cluster members just send the data to the cluster head (CH) and the CH routes the data to the destination. But this protocol is not suitable for wireless sensor networks as, due to high mobility, it requires a lot of “hello” messages to maintain the clusters. The sensor nodes do not have as much mobility and 2-hop-diameter clusters are not adequate to exploit the underlying feature of “adjacent nodes have similar data” in sensor networks.

6.5.2 Scalable Coordination

In [Estrin 1999], a hierarchical clustering method is discussed with emphasis on localized behavior and the need for asymmetric communication and energy conservation in sensor networks. In this method (no experimental results are provided) the cluster formation appears to require considerable amount of energy. Periodic advertisements are needed to form the hierarchy. Also, any changes in the network conditions or sensor energy level result in re-clustering which is not quite acceptable as some parameters tend to change dynamically.

6.5.3 Leach

It is introduced in [Heinzelman 2000b] a hierarchical clustering algorithm for sensor networks, called LEACH (Low-Energy Adaptive Clustering Hierarchy). LEACH is actually a family of protocols [Heinzelman 2000b] which suggests two schemes, distributed and centralized, that have minimal setup time and are also very energy efficient. One important feature of LEACH is that it utilizes randomized rotation of local cluster base stations (cluster-heads) to evenly distribute the energy load among the sensors in the network. They also make use of TDMA/CDMA MAC to reduce inter-cluster and intra-cluster collisions. LEACH is a good approximation of a proactive network protocol, with some minor differences.

Once the clusters are formed, the cluster heads broadcast a TDMA schedule giving the order in which the cluster members can transmit their data. Every node in the cluster is assigned a slot in the frame, during which it transmits data to the cluster head. When the last node in the schedule has transmitted its data, the schedule is repeated.

The report time discussed earlier is equivalent to the frame time in LEACH. The frame time is not broadcasted by the cluster head, but is derived from the TDMA schedule. However, it is not under user control. Also, the attributes are predetermined and are not changed after initial installation. This network can be used to monitor machinery for fault detection and diagnosis. It can also be used to collect data about temperature (or pressure, moisture, etc.) change patterns over a particular area.

But data collection is centralized and done periodically. This can be said to be most appropriate only for constant monitoring of networks. The user mostly always does not need all that data (immediately). So, periodic data transmissions are unnecessary, which it is saying as though very limited energy is consumed drains the limited energy from the sensors. This approach is similar to the warehousing approach.

6.5.4 Threshold sensitive Energy Efficient Network (TEEN)

In this section, we present the reactive network protocol called TEEN (Threshold sensitive Energy Efficient sensor Network protocol) [Manjeshwar 2001] with its time line depicted in Figure 20. In this scheme, at every cluster change time, in addition to the attributes, the cluster-head broadcasts the following to its members:

- *Hard Threshold* (HT): This is a threshold value for the sensed attribute. It is the absolute value of the attribute beyond which, the node sensing this value must switch on its transmitter and report to its cluster head.
- *Soft Threshold* (ST): This is a small change in the value of the sensed attribute which triggers the node to switch on its transmitter and transmit.

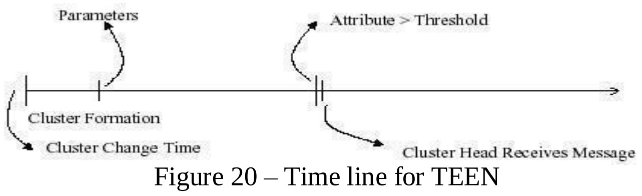


Figure 20 – Time line for TEEN

The nodes sense their environment continuously. The first time a parameter from the attribute set reaches its hard threshold value, the node switches on its transmitter and sends the sensed data. The sensed value is also stored in an internal variable in the node, called the sensed value (SV). The nodes will next transmit data in the current cluster period, only when both the following conditions are true:

- The current value of the sensed attribute is greater than the hard threshold.
- The current value of the sensed attribute differs from SV by an amount equal to or greater than
  the soft threshold.

Whenever a node transmits data, SV is set equal to the current value of the sensed attribute. Thus, the hard threshold tries to reduce the number of transmissions by allowing the nodes to transmit only when the sensed attribute is in the range of interest. The soft threshold further reduces the number of transmissions by eliminating all the transmissions which might have otherwise occurred when there is little or no change in the sensed attribute once the hard threshold is reached. The main features of this scheme are as follows:

- Time critical data reaches the user almost instantaneously. So, this scheme is eminently suited for time-critical data sensing applications.
- Message transmission consumes much more energy than data sensing. So, even though the nodes sense continuously, the energy consumption in this scheme can potentially be much less than in the proactive network, because data transmission is done less frequently.
- The soft threshold can be varied, depending on the criticality of the sensed attribute and the target application.
- A smaller value of the soft threshold gives a more accurate picture of the network, at the expense of increased energy consumption. Thus, the user can control the trade-off between energy efficiency and accuracy.
- At every cluster change time, the parameters are broadcast afresh and so, the user can change them as required.

The main drawback of this scheme is that, if the thresholds are not reached, the nodes will never communicate, the user will not get any data from the network at all and will never be able to know even if all the nodes die. Thus, this scheme is not well suited for applications where the user needs to get data on a regular basis. Another possible problem with this scheme is that a practical implementation would have to ensure that there are no collisions in the cluster. TDMA scheduling of the nodes can be used to avoid this problem. This will, however, introduce a delay in reporting of time-critical data. CDMA is another possible solution to this problem. This protocol is best suited for time critical applications such as intrusion and explosion detection.

6.5.5 Adaptive Periodic Threshold-sensitive Energy Efficient Sensor Network Protocol

There are applications in which the user wants time-critical data and also wants to query the network for analysis on conditions other than collecting time-critical data. In other words, the user might need a network that reacts immediately to time-critical situations and also gives an overall picture of the network at periodic intervals, so that it is able to answer analysis queries. Neither of the above networks can do both the jobs satisfactorily since they have their own limitations.

APTEEN (Adaptive Periodic Threshold-sensitive Energy Efficient Sensor Network Protocol) [Manjeshwar 2002] is able to combine the best features of proactive and reactive networks while minimizing their limitations to create a new type of network called a Hybrid network. In this network, the nodes not only send data periodically, they also respond to sudden changes in attribute values. This uses the same model as the above protocols with following changes. In APTEEN, once the cluster heads are decided the following events take place, in each cluster period. The cluster head first broadcasts the following parameters:

- Attributes (A): This is a set of physical parameters which the user is interested in obtaining data about.
- Thresholds: This parameter consists of a hard threshold (HT) and a soft threshold (ST). HT is a value of an attribute beyond which a node can be triggered to transmit data. ST is a small change in the value of an attribute which can trigger a node to transmit.
- Schedule: This is a TDMA schedule similar to the one used in [Heinzelman 2000b], assigning a slot to each node.
- Count Time (CT): It is the maximum time period between two successive reports sent by a node. It can be a multiple of the TDMA schedule length and it introduces the proactive component in the protocol

The nodes sense their environment continuously. However, only those nodes which sense a data value at or beyond the hard threshold, transmit. Furthermore, once a node senses a value beyond HT, it next transmits data only when the value of that attribute changes by an amount equal to or greater than the soft threshold ST. The exception to this rule is that if a node does not send data for a time period equal to the count time, it is forced to sense and transmit the data, irrespective of the sensed value of the attribute. Since nodes near to each other may fall in the same cluster and sense similar data, they may try sending their data simultaneously, leading to collisions between their messages. Hence, a TDMA schedule is used and each node in the cluster is assigned a transmission slot, as shown in Figure 21. In the sections to follow, we will refer to data values exceeding the threshold value as critical data.

The main features of this scheme are as follows:

- It combines both proactive and reactive policies. By sending periodic data, it gives the user a complete picture of the network, like a proactive scheme. It also senses data continuously and responds immediately to drastic changes, thus making it responsive to time critical situations. It, thus, behaves as a reactive network also.
- It offers a lot of flexibility by allowing the user to set the time interval (CT) and the threshold values for the attributes.
- Changing the count time as well as the threshold values can control energy consumption.
- The hybrid network can emulate a proactive network or a reactive network, based on the application, by suitably setting the count time and the threshold values.

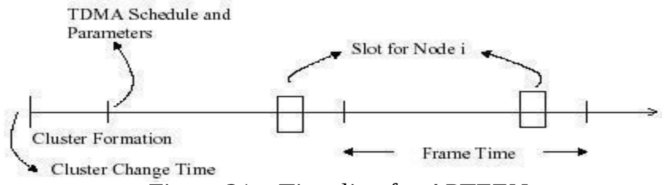


Figure 21 – Time line for APTEEN

The main drawback of this scheme is the additional complexity required to implement the threshold functions and the count time. However, this might be seen as a trade-off.

6.6 Comparison Table

Table 3 illustrates the characteristics of hierarchical and flat topologies for WSN.

Table 3 – Hierarchical versus Flat topologies for WSN

Hierarchical	Flat
Reservation-based scheduling	Contention-based scheduling
Collisions avoided	Collision overhead present
Reduced duty cycle due to periodic sleeping	Variable duty cycle by controlling sleep time of Nodes
Data aggregation by cluster head	Node on multi-hop path aggregates incoming data from neighbors
Simple but non-optimal routing	Routing is complex but optimal
Requires global and local synchronization	Links formed in the fly, without synchronization
Overhead of cluster formation throughout the network	Routes formed only in regions that have data for Transmission
Lower latency as multi-hop network formed by cluster-heads is always available	Latency in waking up intermediate nodes and setting up the multi-hop path
Energy dissipation is uniform	Energy dissipation depends on traffic patterns
Energy dissipation cannot be controlled	Energy dissipation adapts to traffic pattern
Fair channel allocation	Fairness not guaranteed

6.7 Adapting to the Inherent Dynamic Nature of Wireless Sensor Networks

Some important goals that current research in this area is aiming to achieve are as follows:

- Exploit spatial diversity and density of sensor/actuator nodes to, for example, build an adaptive node sleep schedule. Characterize the relationship between deployment density and network size. Explore of the trade-off between data redundancy and bandwidth consumption.
- The nodes on deployment should spontaneously create and assemble network, dynamically adapt to device failure and degradation, manage mobility of sensor nodes and react to changes in task and sensor requirements.
- Adaptability to traffic changes. Some nodes may detect an event which triggers a big sensor, like a camera, generating heavy traffic. But when sensing activity is low, traffic is light.
- Allowing finer control over an algorithm than simply turning off or on. Nodes should be capable of dynamically trading precision for energy or scope for convergence time based on incoming data.

The SCADDS Project (Scalable Coordination Architectures for Deeply Distributed Systems) [SCADDS], also a part of DARPA SensIT program, focuses on Adaptive fidelity, dynamically adjusting the overall fidelity of sensing in response to task dynamics (turn on more sensors when a threat is perceived). They use additional sensors (redundancy) to extend lifetime. Neighboring nodes are free to talk to each other irrespective of their listen schedules; there is no clustering and no inter-cluster communications and interference.

Adaptive Self-Configuring sEnsor Network Topologies (Ascent) [Cerpa 2001], which is part of SCADDS, focuses on how to decide which nodes should join the routing infrastructure to adapt to a wide variety of environmental dynamics and terrain conditions producing regions with non uniform communication density. In Ascent each node assesses its connectivity and adapts its participation in its multi-hop network topology based on the measured operating region. A node signals and reduces its duty cycle when it detects high message loss, requesting additional nodes in the region to join the network in order to rely messages to it. It probes the local communication environment and does not join the multi-hop routing infrastructure until it is helpful to do so. It avoids transmitting dynamic state information repeatedly across the network.

6.8 MAC Layer Design Issues in Wireless Sensor Networks

As with MAC protocols for traditional mobile ad hoc networks, sensor networks have their own issues that must be addressed. Below we will discuss some of the most important issues involved in the design of MAC protocols for wireless sensor networks.

6.8.1 Fighting Node Failure

When many nodes have failed, the MAC and routing protocols must accommodate formation of new links and routes to the sink nodes. This may require actively adjusting transmit powers and signaling rates on the existing links to reduce energy consumption, or rerouting packets through regions of the network where more energy is left.

6.8.2 Sources of Resource Consumption at the MAC Layer

There are several aspects of a traditional MAC protocol that happen to have negative effects on wireless sensor networks including:

- Collisions – When a transmitted packet is corrupted it has to be discarded. The follow-on retransmissions increase energy consumption and increase latency.
- Overhearing – Nodes listen to transmissions that are destined to other nodes.
- Control packets overhead – Sending and receiving control packets consumes energy, and less useful data packets can be transmitted. This overhead increases linearly with node density. Moreover, as more nodes fail in the network, more control messages are required to self configure the system resulting in more energy consumption.
- Idle Listening – Listening to receive possible traffic that is not sent. This is especially true in many sensor network applications. If nothing is sensed, nodes are in idle mode for most of the time.

6.8.3 Measures to Reduce Energy Consumption

One of the most cited methods to conserve energy in sensor networks is by avoiding to listen to idle channels, that is, neighboring nodes periodically sleep (radio off) auto synchronizing on sleep schedules. It is important to note that in wireless sensor networks fairness, latency, throughput and bandwidth utilization are secondary.

S-MAC (Sensor-MAC) [Ye 2002] is new MAC protocol specifically designed for wireless sensor networks. The main goal of the S-MAC protocol design is to reduce energy consumption, while supporting good scalability and collision avoidance. It tries to reduce energy consumption from almost all the sources that we have identified to cause energy waste, i.e., idle listening, collision, overhearing and control overhead. S-MAC consists of three major components: periodic listen and sleep, collision and overhearing avoidance, and message passing. S-MAC assumes that nodes are able to turn their radios off and on, and tune carrier frequency to a large number of available bands. It is a distributed protocol with flat topology that enables collection of nodes to discover their neighbors and establish transmission or reception schedules for communicating with them, without the need for any local or global master nodes. Links are formed on fly because non-synchronous slots are assigned. This concept is known as Non-synchronous scheduled communication (after link establishment each node knows ahead of time when to turn its transceiver on). This is to quickly retrieve information “trapped” in the low-duty cycle network as getting information to its ultimate destination in a timely manner is difficult when routes are blocked, nodes are turned off, and large fractions of the network are not available for long periods.

6.8.4 Comparison of Scheduling & Reservation based and Contention based MAC Design

One approach of MAC design for sensor networks is based on reservation and scheduling, for example TDMA based protocols that conserve more energy compared to contention based protocols like the IEEE 80211 DCF. This is because the duty cycle of the radio is increased and there is no contention-introduced overhead and collisions. However, formation of cluster, management of inter-cluster communication, and dynamic adaptation of the TDMA protocol to variation in the number of nodes in the cluster in terms of its frame length and time slot assignment are still its key challenges.

7. Standard Activities

7.1 Internet Engineering Task Force (IETF) Activities

The MANET Working Group [MANET] is a chartered working group established within the Internet Engineering Task Force (IETF) [IETF] to investigate and develop candidate standard Internet routing support for mobile, wireless IP autonomous segments. The primary focus of the working group is to develop and evolve MANET routing specifications and introduce them to the Internet Standards track. The goal is to support networks scaling up to hundreds of routers. If this proves successful, future work may include development of other protocols to support additional routing functionality. The working group also examines related security issues around a MANET.

Along with the MANET working group, the IETF has also established the mobileip (IP Routing for Wireless/Mobile Hosts) Working Group (WG) [MOBILEIP]. This WG has developed routing support to permit IP nodes (hosts and routers) using either IPv4 or IPv6 to seamlessly “roam” among IP subnetworks. The Mobile IP method supports transparency above the IP layer, including the maintenance of active TCP connections and UDP port bindings. Wherever this level of transparency is not required, solutions such as DHCP and dynamic DNS updates may be adequate and techniques such as Mobile IP not needed.

Future work is expected to focus on deployment issues in Mobile IP and provide appropriate protocol solutions to address known deficiencies and shortcomings. For example, the wireless/cellular industry is considering using Mobile IP as one technique for IP mobility for wireless data. The working group will endeavor to gain an understanding of data service in cellular systems such as GPRS, UMTS, CDMA2000, and interact with other standards bodies that are trying to adopt and deploy Mobile IP WG protocols in these context.

7.2 Bluetooth and Wireless PANs

In the past quarter century we have seen the rollout of three generations of wireless cellular systems attracting end-users by providing efficient mobile communications. On another front, wireless technology became an important component in providing networking infrastructure for localized data delivery. This later revolution was made possible by the induction of new networking technologies and paradigms, such as wireless local area networks (WLAN) and wireless personal area networks (WPAN).

Wireless personal area networks (WPANs) are short to very short-range (from a couple centimeters to a couple of meters) wireless networks that can be used to exchange information between devices in the reach of a person. WPANs can be used to replace cables between computers and their peripherals, to establish communities helping people do their everyday chores making them more productive, or to establish location aware services. Wireless local area networks (WLANs) on the other hand provide with a larger transmission range. Although WLAN equipment usually carries the capability to be set up for ad hoc networking, the premier choice of deployment is yet a cellular like infrastructure mode to interface wireless users with the Internet. The best example representing WPANs is the recent industry standard: Bluetooth [Bluetooth], other examples include Spike [Spike], and in the broad sense HomeRF [Negus 2000]. For WLANs, the most well known representatives are based on the standards IEEE 802.11 [Crow 1997] and HiperLAN [HiperLAN 1995] with all their variations.

The IEEE 802 committee has also realized the importance of short-range wireless networking and initiated the establishment of the IEEE 802.15 WG for WPANs [WPAN] to standardize protocols and interfaces for wireless personal area networking. Altogether, the 802.15 working group is formed by four Task Groups (TG):

- IEEE 802.15 WPAN/Bluetooth TG 1 – The TG 1 was established to support applications which require medium-rate WPANs (such as Bluetooth). These WPANs will handle a variety of tasks ranging from cell phones to PDA communications and have a QoS suitable for voice applications.

- IEEE 802.15 Coexistence TG 2 – Several wireless standards, such as Bluetooth and IEEE 802.11b, and appliances, such as microwaves operate, in the unlicensed 2.4 GHz ISM (Industrial-Scientific-Medical) frequency band. The TG 2 is developing specifications on the ISM band due to the unlicensed nature and available bandwidth. Thus, the IEEE 802.15 Coexistence TG 2 (802.15.2) for Wireless Personal Area Networks is developing Recommended Practices to facilitate coexistence of Wireless Personal Area Networks (802.15) and Wireless Local Area Networks (802.11).
- IEEE 802.15 WPAN/High Rate TG 3 – The TG3 for WPANs is chartered to draft and publish a new standard for high-rate (20Mbit/s or greater) WPANs. Besides a high data rate, the new standard will provide for low power, low cost solutions addressing the needs of portable consumer digital imaging and multimedia applications.
- IEEE 802.15 WPAN/Low Rate TG 4 – The goal of the TG 4 is to provide a standard having ultra-low complexity, cost, and power for a low-data-rate (200Kb/s or less) wireless connectivity among inexpensive fixed, portable, and moving devices. Location awareness is being considered

  as a unique capability of the standard. The scope of the TG 4 is to define the physical and media access control (MAC) layer

  specifications. Potential applications are sensors, interactive toys, smart badges, remote controls, and home automation.

One key issue in the feasibility of WPANs is the inter-working of wireless technologies to create heterogeneous wireless networks. For instance, WPANs and WLANs will enable an extension of the third generation (3G) cellular networks (i.e., UMTS and cdma2000) into devices without direct cellular access. Moreover, devices interconnected in a WPAN may be able to utilize a combination of 3G access and WLAN access by selecting the access that is best for the moment. In such networks 3G, WLAN and WPAN technologies do not compete against each other but enable the user to select the best connectivity for his/her purposes.

Figure 22 clearly shows the operating space of the various 802 wireless standards and activities still in progress.

Given the importance within the WPAN operating space, intensive research activities, and availability of devices, we will now devote a little time in, first, giving a brief introduction on Bluetooth and then provide an overview of the Bluetooth standard as defined by the Bluetooth SIG (Special Interest Group) [Bluetooth].

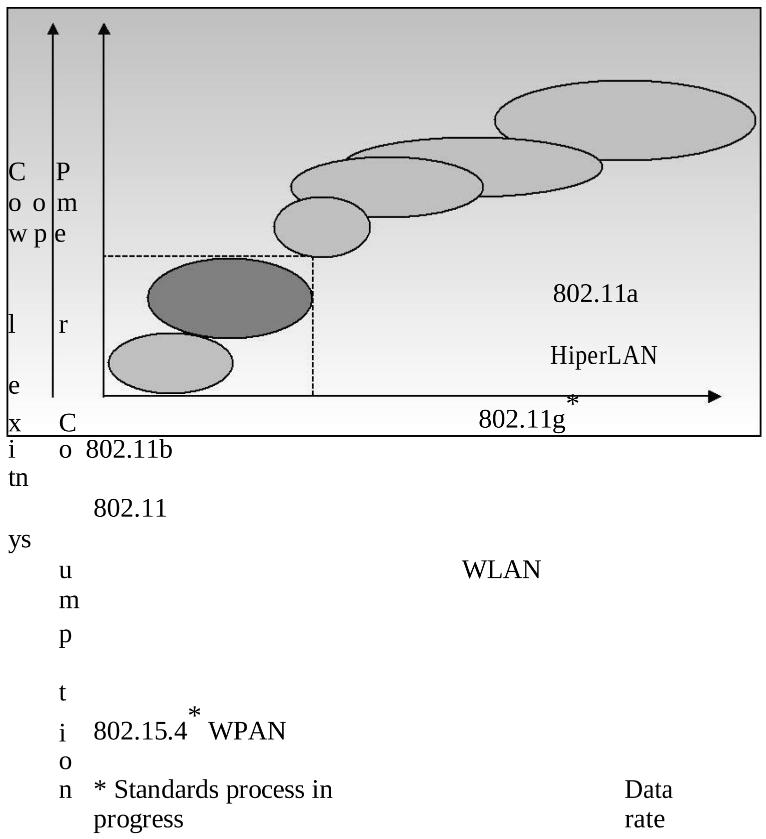


Figure 22 – The scope of the various WLAN and WPAN standards

7.2.1 Brief History and Applications of Bluetooth

In the context of ad hoc wireless networks, the Bluetooth technology came to light in May 1998, and since then the Bluetooth SIG has steered the development of the technology through the development of an open industry specification, including both protocols and applications scenarios. It is predicted that in 2006 Bluetooth will be present in 73 percent of phones and 44 percent of PDAs. It will provide device-to-device communication, enabling seamless communication between phones, printers, PDAs and scanners in the office and between phones, smart home control units, TVs and VCRs in the home. The Bluetooth specification comprises of an end-to-end description for both protocols and application profiles that guarantee value-added to its users right out-of-the-box. As per the current specification (version 1.1), it consists of the following two parts:

- The core specification defining the radio characteristics and the communication protocols for exchanging data between devices and Bluetooth radio links.
- The profile specification that defines how the Bluetooth protocols are to be used to realize a number of selected applications

.

Bluetooth has a tremendous potential in moving and synchronizing information in a localized setting. Potential for Bluetooth applications is huge, because we do business transactions and communicate more frequently with the people who are close by as compared to those who are far away - a natural phenomenon of human interaction.

7.2.2 An Overview of the Bluetooth Wireless Technology

While we give here only an overview of Bluetooth, its system, architecture and protocols are defined in detail in [Bluetooth]. Bluetooth operates in the ISM frequency band starting at 2.402 GHz and ending at 2.483 GHz in USA and most countries of Europe. A total of 79 RF channels of 1 MHz width are defined, where the raw data rate is 1 Mbit/s. A Time Division Multiplexing (TDD) technique divides the channel into 625µs slots and, with a 1Mbit/s symbol rate, a slot can carry up to 625 bits. Transmission in Bluetooth occurs in packets that occupy 1, 3 or 5 slots. Each packet is transmitted on a different hop frequency with a maximum frequency hopping rate of 1600 hops/s.

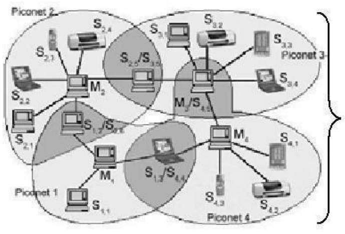
Bluetooth operates on a Master-Slave concept whereby the Master device controls data transmissions through a polling scheme. The Master periodically polls the Slave devices for information and only after receiving such a poll is a Slave allowed to transmit. A Master device can directly control seven active Slave devices in what is defined as a piconet. Multiple piconets can be linked together through common Bluetooth devices to form a scatternet. Figure 23 illustrates a scatternet composed of four piconets, where each piconet has several slaves (indicated by the letter Si,j) and one master (indicated by the letter Mi).

Figure 24 depicts the Bluetooth protocol stack, which also shows the application "layer" where the profiles would reside. The protocols that belong to the core specification are:

- The Radio – The radio layer, which resides below the Baseband layer, defines the technical characteristics of the Bluetooth radios. The Bluetooth radios come in three power classes, depending on their transmit power. Class 1 radios have transmit power of 20 dBm (100mW); class 2 radios have transmit power of 4 dBm (2.5mW); class 3 radios have transmit power of only 0 dBm (1mW).
- The Baseband – The baseband defines the key procedures that enable devices to communicate with each other using the Bluetooth wireless technology. The baseband defines the Bluetooth piconets (see Figure 24) and how they are created, and the Bluetooth links. It also defines the low-level packet types.
- The Link Manager Protocol (LMP) – The LMP is a transactional protocol between two link management entities in communicating Bluetooth devices whose responsibilities is to setup the properties of the link.

- The Logical Link Control & Adaptation Protocol (L2CAP) – The L2CAP layer shields the specifics of the Bluetooth lower layers and provides a packet interface to higher layers. At L2CAP, the concepts of master and slave devices do not exist anymore.

The Bluetooth specification defines two distinct types of links for the support of voice and data applications, namely, SCO (Synchronous connection-oriented) and ACL (Asynchronous connectionless). The first link type supports point to point voice switched circuits while the latter supports symmetric as well as asymmetric data transmission. ACL packets are intended to support data applications and do not have prescribed time slot allocations as opposed to SCO packets, which support periodic audio transmission at 64Kb/s in each direction.



Bluetooth scatternet

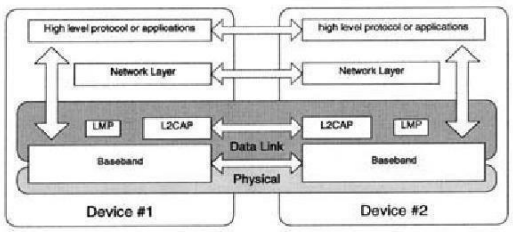


Figure 23 – Four piconets forming a scatternet

Figure 24 – Bluetooth protocol architecture

8. Open Problems

As we have already mentioned, the research in the area of Mobile Ad hoc Networking is far from being exhaustive. Much of the effort so far has been on devising routing protocols to support the effective and efficient communication between nodes that are part of the network. However, there are still many topics that deserve further investigation such as:

- Scalability – To what extent can an ad hoc network grow?
- Address configuration – The address scheme used in wired networks (e.g., DHCP), as well as in Mobile IP, might not be adequate in a MANET. A new addressing approach may be required for MANETs.
- Interoperation with the Internet – How can ad hoc networks seamlessly and efficiently access the Internet in order to obtain advanced services?
- Improvement of interaction between layers – Would it be better to interact layers in order to achieve better performance?
- Quality of service (QoS) – Is it feasible for bandwidth/delay-constrained applications to run well in a MANET?
- Applications for MANET – Have we found a killer application?
- Security – How can the network secure itself from malicious or compromised hosts?
- Power control – How can battery life be maximized?

UNIT-II

DATA TRANSMISSION IN MANETS

Doing network-wide broadcasting in ad hoc networks requires one device to broadcast the information to all its neighbors. For far-away devices, the message is rebroadcasted which could cause collision if multiple device broadcasts the same time and are in the neighborhood. This is also known as the broadcasting storm problem [Nil999] and in this section we discuss ways to perform efficient broadcasting of messages.

we assume that MHs in the MANET share a single common channel with carrier sense multiple access (CSMA) [Agrawal2002], but no collision detection (CD) capability (e.g., the IEEE standard 802.11 [IEEE-802.111997]). Synchronization in such a network with mobility is unlikely, and global network topology information is unavailable to facilitate the scheduling of a broadcast. Thus, one straightforward and obvious solution is to achieve broadcasting by flooding (for example, as it is done by mostly all MANET routing algorithms). Unfortunately, as we will see later, it is observed that redundancy, contention, and collision could exist if flooding is done blindly. Several problems arise in these situations including:

• As the radio propagation is omnidirectional and a physical location may be covered by the transmission ranges of several hosts, many rebroadcasts are considered to be redundant

• Heavy contention could exist because rebroadcasting hosts are probably close to each other; and

• As the RTS/CTS handshake (e.g., employed in the IEEE standard (802.11) is inapplicable for broadcast transmissions, collisions are more likely to occur as the timing of rebroadcasts is highly correlated.

<u>Broadcasting in a MANET:</u>

A MANET consists of a set of MHs that may communicate with one another from time to time, and here no base stations are present. Each host is equipped with a CSMA/CA (carrier sense multiple access with collision avoidance) [Agrawal2002] transceiver. In such an environment, a MH may communicate with each other directly or indirectly. In the latter case, a multi-hop scenario occurs, where the packets originated from the source host are relayed by several intermediate MHs before reaching the destination. The broadcast problem refers to the transmission of a message to all other MHs in the network. The problem we consider has the following characteristics.

• **The broadcast is spontaneous:** Any MH can issue a broadcast operation at any time. For reasons such as the MH mobility and the lack of synchronization, preparing any kind of global topology knowledge is prohibitive (in fact, this is at least as hard as the broadcast problem). Little or no local information may be collected in advance.

• **The broadcast is frequently unreliable:** Acknowledgement mechanism is rarely used. However, attempt should be made to distribute a broadcast message to as many MHs as possible without putting too much effort. The motivations for such an assumption are:

1 .A MH may miss a broadcast message because it is off-line, it is temporarily isolated from the network, Or it experiences repetitive collisions;
2. Acknowledgements may cause serious medium contention (and thus another "storm") surrounding the Sender
3. In many applications (e.g., route discovery in ad hoc routing Protocols), a 100% reliable broadcast is Unnecessary.

We focus on the flooding behavior in a MANET – the phenomenon where the transmission of a packet will trigger other surrounding MHs to transmit the same (or modified) packet. We shall show that if flooding is used blindly, many redundant messages will be sent and serious contention/collision will be incurred.

MULTICASTING

In this section, we investigate the problem of multicasting in MANETs where the problem is to broadcast a message to a subset of MANET MHs. We begin by understanding the hard task of multicasting to a group of mobile nodes, together with the various issues behind the design and implementation of a multicast protocol for MANETs. Next, we study the existing multicast protocols for MANETs and show how different they are as compared to broadcasting.

3.3.1 Issues in Providing Multicast in a MANET

Well-established routing protocols do exist to offer an efficient multicasting service in conventional wired networks. Protocols, designed for fixed networks, may fail to keep up with node movements and frequent topological changes as MHs become increasingly mobile, these protocols need to evolve to cope up with the new environment. But the host mobility increases the protocol overheads substantially. The broadcast protocols cannot be used either as multicasting requires a selected set of nodes to receive the message while all multicast algorithm depend on the topology of the network and do not consider whether a node belongs to a group or not. Rather, new protocols are being proposed and investigated which take issues such as locations of nodes belonging to a multicast group, and all associated topological changes. Moreover, the nodes of MANET run on batteries, routing protocols must limit the amount of control information that is passed between nodes. The majority of applications are in the areas where rapid deployment and dynamic reconfiguration are necessary and the wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications lend themselves well to multicast operation. In addition, within a wireless medium, it is even more crucial to reduce the transmission overhead and power consumption. Transient loops may form during reconfiguration of distribution structure (e.g., tree) as a result of mobility. Therefore, reconfiguration scheme should be kept simple to maintain low channel overhead. As we can see, providing an efficient multicasting over MANET faces many challenges including dynamic group membership and constant update of delivery path due to node movement. In the next sections, we cover the major protocols proposed so far and compare them under different criteria.

Multicast Routing Protocols:

We can classify the protocols into four categories based on how route to the members of the group is created:

• Tree-Based Approaches

• Meshed-Based Approaches

• Stateless Multicast

• Hybrid Approaches.

1.Tree-Based Approaches

Most of the schemes for providing multicast in wired network are either source-based or shared tree- based. Different researchers have tried to extend the idea of tree-based approach to provide multicast in a MANET environment. Due to simplicity and innate properties of tree structures, many characteristics can be identified such as: a packet traverses each hop and node in a tree at most once, very simple routing decisions at each node, and the number of copies of a packet is minimized, tree structure built presenting shortest paths amongst nodes, and a loop-free data distribution structure. On the other hand, there are many issues that must be addressed in tree-based approaches. As mentioned earlier, trees provide a unique path between any two nodes. Therefore, having even one link failure could mean reconfiguration of the entire tree structure and could be a major drawback. In addition, multiple packets generated by different sources will require some consideration when utilizing multicast trees such that efficient routing can be established and maintained. Thus, it is common to consider the use of either a shared tree or establish a separate tree per each source (i.e., separate source trees). As highlighted in [Garcia-Luna-Aceves 1999a], each approach has to deal with own individual issues. For separate source trees, each router (or node in case of MANETs) involved in multiple router groups must maintain a list of pertinent information for each group in which it is involved. Such management per router is inefficient and not scalable. On the other hand, for shared trees, there is a potential that packets may not only not traverse shorter paths, but in fact may be routed on paths with much longer distances than the shortest paths. While any scheme has positive and negative sides, the simple structured coupled with ease of approach has made multicast trees the primary method for realizing multicasting on the Internet. Due to this fact, tree-based approaches for ad hoc networks have been investigated and we will study them in the following sections.

2.Mesh-Based Approaches:

In contrast to the tree-based approach, mesh-based multicast protocols may have multiple paths between any source and receiver pairs. Existing studies show that tree-based protocols are not necessarily the best suited for multicast in a MANET environment if the network topology changes frequently. In such an environment, mesh-based protocols seem to outperform tree-based proposals due to availability of alternative paths, which allow multicast data grams to be delivered to the receivers even if links fail.

The disadvantage of a mesh is the increase in data-forwarding overhead. The redundant forwarding consumes more bandwidth in the bandwidth constrained ad hoc networks. Moreover, the probability of Collision s is higher when a larger number of packets are generated. Therefore, one common problem mesh-based protocols have to consider is how to minimize the data-forwarding overhead caused by flooding. As we shall see, different protocols attack this issue in different ways through the use of forwarding groups, cores, and so on. This section gives an overview of the mesh-based approaches that support multicasting MANETs.

3. Stateless Approaches:

Tree-based and mesh-based approaches have an overhead of creating and maintaining the delivery tree/mesh with time. In a MANET environment, frequent movement of MHs considerably increases the overhead in maintaining the delivery tree/mesh. To minimize the effect of such a problem, stateless multicast is proposed wherein a source explicitly mentions the list of destinations in the packet header. Stateless multicast approaches focus on small group multicast and assumes the underlying routing protocol to take care of forwarding the packet to the respective destinations based on the addresses contained in the header. In this section we present the main stateless multicast routing protocols proposed for use in MANETs.

Differential Destination Multicast

Differential Destination Multicast (DDM) protocol [Ji2001] is meant for small-multicast groups operating in dynamic networks of any size. Unlike other MANET routing protocols, DDM lets source to control multicast group membership. The source encodes multicast receiver addresses in multicast data packets using a special DDM Data Header. This variable length destination list is placed in the packet headers, resulting in packets being self-routed towards the destinations using the underlying unicast routing protocol. It eliminates maintaining per-session multicast forwarding states at intermediate nodes and thus is easily scalable with respect to the number of sessions. DDM supports two kinds of operating modes: "stateless" and "soft state". In stateless mode, the nodes along the data forwarding paths need not maintain multicast forwarding states. An intermediate node receiving a DDM packet only needs to look at the header to decide how to forward the packet. In the "soft-state" mode, based on in-band routing information, each node along the forwarding path remembers the destinations to which the packet has been forwarded last time and its next hop information. By caching this routing information at each node, the protocol does not need to list the entire destination in future data packets.

In case changes occur in the underlying unicast routing, an upstream node only needs to inform its downstream nodes about the differences in the destination forwarding since the last packet; hence the name "Differential Destination Multicast". At each node, there is one Forwarding Set (FS) for each multicast session, which records to which destinations this node forwards data. The nodes also maintain a Direction Set (DS) to record the particular next hop to which multicast destination data are forwarded. At the source node, FS contains the same set of nodes as the multicast Member List (ML). In the intermediate nodes, the FS is the union of several subsets based on the data stream received from upstream neighbors. Associated with each set FS_k, there is a sequence number SEQ(FS_k) which is used to record the last DDM Block Sequence Number seen in a received DDM data packet from an upstream neighbor k. It helps to detect loss of data packet containing the forwarding set updates. At a given node, FS also needs to be partitioned into subsets according to the next hops for different destination mechanism defined by the DSR unicast protocol to flood the data packets in the network. Although this is derived from DSR, it can be implemented as a stand-alone protocol. In fact, it does not rely on unicast routing to operate. If DSR has already been implemented on the network, minor modifications are required to enable this protocol. This multicast and broadcast protocol utilizes controlled flooding to distribute data in the network and does not require establishment of a state in the network for data delivery. It is not intended as a general purpose multicast protocol. Its applicability is mainly in environments characterized by very high mobility or by a relatively small number of nodes. In the former case, protocols relying on the establishment of multicast state perform inadequately because they are unable to track the rapid changes in topology. In the latter case, the overhead of keeping multicast state exceeds the overhead of flooding.

DSR Simple Multicast and Broadcast Protocol

The DSR Simple Multicast and Broadcast protocol (DSR-MB) [Jetcheva2001b] is designed to provide multicast and broadcast functionality in ad hoc networks. It utilizes the Route Discovery .

Hybrid Approaches

The protocols to provide multicast in ad hoc networks discussed so far, either address efficiency or robustness but not both simultaneously. The tree-based approaches provide high data forwarding efficiency at the expense of low robustness, whereas mesh-based approaches lead to better robustness (link failure may not trigger a reconfiguration) at the expense of higher forwarding overhead and increased network load. Thus, there is a possibility that a hybrid multicasting solution may achieve better performance by combining the advantages of both tree and meshed-based approaches. In this section, we explore the different hybrid approaches to enable ad hoc multicasting.

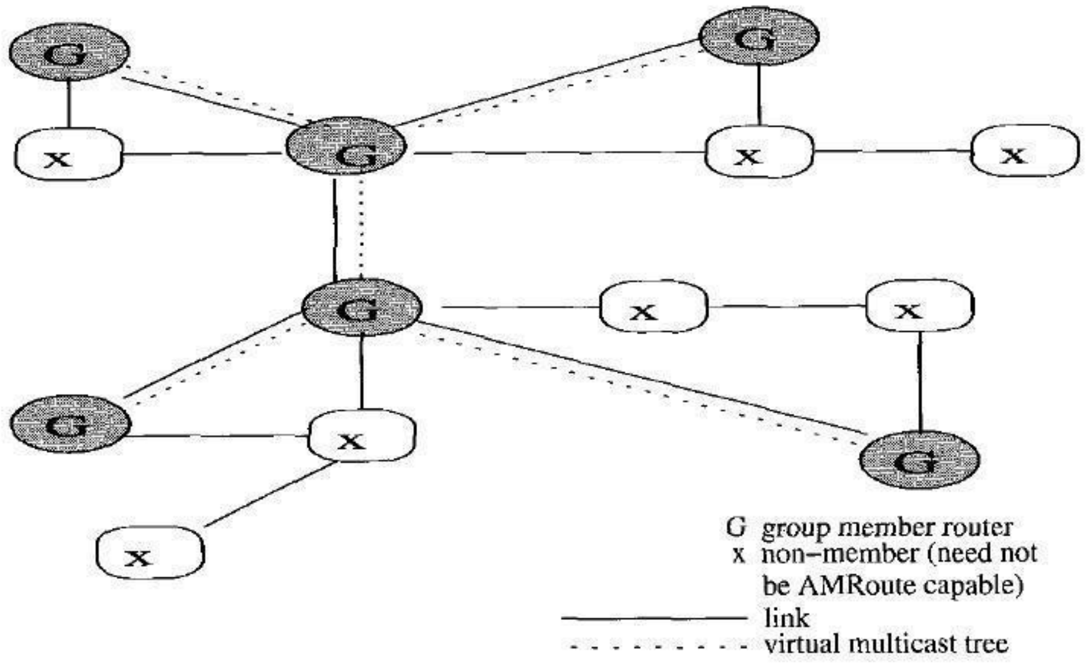
Ad Hoc Multicast Routing Protocol

The Ad hoc Multicast Routing Protocol (AMRoute) [Bommaiahl998] creates a bi-directional, shared tree by using only group senders and receivers as tree nodes for data distribution. The protocol has two main components: mesh creation and tree setup The mesh creation identifies and designates certain nodes as logical cores and these are responsible for initiating the signaling operation and maintaining the multicast tree to the rest of the group members. A non core node only responds to messages from the core nodes and serves as a passive agent. The selection of logical core in AMRoute is dynamic and can migrate to any other member node, depending on the network dynamics and the group membership. AMRoute does not address network dynamics and assumes the underlying unicast protocol to take care of it. To create a mesh, each member begins by identifying itself as a core and broadcasts JOIN_REQ packets with increasing TTL to discover other members. When a core receives JOIN_REQ from a core in a different mesh for the same group, it replies with a JOIN_ACK. A new bi-directional tunnel is created between the two cores and one of them is selected as core after the mesh merger. Once the mesh has been established, the core initiates the tree creation process. The core sends out periodic TREE_CREATE messages along all links incident on its mesh. Using unicast tunnels, the TREE_CREATE messages are sent only to the group members. Group members receiving non-duplicate TREE_CREATE message forwards it to all mesh links except the incoming one, and marks the incoming and outgoing links as a tree links. If a link is not going to be used as part of the tree, the TREE_CREATE is discarded and TREE_CREATE__NAK is sent back to incoming links. A member node, which wants to leave a group, can do so by sending a JOESLNAK message to its neighboring nodes. AMRoute employs the virtual mesh links to establish the multicast tree, which helps in keeping the multicast delivery tree the same even with the change of network topology as long as routes between core nodes and tree members exist via mesh links. The main disadvantage of this protocol is that it may have temporary loops and may create nonoptimal trees in case of mobility.

Multicast Core-Extraction Distributed Ad Hoc Routing

The Multicast Core-Extraction Distributed Ad hoc Routing (MCEDAR) [Sinhal999] is a multicast extension to the CEDAR architecture. The main idea of MCEDAR is to provide the efficiency of the tree-based forwarding protocols and robustness of mesh-based protocols by combining these two approaches. It is worth pointing out that a source-based forwarding tree is created on a mesh. As such, this ensures that the infrastructure is robust and data forwarding occurs at minimum height trees. MCEDAR decouples the control infrastructure from the actual data forwarding in order to reduce the control overhead.

Chapter 3: Broadcasting, Multicasting and Geocasting 123



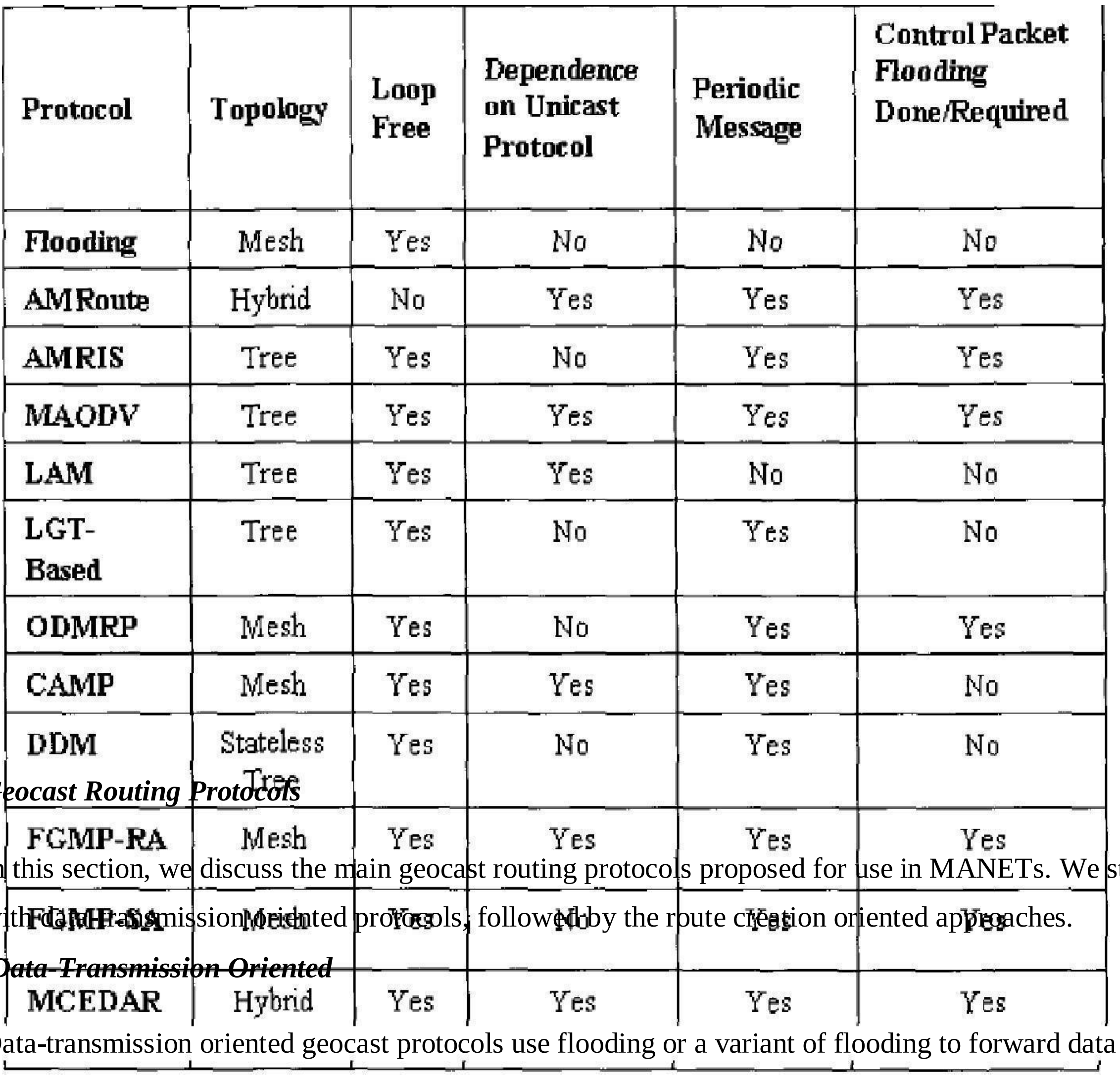
3.10 – AMRoute virtual multicast tree [Taken from IEEE Publication Cordeiro2003]

Mobility-Based Hybrid Multicast Routing

The Mobility-based Hybrid Multicast Routing (MHMR) protocol [An2001] is built on top of the mobility-based clustering infrastructure. In order to deal with the issues of scalability and stability, the structure is hierarchical in nature. The mobility and positioning information is provided via a GPS for each node. For a group of nodes, a cluster-head is chosen to manage and monitor the nodes in a cluster. A mesh structure is built based on all the current clusters. Thus, MHMR achieves high stability. This is followed by a tree structure built based on the mesh to ensure that the multicasting group achieves maximal efficiency. MHMR also provides a combination of proactive and reactive concepts which enable low route acquisition delay of proactive schemes while achieving low overhead of reactive methods. It is interesting to note that cores are employed in both AMRoute and MCEDAR, as well as in many tree and mesh multicast algorithms. The use of cores has been shown to lower the control overhead. The use of cluster-heads has been proposed in MHMR. This has been shown to be a reasonable approach since dividing the nodes in an ad hoc network into clusters seems to be a promising method in taking care of highly dynamic nodes. Hybrid methods can reveal themselves to be attractive as they can provide protocols that can address further robustness and efficiency. Though hybrid protocols have not been as deeply investigated as tree and mesh protocols, they are under development and recent results indicate its promising future.

Geocasting

We now turn our attention to the problem of geocasting over MANETs. As we have mentioned earlier, geocasting is a variant of the conventional multicasting problem and distinguishes itself by specifying hosts as group members within a specified geographical region. In geocasting, the nodes eligible to receive packets are implicitly specified by a physical region and membership changes as mobile nodes move in or out of the geocast region.



Protocol	Topology	Loop Free	Dependence on Unicast Protocol	Periodic Message	Control Packet Flooding Done/Required
Flooding	Mesh	Yes	No	No	No
AMRoute	Hybrid	No	Yes	Yes	Yes
AMRIS	Tree	Yes	No	Yes	Yes
MAODV	Tree	Yes	Yes	Yes	Yes
LAM	Tree	Yes	Yes	No	No
LGT-Based	Tree	Yes	No	Yes	No
ODMRP	Mesh	Yes	No	Yes	Yes
CAMP	Mesh	Yes	Yes	Yes	No
DDM	Stateless Tree	Yes	No	Yes	No
FGMP-RA	Mesh	Yes	Yes	Yes	Yes
FGMP-SA	Mesh	Yes	No	Yes	Yes
MCEDAR	Hybrid	Yes	Yes	Yes	Yes

Geocast Routing Protocols

In this section, we discuss the main geocast routing protocols proposed for use in MANETs. We start with data-transmission oriented protocols, followed by the route creation oriented approaches.

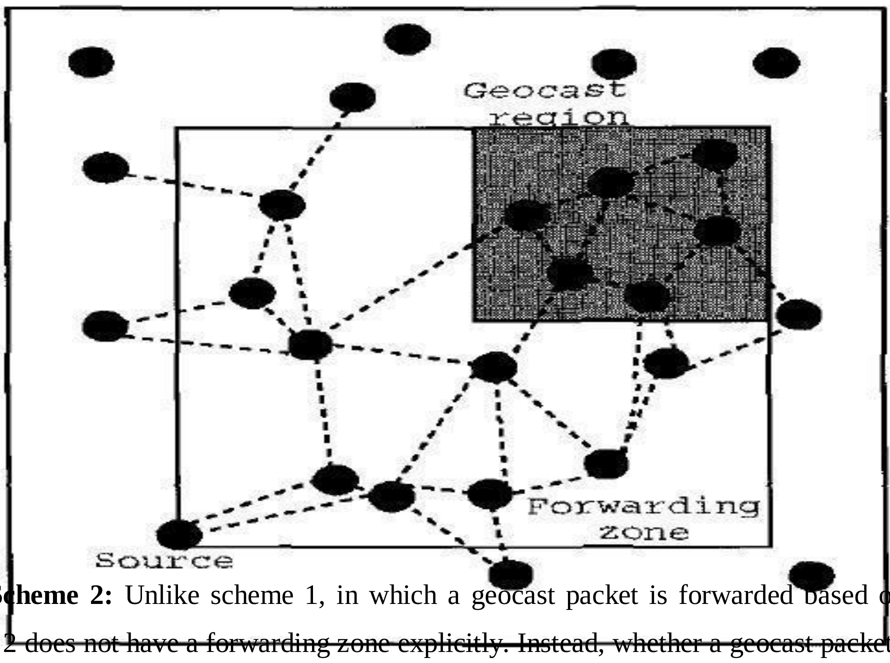
Data-Transmission Oriented

Data-transmission oriented geocast protocols use flooding or a variant of flooding to forward data from the source to the geocast region - and are described here.

3.4.1.1.1 Location-Based Multicast

The Location-Based Multicast (LBM) protocol [Kol999] extends the LAR unicast routing algorithm for geocasting. As we have seen, LAR is an approach to utilize location information to improve the performance (i.e., higher data packet delivery ratio and lower overhead) of a unicast routing protocol in a MANET. Similarly, the goal of LBM is to decrease delivery overhead of geocast packets by reducing the forwarding space for geocast packets, while maintaining accuracy of data delivery. The LBM algorithm is based upon a flooding approach. LBM is essentially identical to flooding data packets, with the modification that a node determines whether to forward a geocast packet further via one of two schemes.

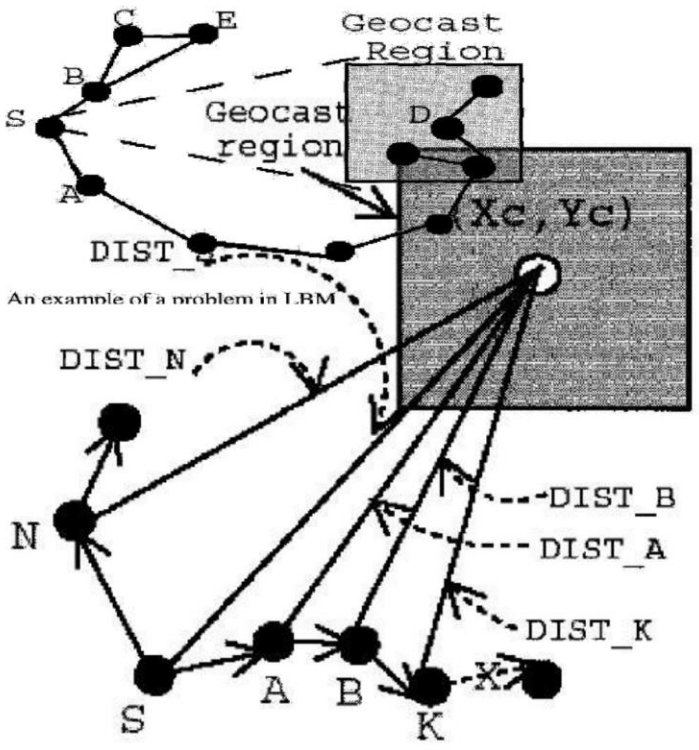
• **LBM Scheme 1:** When a node receives a geocast packet, it forwards the packet to its neighbors if it is within *& forwarding zone;* otherwise, it discards the packet. Thus, how to define the forwarding zone becomes the key point of this scheme. Figure 3.11 shows one example of a forwarding zone Boleng2001]. In Figure 3.11, the size of the forwarding zone is dependent on (i) the size of the geocast region and (ii) the location of the sender. In a BOX Forwarding Zone, the smallest rectangle that covers both the source node and the geocast region defines the forwarding zone. All the nodes in the forwarding zone forward data packets to their neighbors. Other kinds of forwarding zones are possible, such as the CONE Forwarding Zone [Boleng2001]. A parameter <5is discussed in [Kol999] to provide additional control on the size of the forwarding zone. When. *S* is positive, the forwarding zone is extended in both positive and negative X and Y directions by <5.(ie., each side increases by 2 *S*).



BOX forwarding zone

LBM Scheme 2: Unlike scheme 1, in which a geocast packet is forwarded based on the forwarding zone, scheme 2 does not have a forwarding zone explicitly. Instead, whether a geocast packet should be forwarded is based on the position of the sender node at the transmission of the packet and the position of the geocast region. That is, for some parameter *8*, node B forwards a geocast packet from node A (originated at node S), if node B is "at least. *S* closer" to the center of the geocast region *(Xc, Yc)* than node A. In other words, *DISTA > DISTB+ S*. We define *(Xc, Yc)* as the location of the geometrical center of the geocast region, and for any node

Z, *DISTz* denotes the distance of node Z from *(Xc, Yc)*. In Figure 3.12 [Kol999], node B will forward a geocast packet transmitted by node A since *DISTA* > *DISTB* and. *d*= 0. Node K will, however, discard a geocast packet transmitted by node B, since node K is not closer to *(Xc, Yc)* than node B. In brief, this protocol ensures that every packet transmission sends the packet closer to the destination. As for the performance, the accuracy (i.e., ratio of the number of geocast group members that actually receive the geocast packets to the number of group members that were supposed to receive the packets) of both LBM schemes is comparable with that of flooding geocast packets throughout the network. However, the number of geocast packets transmitted (a measure of the overhead) is consistently lower for LBM than simple flooding.

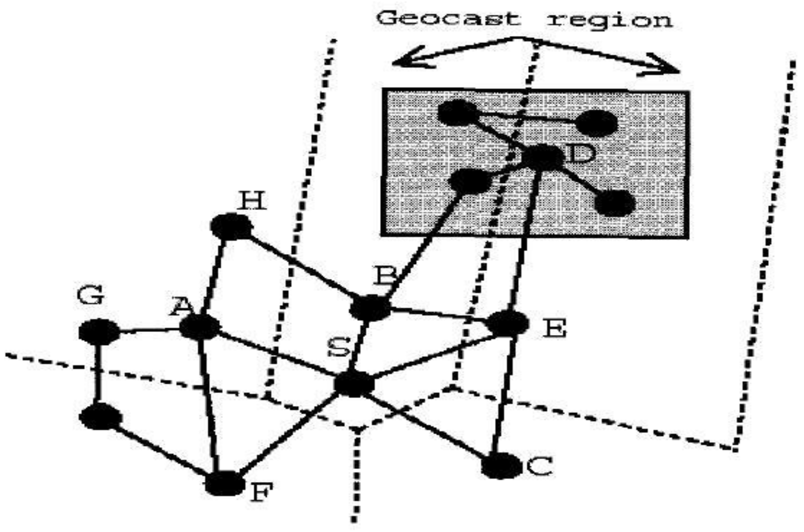


Forwarding zone in LBM scheme 2

Voronoi Diagram Based Geocasting

The goal of the Voronoi Diagram based Geocasting (VDG) protocol [Stojmenovicl999] is to enhance the success rate and decrease the hop count and flooding rate of LBM. It is observed that the forwarding zone defined in LMB may be a partitioned network between the source node and the geocast region, although there exists a path between the source and the destination.

In VDG, the definition of the forwarding zone of LBM has been modified. The neighbors of node A that are located within the forwarding zone in VDG are exactly those neighbors that are closest in the irection of the destination. This definition of a forwarding zone not only resolves the problem of having no nodes inside the forwarding zone, but also precisely determines the expansion of the forwarding zone. This forwarding zone can be implemented with a *Voronoi diagram* for a set of nodes in a given node's neighborhood of a MANET. A Voronoi diagram of *n* distinct points (i.e., *n* neighbors) in a plane is a partition of the plane into *n* Voronoi regions, which, when associated with node A, consists of all the points in the plane that are the closest to A. In other words, the *Voronoi diagram model* is a model where every point is assigned to a Voronoi region. The subdivision induced by this model is called the *Voronoi diagram* of the set of nodes [Berg]. For example, in Fig, five neighbors of source node S (A, B, C, E and F) carve up the plane into five Voronoi regions. The region associated with node A, consists of nodes G and H, since these two nodes are closer to node A than to any other node. The geocast region is the rectangle with the center D. In Figure 3.14, the Voronoi regions of nodes B and E intersect the geocast region; thus, only nodes B and E will forward geocast packets from node S. Although there are not any simulations of the VDG algorithm, it is believed that VDG reduces the flooding rates of LBM Scheme 1, as fewer packets should be transmitted. On the other hand, VDG may offer little improvement over LBM Scheme 2, as the end result of the two protocols appears to be similar.



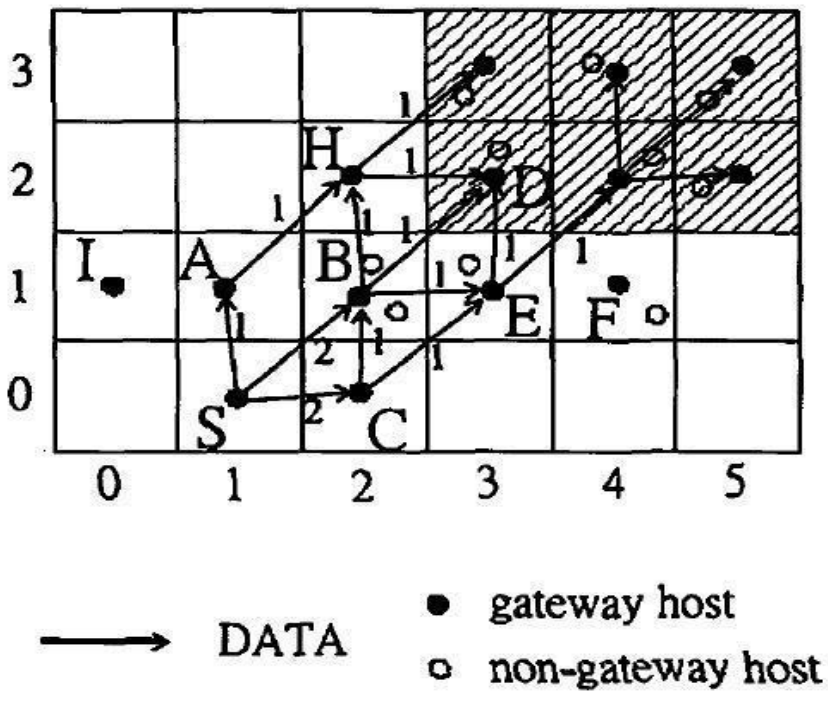
Example of a Voronoi diagram and the request zone

GeoGRID

Based on the unicast protocol GRID [Liao2001], the GeoGRID protocol [Liao2000] uses location information, which defines the forwarding zone and elects a special host (i.e, *gateway)* in each grid area responsible for forwarding the geocast packets. It is argued in [Liao2000] that the forwarding zone in LBM incurs unnecessary packet transmissions, and a tree-based solution is prohibitive in terms of control overhead. GeoGRID partitions the geographic area of the MANET into two-dimensional (2D) logical grids. Each grid is a square of size *d X d* (there are trade-offs in choosing a good value of *d,* as discussed in [Liao2000].) In GeoGRID, a gateway node is elected within each grid. The forwarding zone is defined by the location of the source and the geocast region. The main difference between GeoGRID, LBM and VDG is the following: in GeoGRID, instead of every node in a forwarding zone transmitting data, only gateway nodes take this responsibility. There are two schemes on how to send geocast packets in GeoGRID:

Flooding-Based GeoGRID and *Ticket-Based GeoGRID.* In Flooding-Based GeoGRID, only gateways in every grid within the forwarding zone rebroadcast the received geocast packets. Thus, gateway election becomes the key point of this protocol. In Ticket-Based GeoGRID, the geocast packets are still forwarded by gateway nodes, but not all the gateways in the forwarding zone forward every geocast packet. A total of *m* + *n* tickets are created by the source if the geocast region is a rectangle of *m X n* grids. The source evenly distributes the *m* + *n* tickets to the neighboring gateway nodes in the forwarding zone that are closer to the geocast region than the source. A gateway node that receives *X* tickets follows the same procedure as the one defined for the source. Consider the example in Figure 3.15 where node S begins with five (2+3) tickets. Node S may distribute two tickets to its neighboring nodes A and B, and one ticket to its neighbor node C, which are closer to the geocast region than node S. It is not mentioned in [Liao2000], however, why node C is given fewer tickets than nodes A and B. We believe the philosophy is that each ticket is responsible for carrying one copy of the geocast packet to the geocast region. Hence, if a node is sent a geocast packet that it has seen before, it does not discard it. For example, if node C decides to give its ticket to node B in Figure 3.15, (i.e., node B receives a geocast packet from node C), node B will rebroadcast the packet. In other words, node B will transmit the geocast packet (at least) two times.

Both the Flooding-Based GeoGRID and the Ticket-Based GeoGRID protocols need an efficient solution for the gateway election. Once this node is elected, it remains the gateway until it moves out of the grid. One problem of this selection process is when another potential gateway roams closer to the physical center of the grid than the currently assigned gateway and cannot be elected as the gateway until the current gateway leaves the grid. To eliminate this possibility, multiple gateways could be



A geocast example for the Ticket-Based GeoGRID protocol

allowed to reside in a grid temporally. In this situation, if a gateway hears a packet from another gateway at a location closer to the physical center of its grid, it silently turns itself into a non-gateway node and does not forward any further geocast packets. However, if the grid size is small, or the mobility of the node is low, this problem may not be severe. Another effective way of gateway election is via the concept of Node Weight [Basagnil999]. For example, we could assign the weight of a node as being inversely proportional to its speed. Flooding-Based GeoGRID and Ticket-Based GeoGRID have obvious advantages over LBM Scheme 1 and LBM Scheme 2, especially in dense networks. The two GeoGRID protocols should offer both higher accuracy and lower delivery cost than LBM and VDG due to the reduced number of transmitted packets.

GeoTORA

The goal of the GeoTORA protocol [Ko2000] is to reduce the overhead of transmitting geocast packets via flooding techniques, while maintaining high accuracy. The unicast routing protocol TORA is used by GeoTORA to transmit geocast packets to a geocast region. As TORA is a distributed routing protocol based on a "link reversal" algorithm, it provides multiple routes to a destination. Despite dynamic link failures, TORA attempts to maintain a destination-oriented directed acyclic graph such that each node can reach the destination. In GeoTORA, a source node essentially performs an *anycast* to any geocast group member (i.e, any node in the geocast region) via TORA. When a node in the geocast region receives the geocast packet, it floods the packet such that the flooding is limited to the geocast region. The accuracy of GeoTORA is high, but not as high as pure flooding or LBM. One reason is only one node in the geocast region receives the geocast packet and if that node is partitioned from other nodes in the geocast region, the accuracy reduces.

Mesh-Based Geocast Routing Protocol

The Mesh-based Geocast Routing (MGR) protocol [Boleng2001] uses a mesh for geocasting in an ad hoc environment in order to provide redundant paths between the source and the group members. Since the group members in a geocast region are in close proximity to each other, it is less costly to provide redundant paths from a source to a geocast region than to provide the redundant paths from a source to a multicast group of nodes that may not be in close proximity of each other. Instead of flooding geocast packets, the MGR Protocol tries to create redundant routes via control packets. First, the protocol floods JOIN-DEMAND packets in a forwarding zone. A JOIN-DEMAND packet is forwarded in the network until it reaches a node in the geocast region. This node unicasts a JOIN-TABLE packet back to the source by following the reverse route of the JOIN-DEMAND packet. Thus, the nodes on the edge of the geocast region become a part of the mesh. Once the first JOINTABLE packet is received by the source, data packets can be sent to the nodes in the geocast region. Figure shows an example of geocast communication via a mesh.

UNIT – III

Basics Of Wireless Sensors And Applications

THE MICA MOTE:

The Mica Mote [MICAwww] shown in Figure 8.2(a) is a comprehensive sensor node developed by University of California at Berkeley for this application and marketed by Crossbow. It uses an Atmel Atmega 103 microcontroller running at 4 MHz, with a radio operating at the 916 MHz frequency band which provides bidirectional com unication at 40 kbps. A pair of AA batteries provides the required energy. The Mica Board, shown in Figure 8.2(b), is stacked to the processor board via the 51 pin extension connector. It includes temperature, photo resistor, barometer, humidity, and thermopile sensors.

To conserve energy, later designs include an A/D Converter and an 8x8 power switch on the sensor board, the bypassing of the DC booster, etc. To protect sensors from the variable weather condition, the mica mote is packaged in an acrylic enclosure, which does not obstruct the sensing functionality and the radio communication. MICA 2 motes have three modes based RF frequency band, MPR400 using 915 MHz, MPR 410 employing 433 MHz and MPR420 based on 315 MHz. Three miniaturized sizes (1/4) of Mote are also available as MICA2DOT. Another version of MICA 2 using a combination of ultrasound and RF signal has been jointly developed by Crossbow and MIT and has a flexibility to configure either as a listener or a beacon transmitter. More details on Mica Motes and version 2 can be obtained from details of different types of commercially available sensor transducers could be obtained from many web sites.

Sensing and Communication Range

A wireless sensor network (WSN) consists of a large number of sensor nodes (SNs) and exploring their best possible use is a challenging problem. As the main objective of a SN is to monitor some physical quantity in a given area, the sensors need to be deployed with adequate density so that sensing of the complete area can be done, without leaving any void or unsensed area. In other words, given the sensing range of each SN and the area to be covered, adequate number of SNs should be needed to be placed throughout the area so that no corner is left out. The SNs can be placed deterministically at pre-specified locations or could be distributed randomly. So, if N SNs are put in an area A=LxL, then the SNs density can be given by $X — NIA$. The sensing range r_s of each sensor can be shown in Figure 8.3. As illustrated in Figure 8.3, each SN has its own sensing range and to cover the whole space, adjacent SNs need to be located close to each other and at the most at a distance of $2r_s$ from each other as illustrated by, three positions of SN_2, SN_2 and SN_2 with different overlapping areas between SN| and SN_2. If the SNs are unifonnly distributed with the node density of *X*, theprobability that there are m SNs within the space of S is Poisson distributed [Bettstetter2002] as m! Where space *S — 7Cr~* for two dimensional spaces. This gives the probability that the monitored space is not covered by any SN and hence the probability p_{covcr} of the coverage by at least one SN is:

$p_{aw,,}$=l-P(0) = l - ^

This above relation gives an idea about the coverage of the area so that adequate number of sensors could be deployed.

Design Issues

The advancement in technology has made it possible to have a network of 100s or even thousands of extremely small, low powered devices equipped with programmable computing, multiple parameter sensing and wireless communication capability, enhancing the reliability, accuracy of data and the coverage area. In short, some of the advantages of WSN over wired ones are as follows:

• Ease of deployment - These wireless sensors can be deployed (dropped from a plane or placed in a factory) at the site of interest without any prior organization, thus reducing the installation cost and time, and also increasing the flexibility of deployment;

• Extended range - One huge wired sensor (macro-sensor) can be replaced by many smaller wireless sensors for the same cost. Such a macro-sensor can sense only a limited region whereas a network of smaller sensors can be distributed over a wider range;

• Fault tolerant -With macro-sensors, the failure of one node makes that area completely unmonitored till it is replaced. With wireless sensors, failure of one node does not affect the network operation substantially as there are other adjacent nodes collecting similar data. At most, the accuracy of data collected may be somewhat reduced;

• Mobility - Since these wireless sensors are equipped with battery, they can possess limited mobility (e.g., if placed on robots). Thus, if a region becomes unmonitored we can have the nodes rearrange themselves to distribute evenly, i.e., these nodes can be made to move towards area of interest but having lower mobility as compared to ad hoc networks.

Traditional routing protocols defined for MANETs (discussed in previous chapters) are not well suited for wireless sensor networks due to the following reasons:

• As we mentioned earlier, w centric, routing to and from specific nodes in these networks is not required;

• Adjacent nodes may have similar data. So, rather than sending data separately from each sensor node to the requesting node, it is desirable to aggregate similar data before sending it;

• The requirements of the network change with the application and hence, it is application-specific. For example, in some applications, the sensor nodes are fixed and not mobile while others may need data based only on some selected attributes (viz., attribute is fixed in this network). ireless sensor networks are "data centric", where data is requested based on particular criteria such as "which area has temperature 35°C";

• In traditional wired and wireless networks, each node is given a unique identification (e.g., an IP address) used for routing. This cannot be effectively used in sensor networks because, being data.

Energy Consumption

Minimizing the energy consumption of WSs is critical yet a challenge for the design of WSNs. The energy consumption in WSN involves three different components: Sensing Unit (Sensing transducer and A/D Converter), Communication Unit (transmission and receiver radio), and Computing/Processing Unit. In order to conserve energy, we may make some SNs go to sleep mode and need to consider energy consumed in that state.

Sensing transducer is responsible for capturing the physical parameters of the environment. Its basic function is to do physical signal sampling and convert into electrical signals. The energy consumption of this part depends on the hardware and the application and sensing energy is only a small part of the total energy consumption.

A/D Converter: Based on paper [www.moteiv.com], an AD Converter for sensor consumes only 3.1 *JLlW* , in 31 *pJ/8-bit* sample at lVolt supply. The standby power consumption at IV supply is *4lpW.*Assuming the D/C is not noise limited, the lower bound on energy per sample for the successive approximation architecture is roughly computed as: E_{min}-$C_{,ot}aNref_{2}$, where C_{total} is the total capacitance of the array including the bottom plate parasites, and V_{ref} is a common mode input voltage the comparator works under.

Transmission Energy: Based on [Heinzelman2000] the transmission energy transmits a k-bit message to distance d can be computed as:$E_{Tx}(k,d)=E_{Tx.eiec}(k)+E_{Tx.amp}(k,d)=E_{etec}*k+ £ *k*d_2,$ where $Erx\text{-}eiec$ is the transmission electronics energy consumption, $E_{rx.amp}$ is the transmit amplifier energy consumption. Their model assumes $E_{Tx\_dec}=E_{Rx,elec}=E_{elec}=50nJ/bit,$ and $£_{amp}=\backslash 00pJ/bit/m_2$

Receiver Energy: To receive a k bit message, the energy consumed is $£_{fcW}=£_{,ftc.efc}/\&J=.£'_{e}fec*£$

Computation: The computing unit associated with a wireless sensor is a microcontroller/ processor with memory which can control and operate the sensing, computing and communication unit. The energy consumption of this unit has mainly two parts: switching energy and leakage energy. Switching energy is expressed as [Shih2001] $E_{sw}uch=C_{t0},aiV_{dd\ 2},$ where $C_{ma}i$ is the total capacitance switched by the computation and V_{dd} is the supply voltage. Dynamic voltage scaling (DVS) scheme is used to adaptively adjust operating voltage and frequency meet the dynamically changing workload without degrading performance thus saving energy.

Leakage energy is the energy consumed when no computation work is done. It can be expressed as: $E_{leakageup}$ = $(V'_{dd}i)I\backslash e_{nVr}$, where V_T is the thermal voltage, n' and I_0 are the parameters of processor and can get from experiment. For StrongARM SA-1100, n' =21.26 and $/_0$= 1.196mA

Clustering of Sensors:

From the preceding sections, it is clear that enough number of SNs need to be deployed if every corner of the area of interest; need to sensed for continuous monitoring and necessary action. In addition, successful transfer of sensed data to adjacent SNs necessitates minimum communication distance covered by the wireless radio to be at least twice that of sensing range. So, sensing distance and communication coverage are co-related. It is widely accepted that the energy consumed in one bit of data can be used to perform a large number of arithmetic operations in the sensor processor (power consumed in 1-bit transfer to 100m equals 3000 instructions [Pottie2000]). Moreover, the way sensors are deployed, physical environment would produce similar in close-by SNs and transmitting such data could be termed as more or less redundant.

Therefore, all these facts encourage using some kind of grouping (clustering) so that data from SNs belonging to a single cluster can be combined together in an intelligent way (aggregation) using local transmissions to transfer only the compact data. This can not only reduce the global data to be transferred and localized most traffic to within each individual cluster, but also reduces the traffic and hence contention in a WSN. A lot of research gone into testing coverage of areas by k-sensors (k>l) [Choi2004, Liu2004, Megerian2005, Zhang2005] clustering adjacent SNs and defining the size of the cluster [Chowdhury2005] so that the cluster heads (CHs) can easily get data from their own cluster members and CHs can also communicate among themselves and exchange data. If each cluster is covered by more than one subset of SNs all the time, then some of the SNs can be put into sleep mode so as to conserve energy while keeping full coverage of each cluster and the area. The use of a second smaller radio has been suggested for waking up the sleeping sensor [Miller2005], thereby conserving the power of main wireless transmitter. In this section, we consider cluster formation in two types of WSNs, one place in a predetermined grid form and another, when SNs are placed randomly within a given area.

Applications:

Thousands of sensors over strategic locations are used in a structure such as an automobile or an airplane, so that conditions can be constantly monitored both from the inside and the outside and a real-time warning can be issued whenever a major problem is forthcoming in the monitored entity. These wired sensors are large (and expensive) to cover as much area is desirable. Each of these need a continuous power supply and communicates their data to the end-user using a wired network. The organization of such a network should be pre-planned to find strategic position to place these nodes and then should be installed appropriately. The failure of a single node might bring down the whole network or leave that region completely un-monitored. Unattendability and some degree of fault tolerance in these networks are especially desirable in those applications where the sensors may be embedded in the structure or places in an inhospitable terrain and could be inaccessible for any service. Undoubtedly, wireless sensor networks have been conceived with military applications in mind, including battlefield surveillance and tracking of enemy activities. However, civil applications considerably outnumber the military ones and are applicable to many practical situations

Classifications of WSNs:

A WSN is deployed primarily to collect sensed data by different WSs and it is critical to see how requently the sensed values are collected. Looking at various ways in which one can employ the network resources, WSNs can be classified on the basis of their mode of operation or functionality, and the type of target applications.

Accordingly, we hereby classify WSNs into three types:

• **Proactive Networks** - The nodes in this network periodically switch on their sensors and transmitters, Sense the environment and transmit the data of interest. Thus, they provide a snapshot of the relevant Parameters at regular intervals and are well suited for applications requiring periodic data monitoring.

• **Reactive Networks** - In this scheme, the nodes react immediately to sudden and drastic changes in the value of a sensed attribute. As such, these are well suited for time critical applications.

• **Hybrid Networks** - This is a combination of both proactive and reactive networks where sensor node not only send data periodically, but also respond to sudden changes in attribute values.

Once the type of network is decided, protocols that efficiently route data from the SNs to the users have to be designed, perhaps using a suitable MAC protocol to avoid collisions and subsequent energy consumption. Attempts should be made to distribute energy dissipation evenly among all nodes in the network, as it is usually not common to assume the presence of specialized high-energy nodes in the network. In this chapter we cover proactive, reactive and hybrid protocols, while highlighting the fact that the protocols ought to be directly related to application requirements.

Architecture of Sensor Networks:

Due to the principle differences in application scenarios and underlying communication technology, the architecture of WSNs will be drastically different both with respect to a single WS and the network as a whole. The typical hardware platform of a wireless sensor node will consist of:

• Simple embedded microcontrollers, such as the Atmel or the Texas Instruments MSP 430. A decisive characteristic here is, apart from the critical power consumption, an answer to the important question whether and how these microcontrollers can be put into various operational and sleep modes, how many of these sleep modes exist, how long it takes and how much energy it costs to switch between these modes. Also, the required chip size and computational power and on-chip memory are important

• Currently used radio transceivers include the RFM TR1001 or Infineon or Chip on devices; similar radio modems are available from various manufacturers. Typically, ASK or FSK is used, while the Berkeley Pico Nodes employ OOK modulation. Radio concepts like ultra-wideband are in an advanced stage (e.g., the projects undertaken by the IEEE 802.15 working group). A crucial step forward would be the introduction of a reasonably working wake-up radio concept, which could either wake up all SNs in the vicinity of a sender or even only some directly addressed nodes. A wake-up radio allows a SN to sleep and to be wakened up by suitable transmissions from other nodes, using only a low-power detection circuit. Transmission media other than radio communication are also considered, e.g., optical communication or ultra-sound for underwater-applications. However, this largely depends on the application

• Batteries provide the required energy. An important concern is battery management and whether and how energy scavenging can be done to recharge batteries in the field. Also, self-discharge rates, self recharge rates and lifetime of batteries can be an issue, depending on the application;

• The operating system and the run-time environment is a hotly debated issue in the literature. On one hand, minimal memory footprint and execution overhead are required while on the other, flexible means of combining protocol building blocks are necessary, as meta information has to be used in many places in a protocol stack (e.g., information about location, received signal strength, etc., has an influence on many different protocol functions). Consequently, we believe that structures like blackboards, publish/subscribe or tuple spaces are an interesting starting point for the run-time environments for such SNs.

MAC Layer:

The MAC and the routing layers are the most active research areas in WSNs. Therefore, an exhaustive discussion of all schemes is impossible. However, most of the existing work addresses how to make SNs sleep as long as possible. Consequently, these proposals often tend to include at least some aspects of TDMA. The wireless channel is primarily a broadcast medium. All nodes within radio range of a node can hear its transmission. This can be used as a uni cast medium by specifically addressing a particular node and all other nodes can drop the packet they receive. There are two types of schemes available to allocate a single broadcast channel among competing nodes: Static Channel Allocation and Dynamic Channel Allocation.

• Static Channel Allocation: In this category of protocols, if there are N SNs, the bandwidth is divided

In to N equal portions in frequency (FDMA), in time (TDMA), in code (CDMA), in space (SDMA) or In schemes such as OFDM or are only a small and fixed number of SNs, each of which has Buffered (heavy) load of data

• Dynamic Channel Allocation: In this category of protocols, there is no fixed assignment of bandwidth. When the number of active SNs changes dynamically and data becomes burst at arbitrary SNs, it is most advisable to use dynamic channel allocation scheme. These are contention-based schemes, where SNs contend for the channel when they have data while minimizing collisions with other SNs transmissions. When there is a collision, the SNs are forced to retransmit data, thus leading to increased wastage of energy and unbounded delay.

As we will see shortly, in a hierarchical clustering model, once clusters have been formed, it is desirable to keep the number of nodes in the cluster fixed and due to hierarchical clustering; the number of nodes per cluster is not kept large. So, it may be better to use one of the static channel allocation schemes. In his scheme, each node transmits data in its own slot to the cluster head and at all other times, its radio can be switched off, thereby saving valuable energy.

When it is not feasible to use TDMA, the n des can use non persistent CSMA since the data packets are of fixed size.TDMA is suitable for either proactive or reactive type of networks. In proactive networks, as we have the nodes transmitting periodically, we can assign each node a slot and thus avoid collisions. In reactive Networks, since adjacent nodes have similar data, when a sudden change takes place in some attribute being sensed, all the nodes will respond immediately. This will lead to collisions and it is possible that the data may never reach the user on time. For this reason, TDMA is employed so that each node is given a slot and they transmit only in that slot. Even though this increases the delay and many slots might be empty, it is better than the energy consumption incurred due to dynamic channel allocation schemes.

Design Issues:

As with MAC protocols for traditional MANETs, WSNs have their own inherent characteristics that need to be addressed. Below we discuss some of the most important ones involved in the design of MAC protocols for WSNs.

Coping up with Node Failure:

When many SNs have failed, the MAC and routing protocols must accommodate formation of new links and routes to other SNs and the BS. This may require dynamically adjusting transmit powers and signaling rates on the existing links, or rerouting packets through regions of the network with higher energy level.

Sources of Resource Consumption at the MAC Layer:

There are several aspects of a traditional MAC protocol that have negative impact on wireless sensor networks including:

• Collisions - When a transmitted packet is corrupted due to a collision, it has to be discarded. The follow-on retransmission increases the energy consumption and hence increases the latency

• Overhearing - SNs listen to transmissions that are destined to other SNs

• Control packets overhead - Sending and receiving control packets consume energy and reduce the payload. This overhead increases linearly with node density. Moreover, as more SNs fail in the network, more control messages are required to self configure the system, resulting in more energy consumption

• Idle Listening -Waiting to receive anticipated traffic that is never sent. This is especially true in many sensor network applications. If nothing is sensed, SNs are in the idle mode for most of the time.

Measures to Reduce Energy Consumption:

One of the most cited methods to conserve energy in sensor networks is to avoid listening to idle channels, that is, neighboring nodes periodically sleep (radio off) and auto synchronize as per sleep schedule. It is important to note that fairness, latency, throughput and bandwidth utilization are secondary in the WSNs.

Comparison of Scheduling & Reservation-based andContention-based MAC Design:

One approach of MAC design for WSNs is based on reservation and scheduling, for example TDMA-based protocols that conserve more energy as compared to contention-based protocols like the IEEE 802.11 DCF. This is because the duty cycle of the radio is increased and there is no contention-introduced overhead and collisions. However, formation of cluster, management of inter-cluster communication, and dynamic adaptation of the TDMA protocol to variation in the number of nodes in the cluster in terms of its frame length and time slot assignment are still the key challenges.

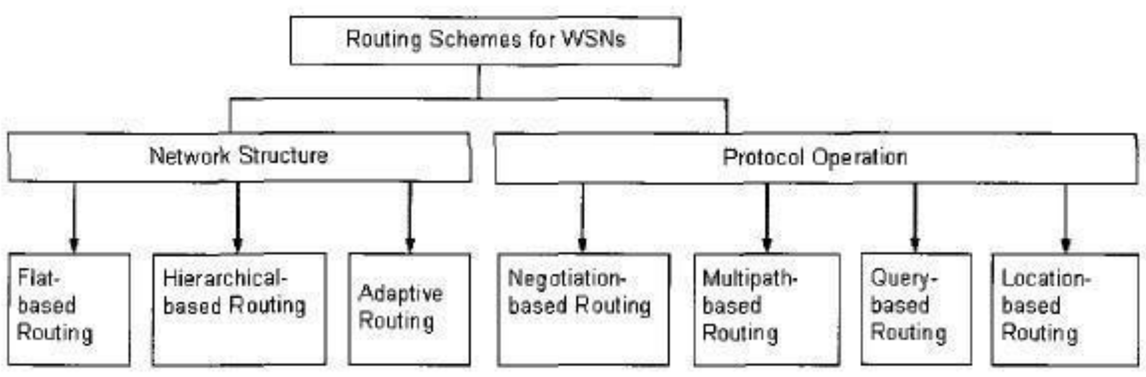
MAC Protocols:

WSNs are designed to operate for long time as it is rather impractical to replenish the batteries. However, nodes are in idle state for most time when no sensing occurs. Measurements have shown that a typical radio consumes the similar level of energy in idle mode as in receiving mode. Therefore, it is important that nodes are able to operate in low duty cycles. As far as the MAC layer is concerned, some of the more recent and relevant studies in this area are Pico Radio MAC, the S-MAC, the SMACS and the STEM As many of these protocols share common characteristics; in this section we discuss only those which are most prominent and necessary to understand the others. It is also important to note that more traditional MAC schemes such as FDMA, TDMA, CDMA, SDMA and a combination of these can also be employed. However, as these techniques are widely known, we do not discuss them.

Routing Layer:

By now, it must be clear that WSNs differ from traditional wireless networks. Conventional flooding-based protocols widely employed in MANETs suffer from data explosion problem, i.e., if a node is a common neighbor to nodes holding the same data item, then it will get multiple copies of the same data item. Therefore, the protocol wastes resources by sending and receiving duplicate data copies. In addition, flooding does not scale well in large networks and wastes resources.

The goal is to send the data from source node(s) to a known destination node, i.e., the BS. The destination node or the sink node is known and addressed by means of its location. A BS may be fixed or mobile, and is capable of connecting the sensor network to an existing infrastructure (e.g., Internet) where the user can have access to the collected data. The task of finding and maintaining routes WSNs is nontrivial since energy restrictions and sudden changes in node status (e.g., failure) cause frequent unpredictable topological changes. Thus, the main objective of routing techniques is to minimize the energy consumption in order to prolong WSN lifetime. To achieve this objective, routing protocols proposed in the literature employ some well known routing techniques as well as tactics special to WSNs. To preserve energy, strategies like data aggregation and in-network processing, clustering, different node role assignment, and data-centric methods are employed.



Classification of routing protocols for WSNs

In sensor networks, conservation of energy is considered relatively more important than quality of data sent. Therefore, energy-aware routing protocols need to satisfy this requirement. Routing protocols for WSNs have been extensively studied in the last few years. Routing protocols for WSNs can be broadly classified into flat-based, hierarchical-based, and adaptive, depending on the network structure. In

Flat-based routing, all nodes are assigned equal role. In hierarchical based routing, however, nodes play different roles and certain nodes, called cluster heads (CHs), are given more responsibility. In adaptive routing, certain system parameters are controlled in order to adapt to the current network conditions and available energy levels. Furthermore, these protocols can be classified into multipath-based, query-based, negotiation-based, or location-based routing techniques. In this section we use a classification according to the network structure and protocol operation (i.e., routing criteria), and is shown in Figure 9.8. In majority of applications, sensor nodes are expected to be stationary. Thus, it may be preferable to have table driven routing protocols rather than employing reactive schemes where a significant amount of energy is used in route discovery and setup. Another class of routing protocols is called the cooperative routing protocols. Where in SNs send data to a CH where data can be aggregated and may be subjected to further processing, hence reducing route cost in terms of energy use. Several other protocols, in turn, rely on timing and position information and are also covered in this section.

Network Structure Based

In this class of routing protocols, the network structure is one of the determinant factors. In addition, the network structure can be further subdivided into flat, hierarchical and adaptive depending upon its organization.

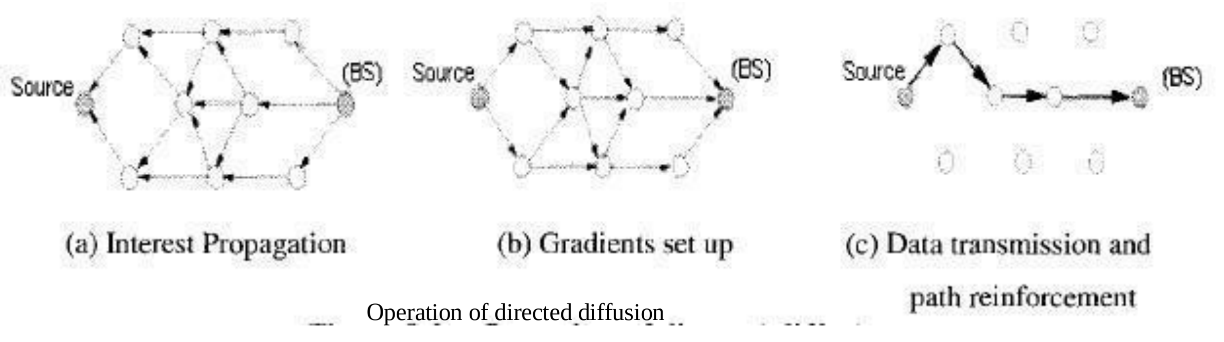
Flat Routing

In flat routing based protocols, all nodes play the same role. Here, we present the most prominent protocols falling in this category.

Directed Diffusion

Directed Diffusion [Intanagonwiwat2000] is a data aggregation and dissemination paradigm for sensor networks. It is a data-centric (DC) and application-aware approach in the sense that all data generated by sensor nodes is named by attribute-value pairs. Directed Diffusion is very useful for applications requiring dissemination and processing of queries. The main idea of the DC paradigm is to combine the data coming from different sources en-route (in-network aggregation) by eliminating redundancy, minimizing the number of transmissions; thus saving network energy and prolonging its lifetime. Unlike traditional end-to-end routing, DC routing finds routes from multiple sources to a single destination (BS) that allows in-network consolidation of redundant data. In Directed Diffusion, sensors measure events and create gradients of information in their respective neighborhoods. The BS requests data by broadcasting *interests*, which describes a task to be done by the network.

Interest diffuses through the network hop-by-hop, and is broadcast by each node to its neighbors. As the interest is propagated throughout the network, gradients are setup to draw data satisfying the query towards the requesting node. Each SN that receives the interest setup a gradient toward the SNs from which it receives the interest. This process continues until gradients are setup from the sources back to the BS. The strength of the gradient may be different towards different neighbors, resulting in variable amounts of information flow. At this point, loops are not checked, but are removed at a later stage. Figure 9.9 depicts an example of the operation of directed diffusion. Figure 9.9(a) presents the propagation of interests, Figure 9.9(b) shows the gradients construction, and Figure 9.9(c) depicts the data dissemination. When interests fit gradients, paths of information flow are formed from multiple paths, and the best paths are reinforced so as to prevent further flooding according to a local rule. In order to reduce ommunication costs, data is aggregated on the way. The BS periodically refreshes and re-sends the interest when it starts to receive data from the source(s). This retransmission of interests is needed because the medium is Inherently unreliable.



Operation of directed diffusion

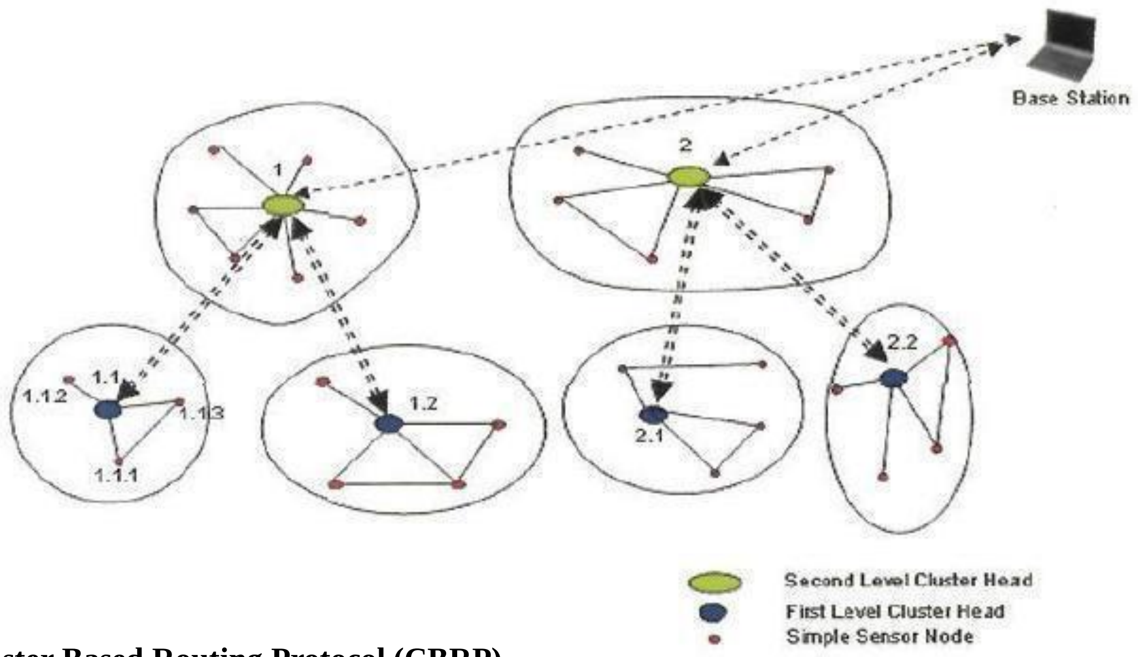
Sensor nodes in a directed diffusion-based network are application aware, which enables diffusion to achieve energy savings by choosing empirically good paths and by caching and processing data in the network. An application of directed diffusion is to spontaneously propagate an important event to regions of the sensor network. Such type of information retrieval is well suited for persistent queries where requesting nodes expect data that satisfy a query for a period of time. However, it may be unsuitable for historical or one-time queries as it is not worth setting up gradients which employ the path only once. The performance of data aggregation methods employed by the directed diffusion paradigm is affected by the location of the source nodes in the network, the number of sources, and the network topology. In order to investigate these factors, two models of source placement shown in Figure 9.10 have been investigated. These models are called the event radius (ER) model, and the random sources (RS) model. In the ER model, a single point in the network area is defined as the location of an event. For example, this may correspond to a vehicle or some other phenomenon being tracked by the sensor nodes. All nodes within a distance S (called the sensing range) of this event that are not sinks, are considered to be data sources. In the RS model, K nodes that are not sinks are randomly selected to be sources. Unlike the ER model, in the RS model the sources are not necessarily closed to each other. In both of these source placement models, however, for a given energy budget, a greater number of sources can be connected to the sink. Therefore, the energy savings with aggregation in directed diffusion can be transformed to provide a greater degree of robustness as per dynamics of the sensed activity.

Sequential Assignment Routing (SAR)

The routing scheme in SAR [Sohrabi2000] is dependent on three factors: energy resources, QoS on each path, and the priority level of each packet. To avoid single route failure, a multi-path approach coupled with a localized path restoration scheme is employed. To create multiple paths from a source node, a tree rooted at the source node to the destination nodes (i.e., the set of BSs) is constructed. The paths of the tree are defined by avoiding nodes with low energy or QoS guarantees. At the end of this process, each sensor node is part of multi-path tree. For each SN, two metrics are associated with each path: delay (which is an additive QoS metric); and energy usage for routing on that path. The energy is measured with respect to how many packets will traverse that path. SAR calculates a weighted QoS metric as the product of the additive QoS metric and a weight coefficient associated with the priority level of the packet. The goal of SAR is to minimize the average weighted QoS metric throughout the lifetime of the network. Also, a path recomputation is carried out if the topology changes due to node failures. As a preventive measure, a periodic re-computation of paths is triggered by the BS to account for any changes in the topology. In addition, a handshake procedure based on a local path restoration scheme between neighboring nodes is used to recover from a failure.

Hierarchical Routing

Hierarchical, or cluster-based, routing has its roots in wired networks, where the main goals are to achieve scalable and efficient communication. As such, the concept of hierarchical routing has also been employed in WSN to perform energy-efficient routing. In a hierarchical architecture, higher energy nodes (usually called cluster heads) can be used to process and send the accumulated information while low energy nodes can be used to sense in the neighborhood of the target and pass on to the CH. In these cluster-based architectures, creation of clusters and appropriate assignment of special tasks to CHs can contribute to overall system scalability, lifetime, and energy efficiency. An example of a general hierarchical clustering scheme is depicted in Figure 9.11. As we can see from this figure, each cluster has a CH which collects data from its cluster members, aggregates it and sends it to the BS or an upper level CH. For example in Figure 9.11, nodes 1.1.1, 1.1.2, 1.1.3 and 1.1 form a cluster with node 1.1 as the CH. Similarly, there exist other CHs such as 1.2, etc. These CHs, in turn, form a cluster with node 1 as their CH. So, node 1 becomes a second level CH as well. This pattern is repeated to form a hierarchy of clusters with the uppermost level cluster nodes reporting directly to the BS. The BS forms the root of this hierarchy and supervises the entire network.



Hierarchical Clustering

Cluster Based Routing Protocol (CBRP)

A simple cluster based routing protocol (CBRP) has been proposed in [Jiang 1998]. It divides the network nodes into a number of overlapping or disjoint two-hop-diameter clusters in a distributed manner. Here, the cluster members just send the data to the CH, and the CH is responsible for routing the data to the destination. The major drawback with CBRP is that it requires a lot of hello messages to form and maintain the clusters, and thus may not be suitable for WSN. Given that sensor nodes are stationary in most of the applications this is a considerable and unnecessary overhead.

Scalable Coordination

In [Estrinl999], a hierarchical clustering method is discussed, with emphasis on localized behavior and the need for asymmetric communication and energy conservation in a sensor network. In this method the cluster formation appears to require considerable amount of energy (no experimental results are available) as periodic advertisements are needed to form the hierarchy. Also, any changes in the network conditions or sensor energy level result in re-clustering which may be not quite acceptable as some parameters tend to change dynamically.

Low-Energy Adaptive Clustering Hierarchy (LEACH)

LEACH is introduced in [Heinzelman2000b] as a hierarchical clustering algorithm for sensor networks, called Low-Energy Adaptive Clustering Hierarchy (LEACH). LEACH is a good approximation of aproactive network protocol, with some minor differences which includes a distributed cluster formation algorithm. LEACH randomly selects a few sensor nodes as CHs and rotates this role amongst the cluster members so as to evenly distribute the energy dissipation across the cluster. In LEACH, the CH nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the BS in order to reduce the amount of information that must be transmitted. LEACH uses a TDMA and CDMA MAC to reduce intra-cluster and inter-cluster collisions, respectively. However, data collection is centralized and is performed periodically. Therefore, this protocol is better appropriate when there is a need for constant monitoring by the sensor network. On the other hand, a user may not need all the dataimmediately. Hence, periodic data transmissions may become unnecessary as they may drain the limited energy of the sensor nodes. After a given interval of time, a randomized rotation of the role of the CH is conducted so that uniform energy dissipation in the sensor network is obtained. Based on simulation, it has been found that only 5% of the nodes actually need to act as CHs.

The operation of LEACH is separated into two phases, the setup phase and the steady state phase. In the setup phase, the clusters are organized and CHs are selected. In the steady state phase, actual data transfer to the BS takes place. Clearly, the duration of the steady state phase is longer than the duration of the setup phase in order to minimize overhead. During the setup phase, a predetermined fraction of nodes, say *p*, elect themselves as CHs as follows. A sensor node chooses a random number, say r, between 0 and 1. If this random number is less than a threshold value, say *T(n)*, the node becomes a CH for the current round.

The threshold value, in turn, is calculated based on an equation that incorporates the desired percentage to become a CH, the current round, and the set of nodes that have not been selected as a CH in the last *{lip)*rounds, denoted by *G.* As a result, *T(n)* is given by:

T(n) = - ifneG

I- p(rmod(l/ p))

where G is the set of nodes that are involved in the CH election. Each elected CH broadcast an advertisement message to the rest of the nodes in the network, informing that they are the new CHs. All the non- CH nodes, after receiving this advertisement, decide on the cluster to which they want to attach to. In LEACH, this decision is based on the signal strength of the advertisement. The non-CH nodes then inform the corresponding CH of their decision to be a member of its cluster. Based on the number of nodes in the cluster, the CH node creates a TDMA schedule and assigns each node a time slot within this schedule where it can transmit. This schedule is then broadcast to all cluster members. During the steady state phase, sensor nodes begin sensing and transmitting data to their respective CHs. Once the CH receives the data from all of its members, it aggregates before relaying data to the BS. After a period time, which is determined a priori, the network goes back into the setup phase and initiates another round for selecting new CHs. Although LEACH is able to increase the network lifetime, there are still a number of issues regarding many assumptions. For example, LEACH assumes that all nodes can transmit with enough power to reach the BS if needed, and that every node has enough computational power to support different MAC protocols. It also assumes that nodes always have data to send, and nodes located close to each other have correlated data. Also, it is not obvious how the number of the predetermined CHs (p) is going to be uniformly distributed through the network. Therefore, there is the possibility that the elected CHs be concentrated in one part of the network. Thus, some nodes will not at all find CHs in their proximity. Finally, the protocol assumes that all nodes begin with the same amount of energy capacity in each election round.

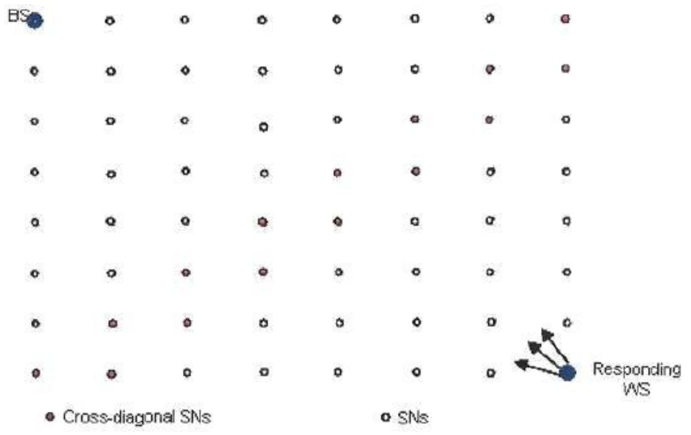
LEACH could be extended to account for non-uniform energy nodes, i.e., to use energy-based threshold. An extension to LEACH, also known as LEACH with negotiation, has been introduced in [Heinzelman2000b] with the goal of preceding data transfers with negotiations similar to meta-data descriptors used in the SPIN protocol discussed later. This ensures that only data that actually provides new information is transmitted to the CHs.

Multipath-Based Routing

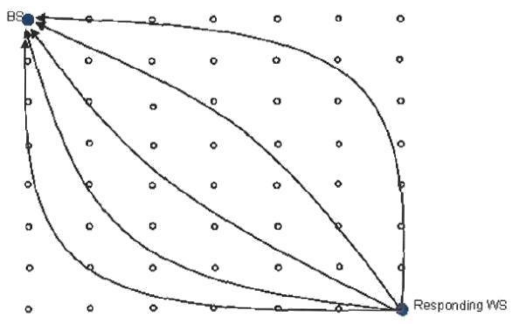
Network performance, and possibly lifetime, in WSNs can be significantly improved if the routing rotocol is able to maintain multiple, instead of a single, paths to a destination, and protocols in this class are called multipath protocols. By employing multipath protocols, the fault tolerance (resilience) of the network is considerably increased. The fault tolerance of a protocol is measured by the likelihood that an alternate path exists between a source and a destination when the primary path fails. Clearly, this can be increased if we maintain multiple paths between the source and the destination at the expense of an increased energy consumption and traffic generation (i.e., overhead), as alternate paths are kept alive by sending periodic messages. We would also like to note here that multipath routes between a source and a destination can be or not node-disjoint. Multiple paths between a source and destination are said to be node-disjoint when there is no node overlap amongst them. For the purpose of our discussion here, we refer to alternate routes as not being node disjoint, i.e., their routes are partially overlapped. In addition, unless otherwise noted, all multiple paths are of alternate types in this section.

A set of suboptimal paths can be occasionally employed as to increases the network lifetime [Rahul2002]. These suboptimal paths are chosen by means of a probability which depends on how low the energy consumption of each path is. Packets are routed through a path with largest residual energy in the algorithm proposed in [Chang2000]. Here, the path is changed whenever a better path is discovered. The primary path is used until its energy falls below the energy of the backup path, at which time the backup path is used. By employing this mechanism, the nodes in the primary path do not deplete their energy resources through continual use of the same route, hence prolonging their lifetime. One issue with this scheme is the cost associated with switching paths, and how to deal with packets which are en-route. As we have seen before, there is a tradeoff between minimizing the total power consumed and the residual energy of a network. As a result, routing packets through paths with largest residual energy may turn out to be very energy-expensive.

To minimize this effect, it has been proposed in [Li2001b] a scheme in which the residual energy of the route is relaxed a bit in order to select a more energy efficient path. Multipath routing was employed in [Dulman2003] to enhance the reliability of WSNs. This scheme is useful for delivering data in unreliable environments. Here, reliability is enhanced by providing several paths from source to destination and by sending the same packet thorough each and every path. Obviously, by using this technique, traffic increases significantly. Therefore, there is a tradeoff between the amount of traffic and the network reliability. This tradeoff is investigated in [Dulman2003] using a redundancy function that is dependent on the multipath degree and on failing probabilities of the available paths.



(a) Routing Region and Cross-diagonal SNs



(b) Paths Through cross-diagonal SNs

Determination of Multiple Paths

Directed Diffusion (discussed earlier). Based on the directed diffusion paradigm, a multipath routing scheme that finds several partially disjoint paths is presented in [Ganesan2001j. The idea is that the use of multipath routing provides a viable alternative for energy efficient recovery from failures in WSN. The motivation of using these braided paths is to keep the cost of maintaining the multiple paths low. In this scheme, the costs of alternate paths are comparable to the primary path as they tend to be much closer to the primary path.

High-Level Application Layer Support:

The protocols we have presented so far are also found, albeit in some different form in traditional wired, cellular, or ad hoc networks. For specific applications, a higher level of abstraction specifically tailored to WSN appears to be useful. In this section, we outline some of the activities in this direction.

Distributed Query Processing:

The number of messages generated in distributed query processing is several magnitudes less than in centralized scheme. [Bonnet2000, Bonnet2001], discusses the application of distributed query execution techniques to improve communication efficiency in sensor and device networks. They discuss two approaches for processing sensor queries: warehousing and distributed. In the warehousing approach, data is extracted in a pre-defined manner and stored in a central database (e.g., the BS). Subsequently, query processing takes place on the BS. In the distributed approach, only relevant data is extracted from the sensor network, when and where it is needed. A language similar to the Structured Query Language (SQL) has been proposed in [Madden2003] for query processing in homogeneous sensor networks. They have developed a suite of techniques for power-based query optimization and built a prototype instantiation for the same, called TinyDB, which runs on Berkeley mica motes [MICAwww].

Sensor Databases:

One can view the wireless sensor network as a comprehensive distributed database and interact with it via database queries. This approach solves, en passant, the entire problem of service definition and interfaces to WSNs by mandating, for example, SQL queries as the interface. The problems encountered here are in finding energyefficiency ways of executing such queries and of defining proper query languages that can express the full richness of WSNs. The TinyDB project [TinyDBwww] carried out at the University of California at Berkeley is looking at these issues. A model for sensor database systems known as COUGAR [COUGARwww] defines appropriate user and internal representation of queries. The sensor queries are also considered so that it is easier to aggregate the data and to combine two or more queries. In COUGAR, routing of queries is not handled. COUGAR has three-tier architecture.

• The Query Proxy: A small database component running on the sensor nodes to interpret and execute queries

• A Front end Component: A query-proxy that allows the sensor network to connect to the outside world. Each front-end includes a full-fledged database server

• A Graphical User Interface (GUI): Through the GUI, users can pose ad hoc and long running queries on the WSN. A map that allows the user to query by region and visualize the topology of sensors in the network.

Distributed Algorithms:

WSNs are not only concerned with merely *sensing* the environment but also with interacting with the environment. Once actuators like valves are added to WSNs, the question of distributed algorithms becomes inevitable. One showcase is the question of distributed consensus, where several actuators have to reach a joint decision (a functionality which is also required for distributed software update, for example). This problem has been investigated to some degree for ad hoc networks [Malpani2000, Nakano2002, Srinivasan2003, Walter2001], but it has not been fully addressed in the context of WSNs where new scalability and reliability issues emerge and where the integration in the underlying, possibly data-centric routing architecture, has not yet been investigated.

Adapting to the Inherent Dynamic Nature of WSNs:

Some important goals that current research in this area is aiming to achieve are as follows:

• Exploit spatial diversity and density of sensor/actuator nodes to build an adaptive node sleep schedule

• Spontaneously create and assemble network, dynamically adapt to device failure and degradation, manage mobility of sensor nodes and react to changes in task and sensor requirements
• Adaptability to drastic changes in the traffic
• Having finer control over the precision and coverage.

The Scalable Coordination Architectures for Deeply Distributed Systems (SCADDS) project SCADDS www], also a part of DARPA Sens IT program [Sens IT www], focuses on adaptive fidelity, dynamically adjusting the overall fidelity of sensing in response to task dynamics (turn on more sensors when a threat is perceived). They use additional sensors (redundancy) to extend lifetime. Neighboring nodes are free to talk to each other irrespective of their listening schedules; there is no clustering and no inter-cluster communication and interference. Adaptive Self-Configuring Ensor Network Topologies (Ascent) [Cerpa2001b], which is part of SCADDS, focuses on how to decide which nodes should join the routing infrastructure to adapt to a wide variety of environmental dynamics and terrain conditions producing regions with non uniform communication density. A node signals and reduces its duty cycle when it detects high message loss, requesting additional nodes in the region to join the network in order to relay messages to it. It probes the local communication environment and does not join the multi-hop routing infrastructure until it is helpful to do so. In addition, it avoids transmitting dynamic state information repeatedly across the network.

UNIT-IV
Security

Introduction

As we have seen in the previous chapters, the advent of ad hoc networks brought with it a flurry of research primarily focused on communication and protocols in every layer of the protocol stack.Practical applications of this research range from simple chat programs to shared whiteboards and other collaborative schemes. Although intended for diverse audiences and contexts, many of these applications Share a common characteristic: they are information-centric. The information transferred may be a trivial conversation between friends, confidential meeting notes shared among corporate executives, or mission- critical military information. Despite the deployment of information-driven applications such as these, the call for ad hoc and sensor network security remains largely unanswered. Ad hoc security is not, however, a concern that has slipped through the cracks unnoticed: numerous research initiatives have been launched to surmount the challenge. Despite that, as we shall see, many questions remain open as many of the existing approaches have limited functionalities, unrealistic computational requirements, or inability to address core security issues.

Security in ad hoc networks is an essential component for basic network functions like packet forwarding and routing and network operation can be easily jeopardized if countermeasures are not embedded into the basic network functions at the early stages of their design. In ad hoc and sensor networks, the basic functions are carried out by all available nodes. This very difference is at the core of the security problems that are specific to these networks. If *a* priori trust relationship exists between the nodes of an ad hoc network, entity authentication can be sufficient to assure the correct execution of critical network functions. A priori trust can only exist in a few special scenarios like military networks and corporate networks, where a common, trusted authority manages the network, and requires tamper-proof hardware for the implementation of critical functions. An environment where a common, trusted authority exists is called a *managed environment.* On the other hand, entity authentication in a large network raises key management requirements. When tamper-proof hardware and strong authentication infrastructure are not available, like for example in an *open environment* where a common authority that regulates the network does not exist, any node of an ad hoc network can endanger the reliability of basic network functions. The correct operation of the network also requires fair share of the functions by each participating node as power saving is a major concern. The considered threats are thus not just limited to maliciousness, and a new type of misbehavior called selfishness should also be taken into account to prevent nodes that simply do not cooperate.

With the *lack of a priori trust,* a classical network security mechanism based on authentication and access control cannot cope up with selfishness. Cooperative security schemes seem to offer the only reasonable solution wherein node misbehavior can be detected through the collaboration between a numbers of nodes, assuming that a majority of nodes do not misbehave. Therefore, security in ad hoc networks is a much harder task than in traditional wired networks and we need to first understand the differences between security in centralized and distributed systems. Then, we delve into the specifics of security over ad hoc and sensor networks including key management schemes, secure routing algorithms, cooperation, and intrusion detection systems.

Distributed Systems Security

According to the most widely used categorization of threats to computer systems, the threats are divided into three categories [Zwicky2000]: disclosure threats, integrity threats and denial of service threats. This by no means covers all the possible threats, but will suffice for our purposes. The disclosure threat involves the leakage of information from the system to a party that should not have seen the information and is a threat against the confidentiality of the information. The integrity threat involves an unauthorized modification of information. Finally, the denial of service threat involves inability to access a system resource that is being blocked by a malicious attacker.

Security in Ad Hoc Networks

As we know, there is no fixed infrastructure in ad hoc networks and as the name implies they are formed on the fly. The devices connect to each other in their own communication range via wireless links.

Individual devices act as routers when relaying messages to other distant devices. The topology of an ad hoc network is not fixed either. It changes all the time when these mobile stations move in and out of each others transmission range. All this makes ad hoc networks very vulnerable to attacks and the security issues become very complex. In this section we give an overview of the security issues over ad hoc and sensor networks.

Requirements

The security services of ad hoc networks are not altogether different than those of other network communication paradigms. Below we describe the requirements ad hoc networks must meet.

Availability

Availability ensures that the desired network services are available whenever they are needed. Systems that ensure availability seek to combat denial of service (DoS) and energy starvation attacks. In ad hoc networks, ensuring availability is perhaps more important than it is in traditional Internet. As all the devices in the network depend on each other to relay messages, DoS attacks are easy to perpetrate. For example, a malicious user could try to jam or otherwise try to interfere with the flow of information. Or else, the routing protocol should be able to malicious users by feeding the network with accurate information. There are routing protocols that can adjust well to the changing topology, but there are none that can defy all the possible attacks [Deng2002, Zhou 1999]. Another vulnerable point, which has no equivalence in traditional networks, is the limited battery power of wireless nodes. Normally, these nodes try to save energy with power saving schemes, so that when the device is not in active use, energy is not consumed. With battery exhaustion attacks, a malicious user can cause higher power consumption from other devices' battery, causing these devices to die prematurely [Stajanol999].

Authorization and Key Management

Authorization is another difficult matter in ad hoc networks. As there is little or no infrastructure, identifying users (e.g., participants in a meeting room) is not an easy task. There are problems with rusted third party schemes and identity-based mechanisms for key agreement. A generic protocol for password authenticated key exchange is described in [Asokan2000]. It has several drawbacks even though it is possible to construct very good authentication mechanisms for ad hoc networks. A password uthenticated multi-party Diffie - Hellman key exchange seems to overcome many problems of the generic protocol.

Confidentiality and Integrity

Data confidentiality is a core security primitive for ad hoc networks. It ensures that the message cannot be understood by anyone other than the authorized personnel. With wireless communication, anyone can sniff the messages going through the air, and without proper encryption all the information is easily available. On the other hand, without proper authentication, it is difficult to enforce confidentiality. And if the proper authenticity has been established, securing the connection with appropriate keys does not pose a big problem. Data integrity denotes the immaculateness of data sent from one node to another. That is, it ensures that a message sent from node A to node B was not modified during transmission by a malicious node C. If a robust confidentiality mechanism is employed, ensuring data integrity may be as simple as adding one-way hashes to encrypted messages. In addition to malicious 520 *AD* handle both the changing topology of the network and attacks from the attacks, integrity may be compromised because of radio interference, etc., so some kind of integrity protection is definitely needed for ad hoc networks.

Non-Repudiation

Non-repudiation ensures that the origin of a message cannot deny having sent the message. It is useful for detection and isolation of compromised nodes. When a node A receives an erroneous message from a node B, non-repudiation allows A to accuse B using this message and to convince other nodes that B is compromised.

Security Solutions Constraints

Historically, network security personnel have adopted a centralized, largely protective paradigm to satisfy aforementioned requirements. This is effective because the privileges of every node in the network are managed by dedicated machines - authentication servers, firewalls, etc. - and the professionals who maintain them. Membership in such a network allows individual nodes to operate in an open fashion - sharing sensitive files, allowing incoming network connections - because it is implicitly guaranteed that any malicious user from outside world will not be allowed access. Although these solutions have been considered very early in the evolution of ad hoc networks, attempts to adapt similar client-server solutions to a decentralized environment have largely been ineffective. To be efficiently applicable, security solutions for ad hoc networks should ideally have the following characteristics:

• **Lightweight:** Solutions should minimize the amount of computation and communication required to ensure the security services to accommodate the limited energy and computational resources of mobile, ad hoc-enabled devices;

• **Decentralized:** Like ad hoc networks themselves, attempts to secure them must be ad hoc: they must establish security without reference to centralized, persistent entities. Instead, security paradigms should levy the cooperation of all trustworthy nodes in the network;

• **Reactive:** Ad hoc networks are dynamic. Nodes - trustworthy and malicious - may enter and leave the network spontaneously and unannounced. Security paradigms must react to changes in network state; they must seek to detect compromises and vulnerabilities. Therefore, these solutions should be reactive;

• **Fault-Tolerant**: Wireless mediums are known to be unreliable; nodes are likely to leave or be compromised without warning. The communication requirements of security solutions should be Designed with such faults in mind; they should not rely on message delivery or ordering.

Naturally, these are not stringent requirements: specific applications may relax some or all of the above based on their domain and the sensitivity of the information involved. Moreover, many ad hoc network applications do not require 2-party secure communication; instead, achieving broadcast or group security may be all that is needed.

Key Management

Public key systems are generally recognized to have an upper hand in key distribution. In a public key infrastructure, each node has a public/private key pair. A node distributes its public key freely to the other nodes in the network; however it keeps its private key to only itself. A CA is used for key management and has its own public/private key pair. The CA's public key is known to every network node. The trusted CA is responsible to sign certificates, binding public keys to nodes, and has to stay online to verify the current bindings. The public key of a node should be revoked if this node is no longer trusted or leaves the network. A single key management service for an ad hoc network is probably not an acceptable solution, as it is likely to become Achilles' heel of the network. If the CA is unavailable, nodes cannot get the current public keys of other nodes to establish secure connections. In addition, if a CA is compromised, the attacker can sign erroneous certificates using the database of the private keys. Naive replication of CAs can make the network more vulnerable, since compromising of any single replica can cause the system to fail. Hence, it may be more prudent to distribute the trust to a set of nodes by letting these nodes share the key management responsibility.

Secure Routing

The contemporary routing protocols designed for ad hoc networks (discussed in Chapter 2) cope well with dynamically changing topology, but are not designed to provide defense against malicious attackers. In these networks, nodes exchange network topology in order to establish routes between them, and are another potential target for malicious attackers who intend to bring down the network. As for attackers, we can classify them into external and internal. External attackers may inject erroneous routing information, replay old routing data or distort routing information in order to partition or overload the network with retransmissions and inefficient routing. Compromised nodes inside the network are harder to detect and are far more detrimental. Routing information signed by each node may not work, as compromised nodes can generate valid signatures using their private keys. Isolating compromised nodes through routing information is also difficult due to the dynamic topology. Solutions must overcome these potential problems and use some properties of ad hoc networks to facilitate secure routing. Once the compromised nodes have been identified, if there is sufficient number of possibly disjoint and valid routes, the routing protocol should be able to bypass the compromised nodes by using alternate routes.

Cooperation in MANETs

As opposed to networks using dedicated nodes to support basic networking functions such as packet forwarding and routing, in ad hoc networks these functions are carried out by all available network nodes.

There is no reason, however, to assume that the nodes in the network will eventually cooperate with one another since network operation consumes energy, a particularly scarce resource in a battery powered environment like MANETs. The new type of node misbehavior that is specific to ad hoc networks is caused by the lack of cooperation and goes under the name of node selfishness [Yoo2006]. A selfish node does not directly intend to damage other nodes with active attacks (mainly because performing active attacks can be very expensive in terms of energy consumption) but it simply does not cooperate to the network operation, saving battery life for its own communications. Damages provoked by selfish behavior can not be underestimated: a simulation study presented in [Michiardi2002a] shows the impact of a selfish behavior in terms of global network throughput and global communication delay when the DSR routing protocol is used. The simulation results show that even a small percentage of selfish nodes present in the network leads to severe performance degradation.

Furthermore, any security mechanism that tries to enforce cooperation among the nodes ought to focus not only on one particular function, but on both the routing and the packet forwarding function. Schemes that enforce node cooperation in a MANET can be divided in two categories: the first is currency based [Yoo2005] and the second uses a local monitoring technique. The currency based systems are simple to implement but rely on a tamperproof hardware. The main drawback of this approach lies in establishing how the virtual currency has to be exchanged, making their use not realistic in a practical system. On the other hand, cooperative security schemes based on local monitoring of neighbors by each node, evaluating a metric that reflects nodes' behavior. Based on that metric, a selfish node can be gradually isolated from the network.

The main drawback of the second approach is related to the absence of a well-accepted mechanism that securely identifies the nodes of the network: any selfish node could elude the cooperation enforcement mechanism and get rid of its bad reputation just by changing its identity. The main research efforts addressing node selfishness problem are presented as follows. The CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad hoc NeT works)

cooperation mechanism [Buchegger2002a, Buchegger2002b] detects malicious nodes by means of observation or reports about several types of attacks, thus allowing nodes to route around misbehaved nodes and thereby isolate them. CONFIDANT works as an extension to a routing protocol such as DSR. Here, nodes are
provided with:

• A monitor for observations; Reputation records for first-hand and trusted second-hand observations about routing and forwarding behavior of other nodes; • Trust records to control trust given to received warnings; • A path manager to adapt their behavior according to reputation and to take action against malicious nodes. The dynamic behavior of CONFIDANT is as follows. Nodes monitor their neighbors and change the reputation accordingly. If they have some reason to believe that a node is misbehaving, they can take action in terms of their own routing and forwarding and they can decide to inform other nodes by sending an ALARM message. When a node receives such an ALARM either directly or by promiscuously listening to the network, it evaluates how trustworthy the ALARM is based on the source of the ALARM and the accumulated ALARM messages about the node in question. It can then decide whether to take action against the misbehaved node in the form of excluding routes containing the misbehaved node, re-ranking paths in the path cache, reciprocating by non-cooperation, and forwarding an ALARM about the node. Simulations with nodes that do not participate in the forwarding function have shown that CONFIDANT can cope well, even if half of the network population acts maliciously. Further simulations on the effect of second-hand information and slander have shown that slander can effectively be prevented while still retaining a significant detection speed-up over using merely first-hand information. The limitations of CONFIDANT lie in the assertion that the reputation is based on the detection. Events have to be observable and classifiable for detection, and reputation can only be meaningful if the identity of each node is persistent; otherwise it is vulnerable to spoofing attacks.

Token-Based

In an approach presented in [Yang2002], each node of the ad hoc network has a token in order to participate in the network operations, and its local neighbors collaboratively monitor to detect any misbehavior in routing or packet forwarding services. Upon expiration of the token, each node renews its token via its multiple neighbors: the period of validity of a node's token is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its token less frequently as time evolves. The security solution is composed of four closely connected components:

• Neighbor verification: describes how to verify whether each node in the network is a legitimate or malicious node;
• Neighbor monitoring: describes how to monitor the behavior of each node in the network and detect occasional attacks from malicious nodes;
• Intrusion reaction: describes how to alert the network and isolate the attackers;
• Security enhanced routing protocol: explicitly incorporates the security information collected by other components into the ad hoc routing protocol. In the token issuing/renewal phase, it is assumed a global secret key (SK)/public key (PK) pair, where PK is well known by every node of the network. SK is shared by k neighbors who collaboratively sign the token requested or renewed by local nodes. On the other hand, token verification follows three steps: 1) identity match between the nodes's ID and the token ID, 2) validity time verification, 3) issuer signature. If the token verification phase fails, the orresponding node is rejected from the network and both routing and data packets are dropped for that node. Routing security relies on the redundancy of routing information rather than cryptographic techniques enforced by suitably modifying the AODV protocol and the Watchdog technique described earlier. However, the proposed solution possesses some drawbacks. The bootstrap phase needed to generate a valid collection of partial tokens, which are used by a node to create its final token, has some limitations. For example, the number of neighbors necessary to complete the signature of every partial token has to be at least k, suggesting the use of such security mechanism in rather large and dense ad hoc networks. On the other side, the validity period of a token increases proportionally to the time during which the node behaves well, and this feature has less impact if node mobility is high. Frequent changes in the local subset of the network nodes who share a key for issuing valid tokens can cause high computational overhead, not to mention the high traffic generated by issuing/renewing a token, suggesting that the token-based mechanism is more suitable in ad hoc networks where node mobility is low. Spoofing attacks, where a node can request more than one token based on different identities, are not taken into account.

Intrusion Detection systems.

Each MH in an ad hoc network is an autonomous unit and is free to move independently. This implies that a node without adequate physical protection is susceptible to being captured or compromised. It is difficult to track down a single compromised node in a large network. Hence, every node in a wireless ad hoc network should be able to work in a mode wherein it trusts no peer. While intrusion prevention techniques such as encryption and authentication can reduce the risks of intrusion, they cannot be completely eliminated. Intrusion detection can be used as a second line of defense to protect network systems, because once an intrusion is detected, appropriate action can be put in place to minimize the damage, launch counter offensive measures, or even gather evidence for possible follow up prosecution.

Authentication

Authentication denotes the accurate, absolute identification of users who wish to participate in the network. Historically, authentication has been accomplished by a well-known central authentication server. The role of the server is to maintain a database of entities, or users, and their corresponding unique IDs. The ID may be a digital certificate, public key, or both. Unfortunately the ad hoc paradigm does not accommodate a centralized entity creating protocol deployment issues.

Trusted Third Parties

One of the most rudimentary approaches to authentication in ad hoc networks uses a Trusted Third Party (TTP). Every node that wishes to participate in an ad hoc network obtains a certificate from a universally trusted third party. When two nodes wish to communicate, they first check to see if the other node has a valid certificate. Although popular, the TTP approach is laden with flaws. Foremost, it probably is not reasonable to require all ad hoc network-enabled devices to have a certificate. Secondly, each node needs to have a unique name. Although this is reasonable in a large internet, it is a bit too restrictive in an ad hoc setting. Recent research has introduced many appropriate variations of TTPs, and these are discussed later.

Unit-V

Wireless Sensor Networks

A real-world sensor network application is likely to incorporate all the functionalities like sensing and estimation, networking, infrastructure services, sensor tasking, data storage and query. This makes sensor network application development quite different from traditional distributed system development or database programming. With ad hoc deployment and frequently changing network topology, a sensor network application can hardly assume an always-on infrastructure that provides reliable services such as optimal routing, global directories, or service discovery.

There are two types of programming for sensor networks, those carried out by end users and those performed by application developers. An end user may view a sensor network as a pool of data and interact with the network via queries. Just as with query languages for database systems like SQL, a good sensor network programming language should be expressive enough to encode application logic at a high level of abstraction, and at the same time be structured enough to allow efficient execution on the distributed platform. On the other hand, an application developer must provide end users a sensor network with the capabilities of data acquisition, processing, and storage. Unlike general distributed or database systems, collaborative signal and information processing (CSIP) software comprise reactive, concurrent, distributed programs running on ad hoc resource- constrained, unreliable computation and communication platforms. For example, signals are noisy, events can happen at the same time, communication and computation take time, communications may be unreliable, battery life is limited, and so on.

Sensor Node Hardware

Sensor node hardware can be grouped into three categories, each of which entails a different trade-offs in the design choices.

- Augmented general-purpose computers: Examples include low-power PCs, embedded PCs (e.g. PC104), custom-designed PCs, (e.g. Sensoria WINS NG nodes), and various personal digital assistants (PDA). These nodes typically run –ff-the-shelf operating systems such as WinCE, Linux, or real-time operating systems and use standard wireless communication protocols such as IEEE 802.11, Bluetooth, Zigbee etc. Because of their relatively higher processing capability, they can accommodate wide

  variety of sensors, ranging from simple microphones to more sophisticated video cameras.

- Dedicated embedded sensor nodes: Examples include the Berkeley mote family [1], the UCLA Medusa family [2], Ember nodes and MIT AMP [3]. These platforms typically use commercial off-the-shelf (COTS) chip sets with emphasis on small form factor, low power processing and communication, and simple sensor interfaces. Because of their COTS CPU, these platforms typically support at least one programming language, such as C. However, in order to keep the program footprint small to accommodate their small memory size, programmers of these platforms are given full access to hardware but rarely any operating system support. A classical example is the TinyOS platform and its companion programming language, nesC.

- System on-chip (SoC) nodes: Examples of SoC hardware include smart dust [4], the BWRC picoradio node [5], and the PASTA node [6]. Designers of these platforms try to push the hardware limits by fundamentally rethinking the hardware architecture trade-offs for a sensor node at the chip design level. The goal is to find new ways of integrating CMOS, MEMS, and RF technologies to build extremely low power and small footprint sensor nodes that still provide certain sensing, computation, and communication capabilities. Among these hardware platforms, the Berkeley motes, due to their small form factor, open source software development, and commercial availability, have gained wide popularity in the sensor network research.

Sensor Network Programming Challenges

Traditional programming technologies rely on operating systems to provide abstraction for processing, I/O, networking, and user interaction hardware. When applying such a model to programming networked embedded systems, such as sensor networks, the application programmers need to explicitly deal with message passing, event synchronization, interrupt handling, and sensor reading. As a result, an application is typically implemented as a finite state machine (FSM) that covers all extreme cases: unreliable communication channels, long delays, irregular arrival of messages, simultaneous events etc.

For resource-constrained embedded systems with real-time requirements, several mechanisms are used in embedded operating systems to reduce code size, improve response time, and reduce energy consumption. Microkernel technologies [7] modularize the operating system so that only the necessary parts are deployed with the application. Real-time scheduling [8] allocates resources to more urgent tasks so that they can be finished early. Event-driven execution allows the system to fall into low-power sleep mode when no interesting events need to be processed. At the extreme, embedded operating systems tend to expose more hardware controls to the programmers, who now have to directly face device drivers and scheduling algorithms, and optimize code at the assembly level. Although these techniques may work well for small, stand-alone embedded systems, they do not scale up for the programming of sensor networks for two reasons:

- Sensor networks are large-scale distributed systems, where global properties are derivable from program execution in a massive number of distributed nodes. Distributed algorithms themselves are hard to implement, especially when infrastructure support is limited due to the ad hoc formation of the system and constrained power, memory, and bandwidth resources.
- As sensor nodes deeply embed into the physical world, a sensor network should be able to respond to multiple concurrent stimuli at the speed of changes of the physical phenomena of interest.

There no single universal design methodology for all applications. Depending on the specific tasks of a sensor network and the way the sensor nodes are organized, certain methodologies and platforms may be better choices than others. For example, if the network is used for monitoring a small set of phenomena and the sensor nodes are organized in a simple star topology, then a client-server software model would be sufficient. If the network is used for monitoring a large area from a single access point (i.e., the base station), and if user queries can be decoupled into aggregations of sensor readings from a subset of nodes, then a tree structure that is rooted at the base station is a better choice. However, if the phenomena to be monitored are moving targets, as in the target tracking, then neither the simple client-server model nor the tree organization is optimal. More sophisticated design and methodologies and platforms are required.

Node-Level Software Platforms

Most design methodologies for sensor network software are node-centric, where programmers think in terms of how a node should behave in the environment. A node- level platform can be node-centric operating system, which provides hardware and networking abstractions of a sensor node to programmers, or it can be a language platform, which provides a library of components to programmers.

A typical operating system abstracts the hardware platform by providing a set of services for applications, including file management, memory allocation, task scheduling, peripheral device drivers, and networking. For embedded systems, due to their highly specialized applications and limited resources, their operating systems make different trade-offs when providing these services. For example, if there is no file management requirement, then a file system is obviously not needed. If there is no dynamic memory allocation, then memory management can be simplified. If prioritization among tasks is critical, then a more elaborate priority scheduling mechanism may be added.

Operating System: TinyOS

Tiny OS aims at supporting sensor network applications on resource-constrained hardware platforms, such as the Berkeley motes.

To ensure that an application code has an extremely small foot-print, TinyOS chooses to have no file system, supports only static memory allocation, implements a simple task model, and provides minimal device and networking abstractions. Furthermore, it takes a language-based application development approach so that only the necessary parts of the operating system are compiled with the application. To a certain extent, each TinyOS application is built into the operating system.

Like many operating systems, TinyOS organizes components into layers. Intuitively, the lower a layer is, the ‘closer’ it is to the hardware; the higher a layer is, the closer it is to the application. In addition to the layers, TinyOS has a unique component architecture and provides as a library a set of system software components. A components specification is independent of the components implementation. Although most components encapsulate software functionalities, some are just thin wrappers around hardware. An application, typically developed in the nesC language, wires these components together with other application-specific components.

A program executed in TinyOS has two contexts, tasks and events, which provide two sources of concurrency. Tasks are created (also called posted) by components to a task scheduler. The default implementation of the TinyOS scheduler maintains a task queue and invokes tasks according to the order in which they were posted. Thus tasks are deferred computation mechanisms. Tasks always run to completion without preempting or being preempted by other tasks. Thus tasks are non-preemptive. The scheduler invokes a new task from the task queue only when the current task has completed. When no tasks are available in the task queue, the scheduler puts the CPU into the sleep mode to save energy.

The ultimate sources of triggered execution are events from hardware: clock, digital inputs, or other kinds of interrupts. The execution of an interrupt handler is called an event context. The processing of events also runs to completion, but it preempts tasks and can be preempted by other events. Because there is no preemption mechanism among tasks and because events always preempt tasks, programmers are required to chop their code, especially the code in the event contexts, into small execution pieces, so that it will not block other tasks for too long.

Another trade-off between non-preemptive task execution and program reactiveness is the design of split-phase operations in TinyOS. Similar to the notion of asynchronous method calls in distributed computing, a split-phase operation separates the initiation of a method call from the return of the call. A call to split-phase operation returns immediately, without actually performing the body of the operation. The true execution of the operation is scheduled later; when the execution of the body finishes, the operation notifies the original caller through a separate method call.

In summary, many design decisions in TinyOS are made to ensure that it is extremely lightweight. Using a component architecture that contains all variables inside the components and disallowing dynamic memory allocation reduces the memory management overhead and makes the data memory usage statically analyzable. The simple concurrency model allows high concurrency with low thread maintenance overhead. However, the advantage of being lightweight is not without cost. Many hardware idiosyncrasies and complexities of concurrency management are left for the application programmers to handle. Several tools have been developed to give programmers language-level support for improving programming productivity and code robustness.

Imperative Language: nesC

nesC [9] is an extension of C to support and reflect the design of TinyOS. It provides a set of language constructs and restrictions to implement TinyOS components and applications. A component in nesC has an interface specification and an implementation. To reflect the layered structure of TinyOS, interfaces of a nesC component are classified as provides or uses interfaces. A provides interface is a set of method calls exposed to the upper layers, while a uses interface is a set of method calls hiding the lower layer components. Methods in the interfaces can be grouped and named.

Although they have the same method call semantics, nesC distinguishes the directions of the interface calls between layers as event calls and command calls. An event call is a method call from a lower layer component to a higher layer component, while a command is the opposite.

The separation of interface type definitions from how they are used in the components promotes the reusability of standard interfaces. A component can provide and use the same interface type, so that it can act as a filter interposed between a client and a service. A component may even use or provide the same interface multiple times.

Component Implementation

There are two types of components in nesC, depending on how they are implemented: modules and configurations. Modules are implemented by application code (written in a C-like syntax). Configurations are implemented by connecting interfaces of existing components. nesC also supports the creation of several instances of a component by declaring abstract components with optional parameters. Abstract components are created at compile time in configuration. As TinyOS does not support dynamic memory allocation, all components are statically constructed at compile time. A complete application is always a configuration rather than a module. An application must contain the main module, which links the code to the scheduler at run time. The main has single StdControl interface, which is the ultimate source of initialization of all components.

Dataflow-Style Language: TinyGALS

Dataflow languages [10] are intuitive for expressing computation on interrelated data units by specifying data dependencies among them. A dataflow diagram has a set of processing units called actors. Actors have ports to receive and produce data, and the directional connections among ports are FIFO queues that mediate the flow of data. Actors in dataflow languages intrinsically capture concurrency in a system, and the FIFO queues give a structured way of decoupling their executions. The execution of an actor is triggered when there are enough input data at the input ports.

Asynchronous event-driven execution can be viewed as a special case of dataflow models, where each actor is triggered by every incoming event. The globally asynchronous and locally synchronous (GALS) mechanism is a way of building event- triggered concurrent execution from thread-unsafe components. TinyGALS is such as language for TinyOS.

One of the key factors that affect component reusability in embedded software is the component composability, especially concurrent composability. In general, when developing a component, a programmer may not anticipate all possible scenarios in which the component may be used. Implementing all access to variables as atomic blocks, incurs too much overhead. At the other extreme, making all variable access unprotected is easy for coding but certainly introduces bugs in concurrent composition. TinyGALS addresses concurrency concerns at the system level, rather than at component level as in nesC. Reactions to concurrent events are managed by a dataflow-style FIFO queue communication.

TinyGALS Programming Model

TinyGALS supports all TinyOS components, including its interfaces and module implementations. All method calls in a component interface are synchronous method calls- that is, the thread of control enters immediately into the callee component from the caller component. An application in TinyGALS is built in two steps: (1) constructing asynchronous actors from synchronous components, and (2) constructing an application by connecting the asynchronous components through FIFO queues.

An actor in TinyGALS has a set of input ports, a set of output ports, and a set of connected TinyOS components. An actor is constructed by connecting synchronous method calls among TinyOS components.

At the application level, the asynchronous communication of actors is mediated using FIFO queues. Each connection can be parameterized by a queue size. In the current implementation of TinyGALS, events are discarded when the queue is full. However, other mechanisms such as discarding the oldest event can be used.

Node-Level Simulators

Node-level design methodologies are usually associated with simulators that simulate the behavior of a sensor network on a per-node basis. Using simulation, designers can quickly study the performance (in terms of timing, power, bandwidth, and scalability) of potential algorithms without implementing them on actual hardware and dealing with the vagaries of actual physical phenomena. A node-level simulator typically has the following components:

- Sensor node model: A node in a simulator acts as a software execution platform, a sensor host, as well as a communication terminal. In order for designers to focus on the application-level code, a node model typically provides or simulates a communication protocol stack, sensor behaviors (e.g., sensing noise), and operating system services. If the nodes are mobile, then the positions and motion properties of the nodes need to be modeled. If energy characteristics are part of the design considerations, then the power consumption of the nodes needs to be modeled.

- Communication model: Depending on the details of modeling, communication may be captured at different layers. The most elaborate simulators model the communication media at the physical layer, simulating the RF propagation delay and collision of simultaneous transmissions. Alternately, the communication may be simulated at the MAC layer or network layer, using, for example, stochastic processes to represent low-level behaviors.

- Physical environment model: A key element of the environment within a sensor network operates is the physical phenomenon of interest. The environment can also be simulated at various levels of details. For example, a moving object in the physical world may be abstracted into a point signal source. The motion of the point signal source may be modeled by differential equations or interpolated from a trajectory profile. If the sensor network is passive- that is, it does not impact the behavior of the environment-then the environment can be simulated separately or can even be stored in data files for sensor nodes to read in. If, in addition to sensing, the network also performs actions that influence the behavior of the environment, then a more tightly integrated simulation mechanism is required.

- Statistics and visualization: The simulation results need to be collected for analysis. Since the goal of a simulation is typically to derive global properties from the execution of individual nodes, visualizing global behaviors is extremely important. An ideal visualization tool should allow users to easily observe on demand the spatial distribution and mobility of the nodes, the connectivity among nodes, link qualities, end-to-end communication routes and delays, phenomena and their spatio-temporal dynamics, sensor readings on each node, sensor nodes states, and node lifetime parameters (e.g., battery power).

A sensor network simulator simulates the behavior of a subset of the sensor nodes with respect to time. Depending on how the time is advanced in the simulation, there are two types of execution models: cycle-driven simulation and discrete-event simulation. A cycle-driven (CD) simulation discretizes the continuous notion of real time into (typically regularly spaced) ticks and simulates the system behavior at these ticks. At each tick, the physical phenomena are first simulated, and then all nodes are checked to see if they have anything to sense, process, or communicate. Sensing and computation are assumed to be finished before the next tick. Sending a packet is also assumed to be completed by then. However, the packet will not be available for the destination node until next tick. This split-phase communication is a key mechanism to reduce cyclic dependencies that may occur in cycle-driven simulations. Most CD simulators do not allow interdependencies within a single tick.

Unlike cycle-driven simulators, a discrete-vent (DE) simulator assumes that the time is continuous and an event may occur at any time. As event is 2-tuple with a value and a time stamp indicating when the event is supposed to be handled. Components in a DE simulation react to input events and produce output events. In node-level simulators, a component can be a sensor node, and the events can be communication packets; or a component can be software module within and the events can be message passings among these nodes. Typically, components are causal, in the sense that if an output event is computed from an input event, then the time stamp of the output should not be earlier than that of the input event. Non-causal components require the simulators to be able to roll back in time, and worse, they may not define a deterministic behavior of a system [11]. A DE simulator typically requires a global event queue. All events passing between nodes or modules are put in the event queue and sorted according to their chronological order. At each iteration of the simulation, the simulator removes the first event (the one with earliest time stamp) from the queue and triggers the component that reacts to that event.

In terms of timing behavior, a DE simulator is more accurate than a CD simulator, and as a consequence, DE simulators run slower. The overhead of ordering all events and computation, in addition to the values and time stamps of events, usually dominates the computation time. At an early stage of a design when only the asymptotic behaviors rather than timing properties are of concern, CD simulations usually require less complex components and give faster simulations. This is partly because of the approximate timing behaviors, which make simulation results less comparable from application to application, there is no general CD simulator that fits all sensor network simulation tasks. Many of the simulators are developed for particular applications and exploit application-specific assumptions to gain efficiency.

DE simulations are sometimes considered as good as actual implementations, because of their continuous notion of time and discrete notion of events. There are several open- source or commercial simulators available. One class of these simulators comprises extensions of classical network simulators, such as ns-2, J-Sim (previously known as JavaSim), and GloMoSim/ Qualnet. The focus of these simulators is on network modeling, protocol stacks, and simulation performance. Another class of simulators, sometimes called software-in-the-loop simulators, incorporate the actual node software into the simulation. For this reason, they are typically attached to particular hardware platforms and are less portable. Example include TOSSIM [12] for Berkeley motes and Em* [13] for Linux-based nodes such as Sensoria WINS NG platforms.

The ns-2 Simulator and its Sensor Network Extensions

The simulator ns-2 is an open-source network simulator that was originally designed for wired, IP networks. Extensions have been made to simulate wireless/mobile networks (e.g. 802.11 MAC and TDMA MAC) and more recently sensor networks. While the original ns-2 only supports logical addresses for each node, the wireless/mobile extension of it (e.g. [14]) introduces the notion of node locations and a simple wireless channel model. This is not a trivial extension, since once the nodes move, the simulator needs to check for each physical layer event whether the destination node is within the communication range. For a large network, this significantly slows down the simulation speed.

There are two widely known efforts to extend ns-2 for simulating sensor networks: SensorSim form UCLA [15] and the NRL sensor network extension from the Navy Research Laboratory [16]. SensorSim also supports hybrid simulation, where some real sensor nodes, running real applications, can be executed together with a simulation. The NRL sensor network extension provides a flexible way of modeling physical phenomena in a discrete event simulator. Physical phenomena are modeled as network nodes which communicate with real nodes through physical layers.

The main functionality of ns-2 is implemented in C++, while the dynamics of the simulation (e.g., time-dependent application characteristics) is controlled by Tcl scripts. Basic components in ns-2 are the layers in the protocol stack. They implement the handlers interface, indicating that they handle events. Events are communication packets that are passed between consecutive layers within one node, or between the same layers across nodes.

The key advantage of ns-2 is its rich libraries of protocols for nearly all network layers and for many routing mechanisms. These protocols are modeled in fair detail, so that they closely resemble the actual protocol implementations. Examples include the following:

- TCP: reno, tahoe, vegas, and SACK implementations.

MAC: 802.3, 802.11, and TDMA. Ad hoc routing: Destination sequenced distance vector (DSDV) routing, dynamic source routing (DSR), ad hoc on-demand distance vector (AOPDV) routing, and temporarily ordered routing algorithm (TORA).

- Sensor network routing: Directed diffusion, geographical routing (GEAR) and geographical adaptive fidelity (GAF) routing.

The Simulator TOSSIM

TOSSIM is a dedicated simulator for TinyOS applications running on one or more Berkeley motes. The key design decisions on building TOSSIM were to make it scalable to a network of potentially thousands of nodes, and to be able to use the actual software code in the simulation. To achieve these goals, TOSSIM takes a cross-compilation approach that compiles the nesC source code into components in the simulation. The event-driven execution model of TinyOS greatly simplifies the design of TOSSIM. By replacing a few low-level components such as the A/D conversion (ADC), the system clock, and the radio front end, TOSSIM translates hardware interrupts into discrete-event simulator events. The simulator event queue delivers the interrupts that drive the execution of a node. The upper-layer TinyOS code runs unchanged.

TOSSIM uses a simple but powerful abstraction to model a wireless network. A network is a directed graph, where each vertex is a sensor node and each directed edge has a bit- error rate. Each node has a private piece of state representing what it hears on the radio channel. By setting connections among the vertices in the graph and a bit-error rate on each connection, wireless channel characteristics, such as imperfect channels, hidden terminal problems, and asymmetric links can be easily modeled. Wireless transmissions are simulated at the bit level. If a bit error occurs, the simulator flips the bit.

TOSSIM has a visualization package called TinyViz, which is a Java application that can connect to TOSSIM simulations. TinyViz also provides mechanisms to control a running simulation by, for example, modifying ADC readings, changing channel properties, and injecting packets. TinyViz is designed as a communication service that interacts with the TOSSIM event queue. The exact visual interface takes the form of plug-ins that can interpret TOSSIM events. Beside the default visual interfaces, users can add application- specific ones easily.

Programming Beyond Individual Nodes: State-centric Programming

Many sensor network applications, such as target tracking, are not simply generic distributed programs over an ad hoc network of energy-constrained nodes. Deeply rooted in these applications is the notion of states of physical phenomena and models of their evolution over space and time. Some of these states may be represented on a small number of nodes and evolve over time, as in the target tracking problem, while others may be represented over a large and spatially distributed number of nodes, as in tracking a temperature contour.

A distinctive property of physical states, such as location, shape, and motion of objects, is their continuity in space and time. Their sensing and control is typically done through sequential state updates. System theories, the basis for most signal and information processing algorithms, provide abstractions for state updates, such as:

$$xk+1 = f(xk, uk) \quad yk = g(xk, uk)$$

where x is the state of a system, u is the system input, y is the output and k is an integer update index over space and/or time, f is the state update function, and g is the output or observation function. This formulation is broad enough to capture a wide variety of algorithms in sensor fusion, signal processing, and control (e.g., Kalman filtering, Bayesian estimation, system identification, feedback control laws, and finite-state automata).

However, in distributed real-time embedded systems such as sensor networks, the formulation is not as clean as represented in the above equations. The relationships among subsystems can be highly complex and dynamic over space and time. The following issues (which are not explicitly tackled in the above equations) must be properly addressed during the design to ensure the correctness and efficiency of the system.

These issues, addressing where and when, rather than how, to perform sensing, computation, and communication, play a central role in the overall system performance. However, these 'non-functional" aspects of computation, related to concurrency, responsiveness, networking, and resource management, are not well supported by traditional programming models and languages. State-centric programming aims at providing design methodologies and frameworks that give meaningful abstractions for these issues, so that system designers can continue to write algorithms on top of an intuitive understanding of where and when the operations are performed.

A collaborative group is such an abstraction. A collaborative group is a set of entities that contribute to a state update. These entities can be physical sensor nodes, or they can be more abstract system components such as virtual sensors or mobile agents hopping among sensors. These are all referred to as agents.

Intuitively, a collaboration groups provides two abstractions: its scope to encapsulate network topologies and its structure to encapsulate communication protocols. The scope of a group defines the membership of the nodes with respect to the group. A software agent that hops among the sensor nodes to track a target is a virtual node, while a real node is physical sensor. Limiting the scope of a group to a subset of the entire space of all agents improves scalability. Grouping nodes according to some physical attributes rather than node addresses is an important and distinguishing characteristic of sensor networks.

The structure of a group defines the "roles" each member plays in the group, and thus the flow of data. Are all members in the group equal peers? Is there a "leader" member in the group that consumes data? Do members in the group form a tree with parent and children relations? For example, a group may have a leader node that collects certain sensor readings from all followers. By mapping the leader and the followers onto concrete sensor nodes, one can effectively define the flow of data from the hosts of followers to the host of the leader. The notion of roles also shields programmers from addressing individual nodes either by name or address. Furthermore, having multiple members with the same role provides some degree of redundancy and improves robustness of the application in the presence of node and link failures.