

1

Routing Approaches in Mobile Ad hoc Networks

1.1 INTRODUCTION

Routing in ad hoc networks has become a popular research topic. Dating back to the early 1980s, there have been a large number of routing protocols designed for multi-hop ad hoc networks. These protocols cover a wide range of design choices and approaches, from simple modifications of Internet protocols, to more complex, multi-level hierarchical schemes.

Many of these routing protocols have been designed based on similar sets of assumptions. For instance, most routing protocols assume that all nodes have homogeneous resources and capabilities. This includes the transmission ranges of the nodes. Also, bi-directional links are often assumed. In some instances protocols have mechanisms for determining whether links are bi-directional. In these cases, the protocols will then eliminate uni-directional links from consideration for routing. In other instances, protocols can actually utilize these uni-directional links, while other protocols simply assume all links are bi-directional. Finally, while the ultimate end-goal of a protocol may be operation in large networks, most protocols are typically designed for moderately sized networks of 10 to 100 nodes.

Before describing the types of approaches and example protocols, it is important to explain the developmental goals for an ad hoc routing protocol so that the design choices of the protocols can be better understood. As has already been stated in previous chapters, the defining characteristics of ad hoc networks include resource poor devices, limited bandwidth, high error rates, and a continually changing topology. Among the available resources, battery power is typically the most constraining. Hence the following are typical design goals for ad hoc network routing protocols:

2 ROUTING APPROACHES IN MOBILE AD HOC NETWORKS

- **Minimal control overhead:** Control messaging consumes bandwidth, processing resources, and battery power to both transmit and receive a message. Because bandwidth is at a premium, routing protocols should not send more than the minimum number of control messages they need for operation, and should be designed so that this number is relatively small. While transmitting is roughly twice as power consuming as receiving [26], both operations are still power consumers for the mobile devices. Hence, reducing control messaging also helps to reserve battery power.
- **Minimal processing overhead:** Algorithms that are computationally complex require significant processing cycles in devices. Because the processing cycles cause the mobile device to use resources, more battery power is consumed. Protocols that are lightweight and require a minimum of processing from the mobile device reserve battery power for more user-oriented tasks and extend the overall battery lifetime.
- **Multi-hop routing capability:** Because the wireless transmission range of mobile nodes is often limited, sources and destinations may typically not be within direct transmission range of each other. Hence, the routing protocol must be able to discover multi-hop routes between sources and destinations so that communication between those nodes is possible.
- **Dynamic topology maintenance:** Once a route is established, it is likely that some link in the route will break due to node movement. In order for a source to communicate with a destination, a viable routing path must be maintained, even while the intermediate nodes, or even the source or destination nodes, are moving. Further, because link breaks on ad hoc networks are common, link breaks must be handled quickly with a minimum of associated overhead.
- **Loop prevention:** Routing loops occur when some node along a path selects a next hop to the destination is also a node that occurred earlier in the path. When a routing loop exists, data and control packets may traverse the path multiple times until either the path is fixed and the loop is eliminated, or until the time to live (TTL) of the packet reaches zero. Because bandwidth is scarce and packet processing and forwarding is expensive, routing loops are extremely wasteful of resources and are detrimental to the network. Even a transitory routing loop will have a negative impact on the network. Hence, loops should be avoided at all times.

With these goals in mind, numerous routing protocols have been developed for ad hoc networks. There are far too many proposed routing protocols than can be discussed in this chapter. This chapter describes the characteristics of classes of routing approaches, and then describes the operation of particular routing protocols within that class. The routing protocols that are described are selected for a number of reasons. They may be popular choices for routing-related research among the ad hoc community; they may, at the time of this writing, be under consideration of the Mobile Ad hoc Networks (MANET) Working Group [34] of the IETF for

standardization; or, they may simply be good, illustrative examples of their particular class of protocol. This chapter does not make comparisons regarding the discussed protocols. There have been many published studies comparing the performance of ad hoc routing protocols in a variety of scenarios. The interested reader should see [8, 15, 16, 23, 32, 53] for comparisons of some of the discussed protocols. Further, the citations for the protocols themselves often contain an evaluation of the protocol, and sometimes a comparison with other ad hoc routing protocols. Finally, a high-level description of each protocol is presented. Consequently, many operational details have omitted. Additional details of each protocol can be found in its respective citations.

1.2 PROACTIVE APPROACHES

The proactive routing approaches designed for ad hoc networks are derived from the traditional distance vector [35] and link state [38] protocols developed for use in the wireline Internet. The primary characteristic of proactive approaches is that each node in the network maintains a route to every other node in the network at all times. Route creation and maintenance is accomplished through some combination of periodic and event-triggered routing updates. Periodic updates consist of routing information exchanges between nodes at set time intervals. The updates occur at specific intervals, regardless of the mobility and traffic characteristics of the network. Event-triggered updates, on the other hand, are transmitted whenever some event, such as a link addition or removal, occurs. The mobility rate directly impacts the frequency of event-triggered updates because link changes are more likely to occur as mobility increases.

Proactive approaches have the advantage that routes are available the moment they are needed. Because each node consistently maintains an up-to-date route to every other node in the network, a source can simply check its routing table, when it has data packets to send to some destination, and begin packet transmission. However, the primary disadvantage of these protocols is that the control overhead can be significant in large networks, or in networks with rapidly moving nodes. Further, the amount of routing state maintained at each node scales as $O(n^2)$, where n is the number of nodes in the network. Proactive protocols tend to perform well in networks where there is a significant number of data sessions within the network. In these networks, the overhead of maintaining each of the paths is justified because many of these paths are utilized.

1.2.1 Destination-Sequenced Distance Vector Routing

The Destination-Sequenced Distance Vector (DSDV) routing protocol [46] is a distance vector protocol that implements a number of customizations to make its operation more suitable for ad hoc mobile networks. DSDV utilizes per-node sequence numbers to avoid the *counting to infinity* problem common in many distance vector

4 ROUTING APPROACHES IN MOBILE AD HOC NETWORKS

protocols. A node increments its sequence number whenever there is a change in its local neighborhood (i.e., a link addition or removal). When given a choice between two routes to a destination, a node always selects the route with the greatest destination sequence number. This ensures utilization of the most recent information.

Because DSDV is a proactive protocol, each node maintains a route to every other node in the network. The routing table contains the following information for each entry: destination IP address, destination sequence number, next hop IP address, hop count, and install time. DSDV utilizes both periodic and event-triggered routing table updates. Every time interval, each node broadcasts to its neighbors its current sequence number, along with any routing table updates. The routing table updates are of the form:

< destination IP address, destination sequence number, hopcount >

After receiving an update message, the neighboring nodes utilize this information to compute their routing table entries using an iterative distance vector approach [35]. In addition to periodic updates, DSDV also utilizes event-triggered updates to announce important link changes, such as link removals. Such event-triggered updates ensure timely discovery of routing path changes.

As stated previously, if a node learns two distinct paths to a destination, the node selects the path with the greatest associated destination sequence number. This ensures the utilization of the most recent routing information for that destination. When given the choice between two paths with equal destination sequence numbers, the node selects the path with the shortest hop count. On the other hand, if all metrics are equivalent, then the choice between routes is arbitrary.

DSDV implements two primary optimizations to improve performance in mobile networks. The first is that it defines two types of updates: full and incremental. Full updates are transmissions of a node's entire routing table. Because the size of these updates scales with the size of the network, these updates are performed relatively infrequently. To reduce processing overhead and bandwidth consumption, incremental updates are transmitted more frequently. Incremental updates include only those routing table entries that have changed since the last full update. Once the number of routing changes becomes too great to fit into a single NPDU, a full update is transmitted during the next update period.

Finally, DSDV also implements a mechanism to damp routing fluctuations. Due to the unsynchronized nature of the periodic updates, routing updates for a given destination can propagate along different paths at different rates. To prevent a node from announcing a routing path change for a given destination while another, better, update for that destination is still en-route, DSDV requires nodes to wait a *settling time* before announcing a new route with a higher metric for a destination. The settling time is the average time to get all the updated advertisements for a route. In this way, the node can be sure to receive all routing path changes for a destination before propagating any of those changes. This reduces bandwidth utilization and power consumption by neighboring nodes.

1.2.2 Optimized Link State Routing

The Optimized Link State Routing (OLSR) protocol [14] is a variation of traditional link state routing, modified for improved operation in ad hoc networks. The key feature of OLSR is its use of *multipoint relays* (MPRs) to reduce the overhead of network floods and the size of link state updates. Each node computes its MPRs from its set of neighbors. The MPR set is selected such that when a node broadcasts a message, the retransmission of that message by the MPR set will ensure that the message is received by each of its two-hop neighbors. Hence, whenever a node broadcasts a message, only those neighbors in its MPR set rebroadcast the message. Other neighbors, that are not in the MPR set, process the message but do not rebroadcast it. Further, when exchanging link state routing information, a node only lists its connections to those neighbors that have selected it as an MPR. That set of neighbors is termed the *MPR Selectors*.



Fig. 1.1 Multipoint Relays.

The MPR set for a given node is the set of neighbors that covers the two-hop neighborhood of the node, as shown in figure 1.1. Nodes learn their set of two-hop neighbors through the periodic exchange of Hello messages. Each node periodically transmits a Hello message that contains a list of all neighbors. Associated with each neighbor is an attribute indicating the directionality of the link to that neighbor. The node is labeled *symmetric* if the link to the neighbor is bi-directional, while the node is labeled *asymmetric* if a Hello has been received from that node, but the link has not been confirmed as bi-directional. When a node receives this Hello message from each of its neighbors, it obtains complete knowledge of its two-hop neighbor set at that point in time. Further, if its own address is listed in the Hello message, it knows the link with that neighbor is bi-directional. It can then update the status of that neighbor to be symmetric.

The MPR set may be calculated according to the following algorithm [27]. Each node starts with an empty MPR set. The set N is defined to be the set of one-hop neighbors with which there exists bi-directional connectivity, and the set N_2 is the set of two-hop bi-directional neighbors. The first nodes that are selected for the MPR set are those nodes in N that are the only neighbors of some node in N_2 . Next, the degree of each node n_2 in N that is not in the MPR set is calculated, where the degree is the number of nodes in N_2 that n_2 covers that are still uncovered by the MPR set. As long as there are still nodes in N_2 that are not covered by nodes in the MPR set,

the node in N that has the highest degree is included in the MPR set. Once all the nodes in N_2 are covered, this process terminates.

Once each node's MPR set is selected, routing paths within the network can be determined. Because OLSR is a proactive protocol, each node maintains a route to every other node in the network. To diffuse topology information, nodes periodically exchange *Topology Control* (TC) messages with their neighbors. The TC message for a given node lists the set of neighbors that have selected the sending node as an MPR. This is called the *Multipoint Relay Selector* set of the node. Only this set of nodes is advertised within the network. As a node receives TC messages from the other network nodes, it can create or modify routing entries to each node in the network using any shortest path routing algorithm, such as a variation of Dijkstra's algorithm.

1.2.3 Topology Dissemination Based on Reverse-Path Forwarding

A different proactive approach is taken by the Topology-Based Reverse Path Forwarding (TBRPF) protocol [6]. Like OLSR, TBRPF is a link-state routing protocol; however, TBRPF employs a different technique for reducing overhead. TBRPF nodes compute a shortest path tree to all network nodes. To minimize bandwidth utilization, the nodes then propagate only part of this tree to their neighbors. TBRPF consists of two main modules: a neighbor discovery module for maintaining neighborhood information, and a routing module for topology discovery and route computation.

The neighbor discovery module enables nodes to detect neighbors and determine the type of connectivity to each neighbor. Connectivity can be either bi-directional, uni-directional, or, in the case of a broken link, the link can be lost. Each node periodically broadcasts a Hello message to its neighbors. The Hello message is *differential*, in the sense that only changes in the status of neighbors are reported. Each Hello message contains three categories of neighbor information. A neighbor can be listed in the *neighbor request*, *neighbor reply*, or *neighbor lost* category. The categories aid the nodes in determining the directionality of the links with their neighbors. In general, when a node i changes the status of its neighbor j , it includes node j in the appropriate list (indicating the neighbor's new status) in (typically) three consecutive Hello messages. This ensures that node j will either learn of the status change or will declare node i to be lost after missing this number of Hello messages.

The neighbor request list contains the addresses of those neighbors from which the node has recently received Hellos, but for which it has not yet been determined that the link to those neighbors is bi-directional. When a node i receives a Hello from a new neighbor j , node i lists j in its neighbor table and flags the entry as a uni-directional link. The next time node i transmits a Hello message, node i lists node j in the neighbor request list of that message. This indicates that i is requesting that j confirm the receipt of i 's Hello, so that the bi-directionality of the link can be confirmed. When node j receives node i 's Hello listing j in the neighbor request, node j creates a neighbor table entry for i (if one does not already exist), and marks the entry as bi-directional. In its next Hello, node j includes i 's address in the neighbor reply

list, indicating that the Hello message from i was received and a bi-directional link exists. Node i can then update the entry for node j to be bi-directional. To prevent the inclusion of transient links, nodes can wait to receive some threshold number of Hello messages from a neighbor before creating a neighbor table entry for that node.

Once a node has created a neighbor table entry for a neighbor, the node must monitor the status of that link to ensure connectivity to the neighbor continues to exist. If a node i misses the threshold number of Hello messages from a neighbor j , it updates the state of that neighbor to be lost. The next time i sends a Hello message, it includes node j 's address in the neighbor lost list. If j receives the Hello, it notes that the bi-directional connection to i has been lost, and it changes the status of node i in its neighbor table to uni-directional. Otherwise, if node j fails to receive further Hello messages, it deletes node i from its neighbor table and includes it in the neighbor lost list of future Hello messages.

To perform routing, each TBRPF node computes a shortest path source tree to each reachable node in the network. The tree is computed using a modified version of Dijkstra's algorithm. After computing the tree, nodes report only a part of the tree, called the *reportable subtree* (RT), to neighboring nodes. To report RT, two types of topology updates message are used. Periodic updates are sent reporting the entire RT. These full updates are utilized to inform new neighbors of RT and ensure that all the necessary topology information is eventually propagated. Smaller, more frequent updates are sent as differential updates. The differential updates report only those changes to RT that have occurred since the last periodic update. To reduce the number of control messages, topology updates can be combined with Hello messages so that fewer control packets are transmitted.

To calculate RT, let $T(j)$ denote the subtree of node i 's source tree rooted at neighbor j . For each neighbor j , node i includes $T(j)$ in its reportable subtree RT if and only if it determines that one of its neighbors may select i to be its next hop on its shortest path to j . To make this determination, node i computes the min-hop paths from each neighbor to every other neighbor, using the node ID to break any ties.

1.3 REACTIVE APPROACHES

Reactive routing techniques, also called *on-demand* routing, take a very different approach to routing than proactive protocols. A large percentage of the overhead from proactive protocols stems from the need for every node to maintain a route to every other node at all times. In a wired network, where connectivity patterns change relatively infrequently and resources are abundant, maintaining full connectivity graphs is a worthwhile expense. The benefit is that when a route is needed, it is immediately available. In an ad hoc network, however, link connectivity can change frequently and control overhead is costly. Because of these reasons, reactive routing approaches take a departure from traditional Internet routing approaches by not continuously maintaining a route between all pairs of network nodes. Instead, routes are only discovered when they are actually needed. When a source node needs to send

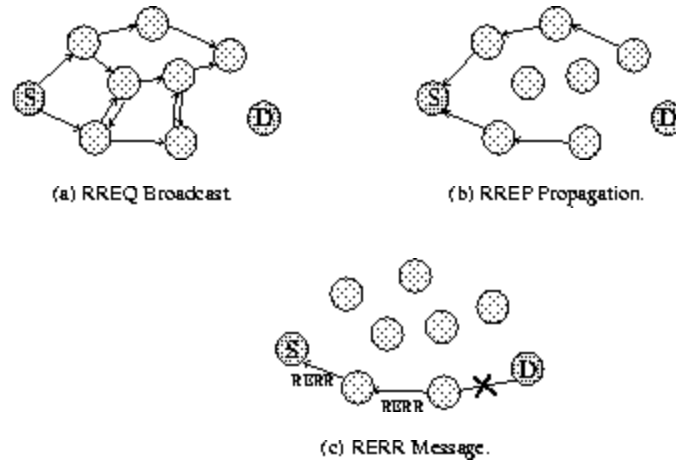


Fig. 1.2 AODV Route Discovery and Maintenance.

data packets to some destination, it checks its route table to determine whether it has a route. If no route exists, it performs a *route discovery* procedure to find a path to the destination. Hence, route discovery becomes on-demand. If two nodes never need to talk to each other, then they do not need to utilize their resources maintaining a path between each other. The route discovery typically consists of the network-wide flooding of a request message. To reduce overhead the search area may be reduced by a number of optimizations [10, 25, 31].

The benefit of this approach is that signaling overhead is likely to be reduced compared to proactive approaches, particularly in networks with low to moderate traffic loads. When the number of data sessions in the network becomes high, then the overhead generated by the route discoveries approaches, and may even surpass, that of the proactive approaches. The drawback to reactive approaches is the introduction of a *route acquisition latency*. That is, when a route is needed by a source node, there is some finite latency while the route is discovered. In contrast, with a proactive approach, routes are typically available the moment they are needed. Hence there is no delay to begin the data session.

1.3.1 Ad hoc On-Demand Distance Vector Routing

The Ad hoc On-Demand Distance Vector (AODV) Routing Protocol [48] provides on-demand route discovery in mobile ad hoc networks. Like most reactive routing protocols, route finding is based on a route discovery cycle involving a broadcast network search and a unicast reply containing discovered paths. Similar to DSDV, AODV relies on per-node sequence numbers for loop freedom and for ensuring selection of the most recent routing path. AODV nodes maintain a route table in which

next hop routing information for destination nodes is stored. Each routing table entry has an associated lifetime value. If a route is not utilized within the lifetime period, the route is expired. Otherwise, each time the route is used, the lifetime period is updated so that the route is not prematurely deleted.

When a source node has data packets to send to some destination, it first checks its route table to determine whether it already has a route to the destination. If such a route exists, it can use that route for data packet transmissions. Otherwise, it must initiate a route discovery procedure to find a route. To start route discovery, the source node creates a *route request* (RREQ) packet. It places in this packet the destination node's IP address, the last known sequence number for that destination, and the source's IP address and current sequence number. The RREQ also contains a hop count, initialized to zero, and a RREQ ID. The RREQ ID is a per-node monotonically increasing counter that is incremented each time the node initiates a new RREQ. In this way, the source IP address, together with the RREQ ID, uniquely identifies a RREQ and can be used to detect duplicates. After creating this message, the source broadcasts the RREQ to its neighbors.

When a neighboring node receives a RREQ, it first creates a *reverse route* to the source node. The node from which it received the RREQ is the next hop to the source node, and the hopcount in the RREQ is incremented by one to get the hop distance from the source. The node then checks whether it has an unexpired route to the destination. If it does not have a valid route to the destination, it simply rebroadcasts the RREQ, with the incremented hop count value, to its neighbors. In this manner, the RREQ floods the network in search of a route to the destination. Figure 1.2(a) illustrates this flooding procedure.

When a node receives a RREQ, it checks whether it has an unexpired route to the destination. If it does have such a route, then one other condition must hold for the node to generate a reply message indicating the route. The node's route table entry for the destination must have a corresponding sequence number that is at least as great as the indicated destination sequence number in the route request. That is,

$$dseq_r \geq dseq_{RREQ}$$

When this condition holds, the node's route table entry for the destination is at least as recent as the source node's last known route to the destination. This condition ensures the most recent route is selected, and also guarantees loop freedom (see [47] for a proof). Once this condition is met, the node can create a *route reply* (RREP) message. The RREP contains the source node's IP address, the destination node's IP address, and the destination's sequence number as given by the node's route table entry for the destination. In addition, the hop count field in the RREP is set equal to the node's distance from the destination. If the destination itself is creating the RREP, the hop count is set equal to zero. After creating the reply, the node unicasts the message to its next hop towards the source node. Thus, the reverse route that was created as the RREQ was forwarded is utilized to route the RREP back to the source node.

When the next hop receives the RREP, it first creates a *forward route* entry for the destination node. It uses the node from which it received the RREP as the next hop toward the destination. The hop count for that route is the hop count in the RREP,

incremented by one. This forward route entry for the destination will be utilized if the source selects this path for data packet transmissions to the destination. Once the node creates the forward route entry, it forwards the RREP to the destination node. The RREP is thus forwarded hop by hop to the source node, as indicated in figure 1.2(b). Once the source receives the RREP, it can utilize the path for the transmission of data packets. If the source receives more than one RREP, it selects the route with the greatest sequence number and smallest hop count.

Once a route is established, it must be maintained as long as it is needed. A route that has been recently utilized for the transmission of data packets is called an *active* route. Because of the mobility of the nodes, links along paths are likely to break. Breaks on links that are not being utilized for the transmission of data packets do not require any repair; however, breaks in active routes must be quickly repaired so that data packets are not dropped. When a link break along an active path occurs, the node upstream of the break (i.e., closer to the source node) invalidates the routes to each of those destinations in its route table. It then creates a *route error* (RERR) message. In this message it lists all of the destinations that are now unreachable due to the loss of the link. After creating the RERR message, it sends this message to its upstream neighbors that were also utilizing the link. These nodes, in turn, invalidate the broken routes and send their own RERR messages to their upstream neighbors that were utilizing the link. The RERR message thus traverses the reverse path to the source node, as illustrated in figure 1.2(c). Once the source node receives the RERR, it can repair the route if the route is still needed.

AODV contains a number of optimizations and optional features [45]. To improve the protocol performance and reduce overhead, source nodes can utilize an *expanding ring search* to search for routes to the destination. The propagation of the RREQ is controlled by modifying the time to live (TTL) value of the packet. Incrementally larger areas of the network are searched until a route to the destination is discovered. If a route to the destination can be found in the local area, a network-wide flood can be avoided.

Another optimization is the local repair of link breaks in active routes. When a link break occurs, instead of sending a RERR to the source, the node upstream of the break can try to repair the link locally itself. If successful, fewer data packets are dropped because the route is repaired more quickly. If the local repair attempt is unsuccessful, a RERR message is sent to the source node as previously described.

In addition to these optimizations, AODV contains a number of optional features to improve operation in a wide range of scenarios. For instance, during route discovery, if only intermediate nodes respond and the destination never receives a copy of the RREQ, the destination will not necessarily have a route to the source node. If two-way conversation with the destination is desired, this lack of route from the destination to the source can be problematic. Hence, AODV defines a *gratuitous RREP* that can be sent to the destination node when a RREP is created by an intermediate node. This gratuitous RREP informs the destination of a route to the source, as if the destination had performed a route discovery. Another optional feature is the *RREP acknowledgment* (RREP-ACK). When uni-directional links are suspected, the RREP-ACK can be utilized to ensure the next hop received the RREP. If an RREP-

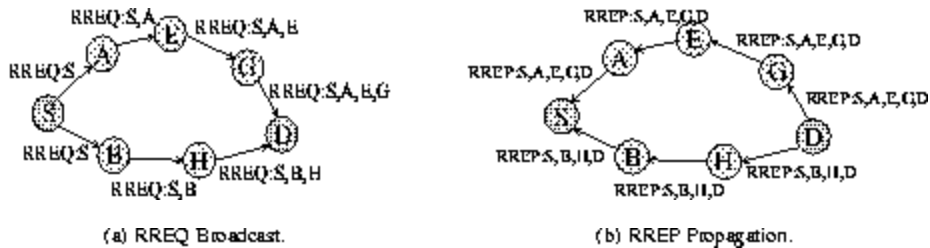


Fig. 1.3 DSR, Route Discovery.

ACK is not received, *blacklists* can be utilized to indicate uni-directional links so that these links are not used in future route discoveries. In addition, AODV allows the use of period *Hello* messages for monitoring connectivity to neighboring nodes.

1.3.2 Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol [24] is similar to AODV in that it is a reactive routing protocol with a route discovery cycle for route-finding. However, it has a few, important, differences.

One of the primary characteristics of DSR is that it is a source routing protocol; instead of being forwarded hop-by-hop, data packets contain strict source routes that specify each node along the path to the destination. Route request (RREQ) and route reply (RREP) packets accumulate source routes so that once a route is discovered, the source learns the entire source route and can place that route into subsequent data packets. Figure 1.3(a) illustrates the process of route discovery. The source node places the destination IP address, as well as its own IP address, into the RREQ and then broadcasts the message to its neighbors. When the neighboring nodes receive the message, they update their route to the source and then append their own IP addresses to the RREQ. Thus, as the RREQ is forwarded throughout the network, the traversed path is accumulated in the message. When intermediate nodes receive the RREQ, they can create or update routing table entries for each of the nodes listed in the source route, not just the source node.

When a node with a route to the destination receives the RREQ, it responds by creating a RREP. If the node is the destination, it places the accumulated source route from the RREQ into the RREP. Otherwise, if the node is an intermediate node, it concatenates its source route to the destination to the accumulated route in the RREQ, and places this new route into the RREP. Hence, in either scenario the message contains the full route between the source and the destination. The source route in the RREP is reversed and the RREP is unicast to the source. Note that as intermediate nodes receive and process the RREP, they can create or update routing table entries to each of the nodes along the source route. Figure 1.3(b) illustrates the propagation

of two RREPs back to the source. When a link break in an established path occurs, the node upstream of the break creates a route error (RERR) message and sends it to the source node.

Instead of maintaining a route table for tracking routing information, DSR utilizes a *route cache*. The cache allows multiple route entries to be maintained per destination, thereby enabling *multi-path* routing, as will be discussed in section 1.7.1. When one route to a destination breaks, the source can utilize alternate routes from the route cache, if they are available, to prevent another route discovery. Similarly, when a link break in a route occurs, the node upstream of the break can perform *route salvaging*, whereby it utilizes a different route from its route cache, if one is available, to repair the route. However, even when route salvaging is performed, a RERR message must still be sent to the source to inform it of the break.

Other characteristics that distinguish DSR from other reactive routing protocols include the fact that DSR's route cache entries need not have lifetimes. Once a route is placed in the route cache, it can remain there until it breaks. However, timeouts, capacity limits, and cache-replacement policies have been shown to improve DSR's performance [20]. Additionally, DSR nodes have the option of *promiscuous listening*, whereby nodes can receive and process data and control packets that are not addressed, at the MAC layer, to themselves. Through promiscuous listening, nodes can utilize the source routes carried in both DSR control messages and data packets to gratuitously learn routing information for other network destinations. Finally, to reduce the overhead of carrying source routes in data packets, DSR also allows flow state to be established in intermediate nodes. This flow state effectively allows hop-by-hop forwarding with the same source-based route control, as provided by the source route [21].

For a detailed study of the performance of AODV and DSR, as well as an explanation of how the protocol differences result in performance differentials, the reader is referred to [16].

1.4 GEOGRAPHICAL APPROACHES

Geographical approaches build on the proactive or reactive techniques previously described and in addition incorporate geographical information to aid in routing [3, 25, 33, 58, 59]. This geographical information can be in the form of actual geographic coordinates (as obtained through the global positioning system (GPS)), or can be obtained through reference points on some fixed coordinate system. The use of geo-location information can prevent network-wide searches for destinations, as either control packets or data packets can be sent in the general direction of the destination if the recent geographical coordinates for that destination are known. This reduces the control overhead generated in the network; however, all nodes must have continual access to their geographical coordinates for these approaches to be useful. The following describes one such geographical routing approach.

1.4.1 Location-Aided Routing

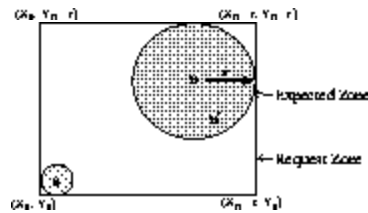


Fig. 1.4 LAR, Expected and Request Zone.

The Location-Aided Routing (LAR) protocol [25] is a reactive routing protocol that utilizes geographical coordinates to direct route request messages to the previously known location of the destination. The protocol defines two areas: the *expected zone*, and the *request zone*. The expected zone is the area in which the destination is most likely to be discovered. To calculate this area, the source must know a previous location of the destination at time t_0 , as well as an estimate of the velocity, v , at which the destination was traveling at t_0 . If the current time is t_1 , the expected zone can be calculated as a circle of radius $v(t_1 - t_0)$ centered at D . The request zone is the area in which the route request for the destination should propagate. In order to have the greatest probability of finding the destination, the request zone is defined to be the smallest rectangle that contains both the expected zone and the source node. Figure 1.4 illustrates an example expected and request zone.

The basic route discovery procedure of LAR is similar to that of other reactive routing protocols. When a source needs a route to a destination, it creates a route request (RREQ) message for that destination. If the source recently had a route to the destination, then the source calculates the expected zone and the request zone, and places the coordinates of the request zone boundary into the RREQ message. If the source does not have any previous information about the destination, then it is unable to calculate the expected and request zones. In this case the algorithm defaults to basic flooding.

When a node receives the RREQ, it processes it as described in the previous sections with the following exception. The node first determines whether it lies in the request zone defined in the RREQ. Because every node knows its current geographical coordinates, it can easily make this determination. If the node does not lie within the request zone, then it does not process the packet. Otherwise, if it does lie within the request zone, it processes the packet and either rebroadcasts it or sends a reply, depending on whether it has a current route to the destination.

The size of the request zone is a tradeoff between control overhead and probability of finding the destination. A small request zone runs the risk of not including the area in which the destination is currently located. It is also possible that, while the destination may lie within the request zone, the path between the source and the destination may not be completely contained within this zone. In this case, a route to

the destination will not be discovered. On the other hand, if the request zone is too large, the control overhead savings will be minimal.

In addition to the previously described approach, LAR also defines a second method for determining the request zone. Instead of calculating a rectangular area, the source places its distance from the previous location of the destination, along with the coordinates of the destination's previous location, in the RREQ. When neighboring nodes receive the RREQ, they calculate their distance from the destination ($DIST_i$) and then compare that value with the source's distance as reported in the RREQ ($DIST_S$). For some parameter δ , if $DIST_S + \delta \geq DIST_i$, then the node i processes the request. When it forwards the request, the node replaces $DIST_S$ in the RREQ with its distance $DIST_i$. On the other hand, if $DIST_S + \delta < DIST_i$, the node discards the RREQ. In practical implementations, δ typically equals 0. By using this approach, the nodes are forcing the RREQ to make forward progress to the estimate of the destination's location.

In both approaches, the RREQ is prevented from flooding the entire network because it is restricted to areas that are likely to be en route to the destination. This results in a reduction in both bandwidth and processing overhead.

1.5 HYBRID APPROACHES

The characteristics of proactive and reactive routing protocols can be integrated in various ways to form *hybrid* networking protocols. Hybrid networking protocols may exhibit proactive behavior given a certain set of circumstances, while exhibiting reactive behavior given a different set of circumstances. These protocols allow for flexibility based on the characteristics of the network. Hybrid approaches include the Zone Routing Protocol [41] and the Distance Routing Effect Algorithm for Mobility [3].

1.5.1 Zone Routing Protocol

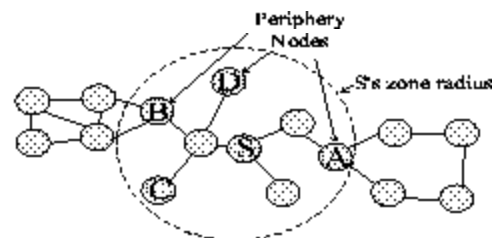


Fig. 1.5 ZRP Zone Radius.

The Zone Routing Protocol (ZRP) [41] integrates both proactive and reactive routing components into a single protocol. Around each node, ZRP defines a *zone* whose radius is measured in terms of hops. Each node utilizes proactive routing within its

zone and reactive routing outside of its zone. Hence, a given node knows the identity of and a route to all nodes within its zone. When the node has data packets for a particular destination, it checks its routing table for a route. If the destination lies within the zone, a route will exist in the route table. Otherwise, if the destination is not within the zone, a search to find a route to that destination is needed. Figure 1.5 illustrates the zone concept. In this figure, the zone radius is two hops.

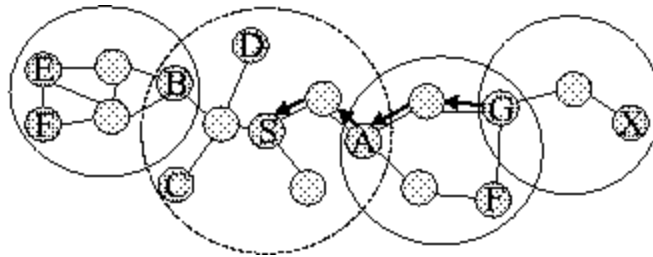


Fig. 1.6 Example ZRP Route Discovery.

For intra-zone routing, ZRP defines the Intrazone Routing Protocol (IARP). IARP is a link state protocol that maintains up-to-date information about all nodes within the zone. For any given node X , X 's *peripheral nodes* are defined to be those nodes whose minimum distance to X is the zone radius. In figure 1.5, S 's peripheral nodes are nodes A , B , C , and D . These peripheral nodes are important for reactive route discovery. ZRP utilizes the Interzone Routing Protocol (IERP) for discovering routes to destinations outside of the zone. For route discovery, the notion of *bordercasting* is introduced. Once a source node determines the destination is not within its zone, the source *bordercasts* a query message to its peripheral nodes. During the bordercast, the query message is relayed toward these peripheral nodes using trees constructed within the intrazone topology. After receiving the message, the peripheral nodes, in turn, check whether the destination lies within their zone. If the destination is not located, the peripheral nodes in turn bordercast the query message to their peripheral nodes. This process continues until either the destination is located, or until the entire network is searched. Once a node discovers the destination, it unicasts a reply message to the source node.

Figure 1.6 illustrates an example of the bordercast discovery procedure. In the figure, node S performs a query for the destination X . By using the IARP, it learns that X is not within its zone. It bordercasts the query message to its peripheral nodes. In the figure, the dotted circle represents the radius of S 's zone. The peripheral nodes in turn check their zone, and after not finding the destination, bordercast the query message to their peripheral nodes. The solid circles in the figure represent the forward propagation of the query messages to each node's peripheral nodes (i.e., the circles do not enclose nodes that have already received the query). Hence only the portion of each node's zone that have not been previously traversed by the query message is shown. Eventually, node G discovers X within its zone, and hence unicasts a reply back to node S .

To improve query efficiency, a random query processing delay can be used as an effective query control mechanism. By waiting a random interval between query reception and query forwarding, the chance of collisions during forwarding is reduced, and therefore the effectiveness of the protocol is improved. In addition, ZRP defines other optimizations to reduce the messaging and processing overhead [19]. In particular, these include early termination of queries by preventing a query from propagating into a zone that has already been searched for a destination.

Recently, a new version of ZRP, ZRPv2, has been introduced [54]. ZRPv2 differs from the original ZRP in the manner in which bordercasting is performed. In both versions, route discovery is initiated with the query source node constructing a bordercast tree to its *uncovered* peripheral nodes. An *uncovered* node is one that does not belong to the routing zone of a node that already has received the query. The node then forwards the query message to its bordercast tree neighbors. When these neighbors receive the query message, rather than forwarding the message to the query source's downstream peripheral nodes (as in the original ZRP), they each construct bordercast trees to their own uncovered peripheral nodes, and forward the route query to their bordercast tree neighbors. Each node that receives a route query follows the above procedure until the destination, or a node possessing a fresh route to the destination, is reached. At that point a route reply is unicast back to the source.

Performing bordercasting on a hop-by-hop basis yields a uniform, and thus simpler, protocol implementation. Also, the need for maintaining an extended routing zone is eliminated.

1.6 CLUSTERING AND HIERARCHICAL ROUTING

IP addresses are hierarchical in that they identify the location of the end device within the global Internet. Routing protocols in the Internet take advantage of this hierarchy when determining routes between networks. Ad hoc networks, on the other hand, are not necessarily able to take advantage of a hierarchy based on IP addresses. Nodes joining an ad hoc network are likely to have pre-assigned IP addresses from other networks. Hence, the nodes form a hodge-podge of addresses, and hierarchical routing based on addresses is not necessarily possible.

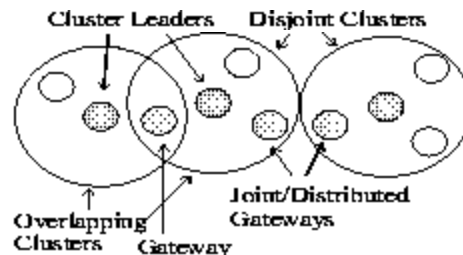


Fig. 1.7 Example Cluster Configuration.

However, flat routing, as performed by the previously discussed protocols, has a number of disadvantages. The primary disadvantage is that it is not scalable. In the worst case, a node must maintain a routing table entry for every other node in the network, and the amount of information exchanged to create these routing table entries is $O(n^2)$, where n is the number of nodes in the network.

To increase the scalability of the ad hoc network, hierarchical, or clustering, protocols, can be utilized. Hierarchical protocols place nodes into groups, often called clusters. These groupings may be based on a number of criteria, but most commonly they are based on either location [1, 2, 4, 5] or functionality [43, 60].

There have been numerous hierarchical routing protocols developed that take a variety of approaches towards clustering. In this section, a survey of the characteristics and algorithms for clustering is given; individual protocols are not examined.

The physical properties of the clusters vary between clustering protocols. As shown in figure 1.7, clusters can be either overlapping, or completely disjoint. Further, a one-level hierarchy can be created, or recursive multi-level hierarchies are also possible. Finally, control within a cluster can be held by a cluster leader, or the cluster can be completely distributed with no cluster leader. In networks where cluster leaders exist, these leaders typically process control packets on behalf of their member nodes. It is also possible for cluster leaders to form a routing backbone within the network [4, 12]. This can be a desirable property if the cluster leaders are pre-selected as nodes with greater resources than non-leader nodes.

Figure 1.7 illustrates an example cluster topology with cluster leaders. The cluster boundaries are based on the transmission range of the cluster leaders. All nodes within a cluster must be within direct transmission range of the cluster leader. In this example, cluster boundaries are allowed to overlap. Nodes that are located within the boundaries of multiple clusters are called *gateways*. These nodes serve as routers between the two clusters. With many clustering protocols, it is also possible for pairs of nodes to serve as *distributed*, or *joint* gateways. Distributed gateways are a pair of nodes that are within direct transmission range of each other, where each node lies in a different cluster. Together, the two nodes can be used to route between clusters.

Cluster leaders are typically initialized through some distributed algorithm. For instance, there can be a leader election algorithm, where the node with the highest ID within some area becomes the leader for that area [12]. Alternatively, weights, such as number of neighbors, transmission range, etc., can be used instead of the ID of the node [2, 7]. Other algorithms take a more "first come, first elected" approach [4]. In this method, when a node joins a network, it queries its one-hop neighbors to determine whether it is within range of an existing cluster leader. If so, the node may choose to join the already existing cluster. Otherwise, if there is not a nearby cluster leader, the node becomes a leader itself. Using this approach, during the initialization of a network, the first node to join the network would become a cluster leader.

Once the network has been initialized and cluster leaders have been established, there must be leader selection and revocation algorithms in place. The leader selection algorithm specifies under what conditions a node should become a cluster leader. For instance, if a node wanders to the periphery of the network, it may lose contact with all current cluster leaders. In this case, it may become a cluster leader

itself. Similarly, the network must also have in place a cluster revocation algorithm. Without such an algorithm, there will be a slow, continual growth in the number of leaders as more nodes wander to the network perimeter and become leaders. In the worst case, without a revocation algorithm, it would be possible for each network node to eventually become a leader. If all nodes became leaders, the hierarchy would become ineffective.

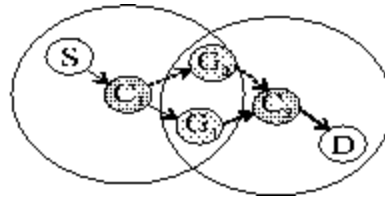


Fig. 1.8 Cluster Routing Example.

There are a number of leader revocation algorithms. Most commonly, when two leaders come within direct transmission range of each other, one of the leaders must give up its leader status. The leader to give up its status may be the leader with the lowest ID [12], or a weight-based approach may be used [2, 7]. An alternate method has been proposed in [4], where instead of requiring one node to become a non-leader whenever two leaders come within transmission range, a leader gives up its leader status only when its cluster becomes a *subset* of the other cluster. This approach has some beneficial properties such as preventing a rippling effect, where one leadership change results in further network leadership changes. Because leadership changes are expensive in terms of control overhead and routing changes, they should be minimized overall.

There are two key benefits from utilizing clustering protocols in an ad hoc network. The first of these is that they enable hierarchical routing. Consider the network in figure 1.8. Suppose a source node S wants to send packets to the destination D . The path it might discover with one of the previously discussed flat routing protocols is

$$S \rightarrow C_1 \rightarrow G_1 \rightarrow C_2 \rightarrow D$$

Using a flat routing scheme, whenever any link along this path breaks, the route must be repaired with a route maintenance procedure (i.e. sending a route error to the source, local repair, route salvaging, etc.). However, with a hierarchical routing scheme, the path is instead recorded as

$$S \rightarrow C_1 \rightarrow C_2 \rightarrow D$$

The difference is that, with the hierarchical route, the path is recorded at the cluster level. Hence the route is recorded from cluster leader C_1 to C_2 , and the intermediate node is not specified. To get from C_1 to C_2 , any gateway node connecting the two clusters can be utilized. For instance, gateway G_1 may be selected to route between the clusters. In the event that G_1 wanders out of transmission range of one of the clusters and can no longer serve as a gateway, one of the other gateways (i.e., G_2) can be used instead. Because of this increased routing flexibility, link breaks do not

necessarily result in route repairs if another gateway node is available. Decreasing the number of route repairs decreases the amount of control overhead generated in the network, and consequently increases the number of data packets that can be delivered to the destination.

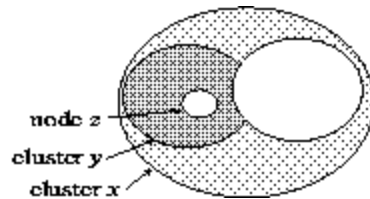


Fig. 1.9 Hierarchical Addressing.

The second benefit of hierarchical protocols is that the hierarchy can be used to implement hierarchical addressing schemes. Addresses can be assigned to nodes based on their cluster membership. For instance, in the network shown in figure 1.9, a node z is a member of a cluster y . In that case, its address may be $y.z$. If the hierarchy were multiple levels, then cluster y might in turn be within a larger cluster x . In this case the node's address would be $x.y.z$. If the node were a member of multiple clusters, the node could have multiple, concurrent addresses. In mobile networks, multiple addresses would be beneficial because at any given time, at least one of the addresses is likely to still be valid.

As was seen in the previous example, multi-level hierarchies can be created. Multi-level hierarchies that dynamically adjust their depth based on the network topology can further increase the scalability of the network. In multi-level hierarchies, each cluster becomes a node at the next highest cluster level. Routes can be recorded up and down the hierarchical routing tree. A route ascends the hierarchical tree only as high as needed to reach the branch containing the destination. Multi-level hierarchical routing protocols include [5, 28, 44, 52, 56].

Hierarchical routing protocols have many clear advantages. They improve route robustness by increasing routing flexibility; routes that are recorded between clusters, as opposed to between nodes, have more routing options, and hence can be repaired more easily. Increasing route robustness leads to an increase in route lifetimes, thereby resulting in fewer route reconstructions, less control traffic from route repairs, and increased data delivery. However, there are also disadvantages that many hierarchical routing protocols suffer from. To create and maintain the clusters, many clustering protocols require periodic overhead [1, 4, 64]. Such overhead is needed to maintain current information about cluster memberships and gateway availability. To overcome this drawback, some clustering approaches have opted for a more on-demand approach, such that clusters are only created when needed [18]. This can eliminate a significant fraction of the overhead in networks with low traffic demands. Other disadvantages include the centralization of routes through cluster leaders. If the cluster leaders can be selected solely from nodes with greater resources, than this centralization is likely to be beneficial. However, in networks without spe-

cialized, high power nodes, centralizing routes unfairly taxes leader nodes, and is likely to result in premature depletion of their batteries. Protocols that utilize cluster leaders for the establishment of routes but that do not actually require cluster leaders to participate on the routes can eliminate this unfair burden. Finally, the cluster leader-gateway-cluster leader routing requirement can result in the use of non-shortest paths.

1.7 OTHER TECHNIQUES

1.7.1 Multi-path Routing

The majority of routing protocols previously described utilize route tables for storing routing information. In such tables, there is one next hop entry for each destination. While in the IP route table it is only possible to store one route entry for each destination, it is possible in the routing protocol to create a route cache, in which multiple routing entries per destination can be maintained. In the event that the path in use to some destination breaks or becomes otherwise unusable, an alternate route in the route cache can be utilized instead. When alternate routes are available, route discoveries can be saved, and control overhead can be reduced.

There are numerous proposals for multi-path routing. The vast majority of these study multipath routing with on-demand routing approaches. The DSR protocol utilizes route caches for maintaining multiple paths to each destination. In [20], various caching strategies and cache configurations are analyzed.

Other proposals investigate modifications to existing protocols to add multi-path routing capability. For instance, the Ad hoc On-Demand Multipath Distance Vector (AOMDV) routing protocol is described in [36]. In AOMDV, multiple routing entries are stored per destination, and there is one lifetime value associated with all the routing entries for a destination. Hence, all the routing possibilities are refreshed or expire at the same time. Also, the emphasis in AOMDV is on finding multiple routes that are link disjoint, implying that the routes do not share any common links.

Another approach is taken in AODV-BR [29], which proposes back-up paths for routing around link breaks in active routes. When a link break in an active route occurs, the node closer to the source sends an error message to the source, and also broadcasts the data packet to its neighbors. The packet indicates in the data header that the link has broken and that the packet is in need of alternate routing. When the neighboring nodes receive the packet, they forward it to the destination if they have a route for that destination.

There are additional proposals that describe new protocols to handle multi-path routing [30, 39, 50]. One of these, the Split Multipath Routing (SMR) protocol [30], operates similarly to the reactive protocols described in section 1.3; however, the protocol makes the following modifications. To obtain maximal node disjoint paths, intermediate nodes are not allowed to respond to route requests. Further, route requests and route replies contain entire routing paths. Intermediate nodes forward the first route request they receive, and only rebroadcast subsequent ones if the requests

arrive from different neighbors and have not traveled further from the source than the first route request received. When the destination receives the request, it responds to the first request, and then waits a time interval to receive additional requests. At the end of the time interval, it responds to the request that traveled the route most disjoint from the original request received. The protocol can be easily configured so that the destination responds to more than two requests.

Finally, [42] investigates the impact of alternate path routing on ad hoc routing protocols, particularly focusing on load balancing and the concurrent utilization of multiple routes. In particular, it studies the impact of *route coupling* in ad hoc networks with only a single available channel. Route coupling occurs when two routes have nodes or links in common. The authors discover that, while alternate routes enable a reduction in end-to-end delay, the route coupling results in an under-utilization of network resources, thereby preventing alternate path routing from achieving significant performance improvements.

1.7.2 Energy-conserving Protocols

All of the routing protocols previously described in this chapter were designed with the characteristics of a mobile environment in mind. Hence, each protocol attempts to reduce control overhead and processing requirements so as to minimize power utilization. However, there is a set of ad hoc routing protocols that have been designed with the specific goal of further minimizing energy consumption. These protocols take a variety of approaches towards energy efficiency, including powering down un-utilized nodes, load balancing, and dynamic transmission power adjustment.

The Geographical Adaptive Fidelity (GAF) algorithm [61] is an example of an energy-conserving protocol that powers down un-utilized nodes. GAF is able to identify *routing-equivalent* nodes in a network so that unnecessary nodes can be turned off. GAF nodes utilize location information, such as through GPS, to create a virtual grid. All nodes within a given grid square are equivalent with respect to routing functionality. Nodes in the grid square can therefore coordinate to determine the sleep duration for each node. The nodes can be coordinated such that load balancing aids in the overall energy utilization of each node; each node takes a turn forwarding data packets.

Routing based on overall energy cost and remaining battery lifetimes is performed by the Battery Energy Efficient (BEE) protocol [13]. This protocol assigns to each route a cost function that takes into account both energy cost and battery lifetime. When a source initiates a data stream, it evaluates the cost function for all the possible loop-free routes to the destination. It selects the route that minimizes the cost function. Similar to this work, [11] describes an approach that balances path selection and energy consumption rates among the network nodes in proportion to the available energy reserves of the nodes.

A number of protocols dynamically adapt node transmission ranges to reduce overall power consumption. For instance, the approach described in [17] computes the power cost along the discovered paths from the source to the destination. The path with the minimum such cost is selected. The network nodes use a power man-

agement scheme such that nodes are grouped into clusters, and each node transmits at the minimum power level to reach each of the nodes in the cluster. Along these same lines, the protocol proposed in [51] dynamically adjusts node transmit powers to maintain a connected topology using minimum power. Nodes in dense portions of the network decrease their transmission power to reach fewer nodes, while nodes in remote areas of the network increase their transmission power to become more fully connected. It should be noted, however, that increasing the number of hops in a path has the drawback of increasing the likelihood of a link break in that path. This can result in an increase in control overhead due to the additional route repairs.

In addition to routing protocols, there is a wide range of research on power-efficient MAC protocols and transport layer protocols. For instance, the Power-Aware Multiple Access protocol with Signalling (PAMAS) [57] utilizes information gleaned from RTS/CTS exchanges to turn off nodes when they are not receiving packets. Because the RTS and CTS messages contain the length of an upcoming data packet, nodes receiving this message that are not the sender or receiver of the data packet can safely turn off their interfaces while the data packet is being transmitted.

1.7.3 Security-Aware Protocols

The open nature of wireless communication and the portability of mobile devices creates a challenge for securing mobile networks. In contrast to wired networks, intruders do not need to compromise network hosts to gain access to the network; malicious nodes need to only be within transmission range of a node in order to participate in and overhear network traffic. Although securing ad hoc networks is a challenge, it is a necessary component for ad hoc networks to be universally deployed. Recent ad hoc network security research has addressed a wide range of security topics, from secure routing, to intrusion detection and monitoring. In this section a sampling of approaches from these two areas is highlighted.

1.7.3.1 Secure Routing When performing an on-demand route discovery such as those described in section 1.3, there are a number of possible attacks. Malicious nodes can lie about the existence of paths and can modify the information in routing messages to influence path selection. Further, a source node has no method for determining whether a path actually exists when it receives a route reply. However, in nearly every application scenario a user would like to be assured that his or her data traffic is actually reaching its intended destination. To address this issue, a number of *secure routing protocols* have been developed. Some of these protocols attempt to secure existing routing protocols, such as those described in section 1.3. Others are new routing protocols that have been designed with the primary goal of security.

An example of this latter type of protocol is the Authenticated Routing for Ad hoc Networks (ARAN) protocol [55]. ARAN is based on certificates and assumes that all network nodes can obtain a certificate from a trusted certificate server before joining the ad hoc network. The certificate contains the public key of the node. ARAN utilizes a route discovery procedure similar to AODV. A source node S generates a Route Discovery Packet (RDP) for a destination. The RDP is signed with the

source's private key and contains its certificate. When a neighbor A receives the RDP message, it verifies the signature of the source by extracting S 's public key from its certificate, and then sets up a reverse path back to the source node. The node then signs the contents of the message, appends its own certificate, and broadcasts the message to its neighbors. When A 's neighbor B receives the message, it validates A 's signature, and then replaces this signature with its own signature (the signature of the source node is retained). The packet continues to be rebroadcast in this manner until it reaches the destination. When the first RDP reaches the destination, the destination node verifies the signature of the source node and then sends a digitally signed Route Reply packet (REP) back to the source. The REP travels the same path as the RDP, and the same signing procedure is performed by intermediate nodes. Because the destination must sign the REP message, only the destination is allowed to respond to the RDP. Also, because RDP messages are signed at each hop and do not contain a hop count or a source route, malicious nodes have no opportunity to intentionally redirect traffic.

The Secure Routing Protocol (SRP) [40] is another approach to secure routing that is based on the assumption of the existence of a security association between the source and destination node. To initiate communication, the two nodes negotiate a shared secret key. A message authentication code (MAC) is used to ensure that the reply message is not modified en route to the source node. Only the destination can respond to route query messages, and the source is assured that the destination was reached because the shared key is used as input to the MAC computation.

Like SRP, Ariadne assumes that all pairs of communicating nodes have secret MAC keys [22]. Each pair of nodes maintains two keys, one for each direction of communication. Ariadne utilizes symmetric cryptographic primitives and is based on the DSR routing protocol and the TESLA broadcast authentication protocol [49]. Ariadne has the properties that source and destination nodes can authenticate each other due to the secret keys, and that the source node can authenticate each entry on the path returned in the route reply. In addition, a one-way function is utilized so that no intermediate node can remove a previous node in the source route contained in route request and reply messages.

The Security Aware Routing (SAR) protocol described in [62] relies on trust levels to provide security. Nodes form a trust hierarchy where each node is assigned a specific trust level. Designed to run over a reactive routing protocol such as AODV or DSR, route request and reply messages are assigned a security level by the source node. Only nodes with at least the indicated level of security can process and forward the control messages. Hence, SAR discovers routes where all nodes along the path meet the desired level of security.

Finally, a mechanism for securing the AODV protocol is presented in [63]. The protocol, SAODV, utilizes digital signatures and hash chains for securing AODV control messages. The digital signatures are utilized to authenticate the non-mutable fields of the control messages, while the hash chains are used to secure the hop count information. The approach assumes the nodes have access to a key management system so that the nodes can obtain the public keys of the other nodes within the network.

1.7.3.2 Intrusion Detection and Monitoring Schemes Intrusion detection is a mechanism widely used in wired networks to detect malicious invaders and trigger an appropriate response. Intrusion detection in ad hoc networks is somewhat less straightforward because membership in the network is open to virtually any user. Hence, it is difficult to detect when a user is actually a malicious intruder. Nevertheless, intrusion detects techniques can be employed in ad hoc networks to detect misbehaving nodes, particularly in networks in which the membership is well-defined. Such networks include military networks and collaborative networks comprised of a team of individuals.

An intrusion detection and response mechanism is presented in [65]. It uses the cooperative statistical anomaly detection model to protect against attacks on routing protocols or other wireless applications and services. Each intrusion detection system (IDS) agent runs independently and monitors local activities. Intrusions are detected from local traces, and responses are subsequently initiated. If an anomaly is detected in the local data, or if the evidence is inconclusive and a broader search is warranted, neighboring IDS agents cooperate to participate in global intrusion detection actions.

Another approach taken by a handful of protocols is to monitor node behavior to detect misbehaving nodes. Node misbehavior can come in many forms; however, a common action to monitor is whether a node en route to a destination forwards data packets that it receives for that destination. For instance, the monitoring system described in [37] uses nodes called watchdogs to monitor the forwarding of data packets by intermediate nodes. After a node transmits a data packet, it *promiscuously* listens to determine whether the next hop along the path forwards the data packet to its next hop. For this functionality to work, a node must know the identity of the node two hops further along the path. In addition to the watchdog, network nodes also run a pathrater module to determine the reliability of paths. Nodes maintain ratings for other network nodes, and hence select paths with the highest aggregate rating. The watchdog system is used to maintain the rating for neighboring nodes.

A similar monitoring system is described in [9]. This approach incorporates a trust manager and reputation system with a path manager to detect misbehaving, or non-conforming, nodes and exclude them from routing. The difference with this approach is that nodes propagate information about detected misbehaving nodes so that those nodes can be excluded from participation in the network. When a node receives such a message indicating the misbehavior of another node, the node must be able to authenticate the source of that message. This prevents denial of service attacks by malicious nodes against other, benign nodes.

1.7.4 Conclusion

As has been shown in this chapter, there exists a vast variety of routing protocols designed specifically for ad hoc mobile networks. These networks create a hostile routing environment due to the mobility of the nodes and the resulting ephemeral nature of the network links. However, significant strides have been made towards the

development of robust routing protocols that can deliver high percentages of traffic, even in dynamic environments.

It is likely that there does not exist a single routing protocol that can solve the needs of every conceivable ad hoc network scenario. Rather, the selection of a routing protocol for a given network is likely to be dependent upon the dominating characteristics of that network. Hence, certain routing protocols are likely to perform best in networks of one set of characteristics, while others will perform better in networks with a differing set of characteristics. More work is needed to identify the sets of characteristics that promote the optimum behavior of each individual protocol and class of protocols.

1.7.5 Acknowledgments

This work is largely based on a tutorial on Mobile Ad hoc Networks created jointly by the author and Sung-Ju Lee of Hewlett-Packard Laboratories. The author would like to thank Thomas Clausen, Zygmunt Haas, Yih-Chun Hu, Sung-Ju Lee, Richard Ogier, Marc Pearlman, Prince Samar, and Fried Templin for their insightful comments and contributions to the protocol descriptions in this chapter.

References

1. D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.
2. S. Basagni. Distributed Clustering for Ad Hoc Networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310–315, Australia, June 1999.
3. S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 76–84, Dallas, TX, October 1998.
4. E. M. Belding-Royer. Hierarchical Routing in Ad hoc Mobile Networks. *To appear in the Wireless Communications and Mobile Computing*, 2002.
5. E. M. Belding-Royer. Multi-Level Hierarchies for Scalable Ad hoc Routing. *To appear in Wireless Networks*, 2002.
6. B. Bellur, R. G. Ogier, and F. L. Templin. Topology Broadcast Based on Reverse-Path Forwarding (TBRPF). *IETF Internet Draft, draft-ietf-manet-tbrpf-01.txt*, March 2001. (Work in Progress).
7. C. Bettstetter and R. Krausser. Scenario-Based Stability Analysis of the Distributed Mobility-Adaptive Clustering (DMAC) Algorithm. In *Proceedings of*

- the 2nd Annual Symposium on Mobile Ad hoc Networking and Computing*, Long Beach, California, October 2001.
8. J. Broch, D. A. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, Dallas, Texas, October 1998.
 9. S. Buchegger and J.-Y. L. Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth EuroMicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEE Computer Society.
 10. R. Castaneda and S. R. Das. Query Localization Techniques for On-demand Routing Protocols in Ad Hoc Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 186–194, Seattle, WA, August 1999.
 11. J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 22–31, Tel Aviv, Israel, March 2000.
 12. C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *Proceedings of IEEE Singapore International Conference on Networks (SICON)*, pages 197–211, April 1997.
 13. C.-F. Chiasserini and R. R. Rao. Routing protocols to maximize battery efficiency. In *Proceedings of IEEE MILCOM*, Los Angeles, CA, October 2000.
 14. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol. In *Proceedings of IEEE INMIC*, Lahore, Pakistan, December 2001.
 15. S. R. Das, R. Castaneda, and J. Yan. Comparative Performance Evaluation of Routing Protocols for Mobile, Ad hoc Networks. In *Proceedings of the 7th International Conference on Computer Communications and Networks*, pages 153–161, Lafayette, LA, October 1998.
 16. S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 3–12, Tel Aviv, Israel, March 2000.
 17. T. A. ElBatt, S. V. Krishnamurthy, D. Connors, and S. Dao. Power Management for Throughput Enhancement in Wireless Ad hoc Networks. In *Proceedings*

- of the *IEEE International Conference on Communications (ICC)*, pages 1503–1513, New Orleans, LA, June 2000.
18. M. Gerla, T. Kwon, and G. Pei. On Demand Routing in Large Ad hoc Wireless Networks with Passive Clustering. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, September 2000.
 19. Z. J. Haas and M. R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *ACM/IEEE Transactions on Networking*, 9(4):427–438, August 2001.
 20. Y.-C. Hu and D. B. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In *Proceedings of the Sixth Annual IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 231–242, Boston, MA, August 2000.
 21. Y.-C. Hu and D. B. Johnson. Implicit source routing in on-demand ad hoc network routing. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 1–10, Oct. 2001.
 22. Y.-C. Hu, D. B. Johnson, and A. Pettig. Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. In *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (Mobicom)*, Atlanta, GA, September 2002.
 23. P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 195–206, Seattle, WA, August 1999.
 24. D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.
 25. Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, Dallas, Texas, October 1998.
 26. R. Kravets and P. Krishnan. Application-Driven Power Management for Mobile Communication. *Wireless Networks*, 6(4):263–277, 2000.
 27. A. Laouiti, A. Qayyum, and L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'2002)*, Waikoloa, HI, January 2002.

30 REFERENCES

28. G. S. Lauer. Packet-Radio Routing. In M. Steenstrup, editor, *Routing in Communications Networks*. Prentice Hall, 1995.
29. S.-J. Lee and M. Gerla. AODV-BR: Backup Routing in Ad hoc Networks. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, Chicago, IL, September 2000.
30. S.-J. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3201–3205, Helsinki, Finland, June 2001.
31. S.-J. Lee, E. M. Royer, and C. E. Perkins. Ad hoc Routing Protocol Scalability. To appear in the *International Journal on Network Management*, 2002.
32. S.-J. Lee, C.-K. Toh, and M. Gerla. A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad-Hoc Networks. *IEEE Network*, 13(4):48–54, July/August 1999.
33. J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad hoc Routing. In *Proceedings of the 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, Boston, MA, August 2000.
34. J. Macker and M. S. Corson. Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter. <http://www.ietf.org/html.charters/manet-charter.html>.
35. G. S. Malkin and M. E. Steenstrup. Distance-Vector Routing. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 83–98. Prentice Hall, 1995.
36. M. Marina and S. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, Riverside, CA, November 2001.
37. S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 255–265, 2000.
38. J. Moy. Link-State Routing. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 135–157. Prentice Hall, 1995.
39. A. Nasipuri and S. Das. On-Demand Multipath Routing for Mobile Ad hoc Networks. In *Proceedings of the IEEE Conference on Computer Communications and Networks (ICCCN)*, pages 64–70, Boston, MA, October 1999.
40. P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad hoc Networks. In *Proceedings of the SCS Communication Networks and Distributed Systems*

- Modeling and Simulation Conference (CNSD 2002)*, San Antonio, TX, January 2000.
41. M. R. Pearlman and Z. J. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1395–1414, August 1999.
 42. M. R. Pearlman, Z. J. Haas, P. Sholander, and S. S. Tabrizi. On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad hoc Networks. In *Proceedings of the 1st Annual Workshop on Mobile and Ad hoc Networking and Computer (MobiHOC)*, pages 3–10, Boston, MA, August 2000.
 43. G. Pei, M. Gerla, and X. Hong. LANMAR: Landmark Routing for Large Scale Wireless Ad hoc Networks with Group Mobility. In *Proceedings of the 1st Annual Workshop on Mobile and Ad hoc Networking and Computer (MobiHOC)*, pages 11–18, Boston, MA, August 2000.
 44. G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A Wireless Hierarchical Routing Protocol with Group Mobility. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1538–1542, New Orleans, LA, September 1999.
 45. C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF Internet Draft, draft-ietf-manet-aodv-10.txt*, March 2002. (Work in Progress).
 46. C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM '94: Computer Communications Review*, 24(4):234–244, October 1994.
 47. C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.
 48. C. E. Perkins and E. M. Royer. The Ad hoc On-Demand Distance Vector Protocol. In C. E. Perkins, editor, *Ad hoc Networking*, pages 173–219. Addison-Wesley, 2000.
 49. A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2001.
 50. J. Raju and J. Garcia-Luna-Aceves. A New Approach to On-Demand Loop-Free Multipath Routing. In *Proceedings of the IEEE Conference on Computer Communications and Networks (ICCCN)*, pages 522–527, Boston, MA, October 1999.

32 REFERENCES

51. R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks using Transmit Power Adjustment. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 404–413, Tel Aviv, Israel, March 2000.
52. R. Ramanathan and M. Steenstrup. Hierarchically-organized, Multihop Mobile Wireless Networks for Quality-of-Service Support. *ACM/Baltzer Mobile Networks and Applications*, 3(1):101–118, 1998.
53. E. M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.
54. P. Samar, M. R. Pearlman, and Z. J. Haas. Hybrid Routing: The Pursuit of an Adaptable and Scalable Routing Framework for Ad Hoc Networks. In M. Ilyas, editor, *Handbook of Ad Hoc Wireless Networks*, chapter 14. CRC Press, 2002.
55. K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A Secure Routing Protocol for Ad hoc Networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.
56. N. Shacham. Hierarchical Routing in Large, Dynamic Ground Radio Networks. In *Proceedings of the 18th Hawaii International Conference on System Sciences*, pages 292–301, 1985.
57. S. Singh, M. Woo, and C. S. Raghavendra. Power-Aware Routing in Mobile Ad hoc Networks. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 181–190, Dallas, TX, October 1998.
58. I. Stojmenovic. Location Updates for Efficient Routing in Ad hoc Wireless Networks. *Handbook of Wireless Networks and Mobile Computing*, Wiley, pages 451–471, 2002.
59. I. Stojmenovic. Position Based Routing in Ad hoc Mobile Networks. *IEEE Communications Magazine*, 40(7):128–134, July 2002.
60. P. F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. In *Computer Communication Review*, pages 35–42, Stanford, CA, August 1988.
61. Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Ad hoc Routing. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–84, Rome, Italy, July 2001.
62. S. Yi, P. Naldurg, and R. Kravets. A Security Aware Routing Protocol for Wireless Ad Hoc Networks. In *Proceedings of the 6th World Multi-Conference on*

- Systemics, Cybernetics and Informatics (SCI)*, pages 286–292, Orlando, FL, July 2002.
63. M. G. Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, Atlanta, GA, September 2002.
 64. J. Zavgiæn. NTDR Mobility Management Protocols and Procedures. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 292–301, November 1997.
 65. Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad hoc Networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 26–33, Seattle, WA, August 1999.