5

DSR
The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks

David B. Johnson
Department of Computer Science
Rice University

David A. Maltz, Josh Broch
Carnegie-Mellon University

Abstract
The *Dynamic Source Routing* protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multihop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two mechanisms of *Route Discovery* and *Route Maintenance,* which work together to allow nodes to discover and maintain *source routes* to arbitrary destinations in the ad hoc network. The use of source routing allows packet routing to be trivially loop free, avoids the need for up-to-date routing information in the intermediate nodes through which packets are forwarded, and allows nodes that are forwarding or overhearing packets to cache the routing information in them for their own future use. All aspects of the protocol operate entirely *on demand,* allowing the routing packet overhead of DSR to scale *automatically* to only that needed to react to changes in the routes currently in use. We have evaluated the operation of DSR through detailed simulation on a variety of movement and communication patterns and through implementation and significant experimentation in a physical outdoor ad hoc networking testbed we have constructed in Pittsburgh, and we have demonstrated the excellent performance of the protocol. In this chapter, we describe the design of DSR and provide a summary of some of our simulation and testbed implementation results for it.

The *Dynamic Source Routing* protocol [Johnson 1994, Johnson+ 1996a, Broch+ 1999a] is a simple and efficient routing protocol designed specifically for use in multihop wireless ad hoc networks of mobile nodes. Using DSR, the network is completely self-organizing and self-configuring, requiring no existing network infrastructure or administration. Network nodes (computers) cooperate to forward packets for each other to allow communication over multiple "hops" between nodes not directly within wireless transmission range of one another. As nodes in the network move about or join or leave the network, and as wireless transmission conditions such as sources of interference change, all routing is automatically determined and maintained by DSR. Because the number or sequence of intermediate hops needed to reach any destination may change at any time, the resulting network topology may be quite rich and rapidly changing.

DSR allows nodes to dynamically discover a *source route* across multiple network hops to any destination in the ad hoc network. Each data packet sent then carries in its header the complete, ordered list of nodes through which the packet must pass, allowing packet routing to be trivially loop free and avoiding the need for up-to-date routing information in the intermediate nodes through which the packet is forwarded. With the inclusion of this source route in the header of each data packet, other nodes forwarding or overhearing any of the packets may also easily cache this routing information for future use.

This work is a part of the Monarch Project at Carnegie-Mellon University [Johnson+ 1996b, Monarch], which is a long-term study that is developing networking protocols and protocol interfaces to allow truly seamless wireless and mobile networking. The Monarch Project is named in reference to the migratory behavior of the monarch butterfly; it can also be considered as an acronym for "*Mo*bile *N*etworking *Arch*itectures." The scope of our research includes protocol design, implementation, performance evaluation, and usage-based validation, roughly ranging from portions of the ISO data link layer (layer 2) through part of the presentation layer (layer 6).

In designing DSR, we sought to create a routing protocol that has very low overhead yet is able to react quickly to changes in the network, providing highly reactive service to help ensure successful delivery of data packets in spite of node movement or other changes in network conditions. On the basis of our evaluations of DSR and other protocols to date, through detailed simulation and testbed implementation, we believe this goal has been well met [Johnson+ 1996a, Broch+ 1998, Maltz+ 1999a, Maltz+ 1999b]. In particular, in our detailed simulation comparison of routing protocols for ad hoc networks [Broch+ 1998], DSR outperformed the other protocols that we studied, and recent results by Johansson et al. [Johansson+ 1999] have shown generally similar results. The protocol specification for DSR has also been submitted to the Internet Engineering Task Force (IETF), the principal protocol standards development body for the Internet, and is currently one of the protocols under consideration in the IETF Mobile Ad Hoc Networks (MANET) Working Group for adoption as an Internet standard for IP routing in ad hoc networks [MANET].

This chapter describes the design of the DSR protocol and provides a summary of some of our current simulation and testbed implementation results for it: Section 5.1 discusses our assumptions in the design. In Section 5.2, we present the design of the protocol and describe its resulting important properties. In particular, we describe the design of the two mechanisms that make up the operation of DSR: *Route Discovery* and *Route Maintenance*. We also discuss the use of DSR in supporting heterogeneous networks and interconnecting to the Internet, and we describe the current support present in DSR for routing of multicast packets in ad hoc networks. Section 5.3 summarizes some of our simulation results for DSR and describes a physical outdoor ad hoc network testbed we have built in Pittsburgh for DSR experiments. Finally, we discuss related work in Section 5.4 and present conclusions in Section 5.5.

5.1 ASSUMPTIONS

We assume that all nodes wishing to communicate with other nodes within the ad hoc network are willing to participate fully in the network protocols. In particular, each participating node should be willing to forward packets for other nodes.

We refer to the minimum number of hops necessary for a packet to travel from any node located at one extreme edge of the ad hoc network to any node located at the opposite extreme as the *diameter* of the network. We assume that the diameter will often be small (perhaps 5 or 10 hops), but it may often be greater than 1.

Packets may be lost or corrupted in transmission on the wireless network. A node receiving a corrupted packet can detect the error and discard the packet.

Nodes within the ad hoc network may move at any time without notice and may even move continuously, but we assume that the speed with which nodes move is moderate with respect to the packet transmission latency and wireless transmission range of the underlying network hardware. In particular, DSR can support very rapid rates of arbitrary node mobility, but we assume that nodes do not continuously move so rapidly as to make the flooding of every individual data packet the only possible routing protocol.

We assume that nodes may enable *promiscuous* receive mode on their wireless network interface hardware, causing the hardware to deliver every received packet to the network driver software without filtering based on link layer destination address. Although we do not require this facility, it is, for example, common in current LAN hardware for broadcast media, including wireless, and some of our optimizations can take advantage of its availability. Use of promiscuous mode does increase the software overhead on the CPU, but we believe that wireless network speeds are more the inherent limiting factor to performance in current and future systems; we also believe that portions of the protocol are suitable for implementation directly within a programmable network interface unit to avoid this overhead on the CPU [Johnson+ 1996a]. Use of promiscuous mode may also increase the power consumption of the network interface hardware, depending on the design of the receiver hardware. In such cases, DSR can easily be used without the optimizations that depend on promiscuous receive mode or can be programmed to switch the interface into promiscuous mode only periodically.

At times, wireless communication between any pair of nodes may not work equally well in both directions, perhaps because of differing antenna or propagation patterns or sources of interference around the two nodes [Bantz+ 1994, Lauer 1995]. That is, wireless communications between each pair of nodes will in many cases be able to operate *bidirectionally,* but at times the wireless link between two nodes may be only *unidirectional,* allowing one node to successfully send packets to the other while no communication is possible in the reverse direction. Although many routing protocols operate correctly only over bidirectional links, DSR can successfully discover and forward packets over paths that contain unidirectional links. Some medium access control (MAC) protocols, however, such as MACA [Karn 1990], MACAW [Bharghavan+ 1994], or IEEE 802.11 [IEEE 1997], limit unicast data packet transmission to bidirectional links because of the required bidirectional exchange of RTS and CTS packets in these protocols and because of the link level acknowledgment feature in IEEE 802.11. When used on top of MAC protocols such as these, DSR can take advantage of additional optimizations, such as the route reversal optimization described below.

Each node selects a *single* IP address by which it will be known in the ad hoc network. Although a single node may have many different physical network interfaces, which in a typical IP network each have a different IP address, we require each node to select one of these and to use only that address when participating in the DSR protocol. This allows each node to be recognized by all other nodes in the ad hoc network as a single entity regardless of which network interface the other nodes use to communicate with it. In keeping with the terminology used by Mobile IP [Johnson 1995], we refer to the address by which each mobile node is known in the ad hoc network as its *home address,* as this is typically the one that the node uses while connected to its home network (rather than while away, being a member of the ad hoc network). Each node's home address may be assigned by any mechanism (e.g., static assignment or use of DHCP for dynamic assignment [Droms 1997]), although the method of such assignment is outside the scope of the DSR protocol.

5.2 DSR PROTOCOL DESCRIPTION—OVERVIEW AND IMPORTANT PROPERTIES

The DSR protocol is composed of two mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

- *Route Discovery,* by which a node **S** wishing to send a packet to a destination node **D** obtains a source route to **D**. Route Discovery is used only when **S** attempts to send a packet to **D** and does not already know a route to it.
- *Route Maintenance,* by which node **S**, while using a source route to **D**, is able to detect, if the network topology has changed such that it can no longer use its route to **D** because a link along the route no longer works. When Route Maintenance indicates that a source route is broken, **S** can attempt to use any other route to **D** it happens to know, or it can invoke Route Discovery again to find a new route. Route Maintenance is used only when **S** is actually sending packets to **D**.

Route Discovery and Route Maintenance each operate entirely *on demand.* In particular, DSR, unlike other protocols, requires *no* periodic packets of *any kind* at *any level* within the network. For example, it does not use any periodic routing advertisement, link status sensing, or neighbor detection packets; nor does it rely on these functions from any underlying protocols in the network. This entirely on-demand behavior and lack of periodic activity allow the number of overhead packets caused by DSR to scale down to *zero* when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR *automatically* scales to only that needed to track the routes currently in use.

In response to a single Route Discovery (as well as through routing information from other packets' overheard), a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid because a node with multiple routes to a destination can try another cached route if the one it has been using fails. This caching of multiple routes also avoids the overhead incurred by performing a new Route Discovery each time a route in use breaks.

Route Discovery and Route Maintenance are designed to allow unidirectional links and asymmetric routes to be easily supported. In particular, as noted in Section 5.1, in wireless networks it is possible that a link between two nodes may not work equally well in both directions because of differing antenna or propagation patterns or sources of interference. DSR allows such unidirectional links to be used when necessary, improving overall performance and network connectivity in the system.

DSR also supports internetworking between different types of wireless networks, allowing a source route to be composed of hops over a combination of any network types available [Broch+ 1999b]. For example, some nodes in the ad hoc network may have only short-range radios while other nodes may have both short-range and long-range radios; the combination of these nodes can be considered by DSR as a single ad hoc network. In addition, the routing of DSR has been integrated into standard Internet routing, where a "gateway" node connected to the Internet also participates in the ad hoc network routing protocols; it has also been integrated into Mobile IP routing, where such a gateway node also serves the role of a Mobile IP foreign agent [Johnson 1995].

5.2.1 DSR Route Discovery

When some node **S** originates a new packet destined for some node **D**, it places in the header of the packet a *source route* giving the sequence of hops that the packet should follow. Normally, **S** obtains a suitable source route by searching its *Route Cache* of routes previously learned, but if no route is found in its cache it initiates the Route Discovery protocol to find a new route to **D** dynamically. In this case, we call **S** the *initiator* and **D** the *target* of the Route Discovery.

Figure 5.1 illustrates an example Route Discovery, in which node **A** is attempting to discover a route to node **E**. To initiate the Route Discovery, **A** transmits a ROUTE REQUEST message as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless

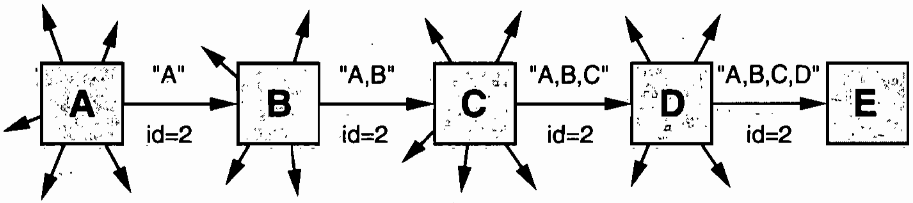


Figure 5.1. Route Discovery Example with Node **A** as the Initiator and Node **E** as the Target

transmission range of **A**. Each ROUTE REQUEST message identifies the initiator and target of the Route Discovery and also contains a unique *request ID,* determined by the initiator of the REQUEST. Each ROUTE REQUEST also contains a record listing the address of each intermediate node through which this particular copy of the ROUTE REQUEST message has been forwarded. This route record is initialized to an empty list by the initiator of the Route Discovery.

When another node receives a ROUTE REQUEST, if it is the target of the Route Discovery it returns a ROUTE REPLY message to the Route Discovery initiator, giving a copy of the accumulated route record from the ROUTE REQUEST; when the initiator receives this ROUTE REPLY, it caches this route in its Route Cache for use in sending subsequent packets to this destination. Otherwise, if the node receiving the ROUTE REQUEST recently saw another ROUTE REQUEST message from this initiator bearing this same request ID, or if it finds that its own address is already listed in the route record in the ROUTE REQUEST message, it discards the REQUEST. If not, this node appends its own address to the route record in the ROUTE REQUEST message and propagates it by transmitting it as a local broadcast packet (with the same request ID).

In returning the ROUTE REPLY to the Route Discovery initiator, such as node **E** replying to **A** in Figure 5.1, node **E** typically examines its own Route Cache for a route back to **A** and, if found, uses it for the source route for delivery of the packet containing the ROUTE REPLY. Otherwise, **E** may perform its own Route Discovery for target node **A**, but to avoid possible infinite recursion of Route Discoveries it must piggyback this ROUTE REPLY on its own ROUTE REQUEST message for **A**. It is also possible to piggyback other small data packets, such as a TCP SYN packet [Postel 1981b], on a ROUTE REQUEST using this same mechanism. Node **E** can also simply reverse the sequence of hops in the route record that it is trying to send in the ROUTE REPLY and use this as the source route on the packet carrying the ROUTE REPLY itself. For MAC protocols such as IEEE 802.11 that require a bidirectional frame exchange as part of the MAC protocol [IEEE 1997], this route reversal is preferred as it avoids the overhead of a possible second Route Discovery and it tests the discovered route to ensure that it is bidirectional before the Route Discovery initiator begins using it. However, this technique will prevent the discovery of routes using unidirectional links. In wireless environments where the use of unidirectional links is permitted, such routes may in some cases be more efficient than those with only bidirectional links, or they may be the only way to achieve connectivity to the target node.

When initiating a Route Discovery, the sending node saves a copy of the original packet in a local buffer called the *Send Buffer*. The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the Send Buffer is stamped with the time that it was placed there and is discarded after residing in the Send Buffer for some time-out period; if necessary to prevent the Send Buffer from overflowing, a FIFO or other replacement strategy can be used to evict packets before they expire.

While a packet remains in the Send Buffer, the node should occasionally initiate a new Route Discovery for the packet's destination address. However, the node must limit the rate at which such new Route Discoveries for the same address are initiated because it is possible that the destination node is not currently reachable. In particular, because of the limited wireless transmission range and the movement of the nodes in the network, the network may at times become partitioned, meaning that there is currently no sequence of nodes through which a packet can be forwarded to reach the destination. Depending on the movement pattern and the density of nodes in the network, such network partitions may be either rare or common.

If a new Route Discovery was initiated for each packet sent by a node in such a situation, a large number of unproductive ROUTE REQUEST packets will be propagated throughout the subset of the ad hoc network reachable from this node. To reduce such overhead, we use exponential backoff to limit the rate at which new Route Discoveries may be initiated by any node for the same target. If the node attempts to send additional data packets to this same node more frequently than this limit allows, the subsequent packets should be buffered in the Send Buffer until a ROUTE REPLY is received, but the node must not initiate a new Route Discovery until the minimum allowable interval between new Route Discoveries for this target has been reached. This limitation on the maximum rate of Route Discoveries for the same target is similar to the mechanism required by Internet nodes to limit the rate at which ARP requests are sent for any single target IP address [Braden 1989].

5.2.2 DSR Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route; the packet is retransmitted (up to a maximum number of attempts) until this confirmation of receipt is received. For example, in the situation illustrated in Figure 5.2, node **A** has originated a packet for **E** using a source route through intermediate nodes **B**, **C**, and **D**. In this case, node **A** is responsible for receipt of the packet at **B**, node **B** is responsible for receipt at **C**, node **C** is responsible for receipt at **D**, and finally node **D** is responsible for receipt at **E**. This confirmation of receipt may in many cases be provided at no cost to DSR, either as an existing standard part of the MAC

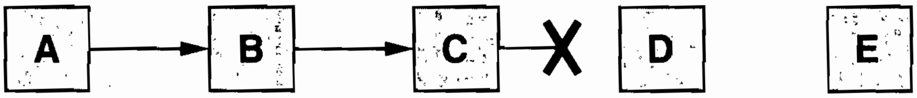


Figure 5.2. Route Maintenance Example Node **C** is unable to forward a packet from **A** to **E** over its link to the next hop, **D**.

protocol in use (such as the link level acknowledgment frame defined by IEEE 802.11 [IEEE 1997]) or by a *passive acknowledgment* [Jubin+ 1987] (in which, for example, **B** confirms receipt at **C** by overhearing **C** transmit the packet to forward it on to **D**). If neither of these confirmation mechanisms is available, the node transmitting the packet may set a bit in the packet's header to request that a DSR-specific software acknowledgment be returned by the next hop; this software acknowledgment will normally be transmitted directly to the sending node, but, if the link between these two nodes is unidirectional, it may travel over a different, multihop path.

If the packet is retransmitted by some hop the maximum number of times and no receipt confirmation is received, this node returns a ROUTE ERROR message to the original sender of the packet, identifying the link over which the packet could not be forwarded. For example, in Figure 5.2, if **C** is unable to deliver the packet to the next hop **D**, **C** returns a ROUTE ERROR to **A**, stating that the link from **C** to **D** is currently "broken." Node **A** then removes this broken link from its cache, and any retransmission of the original packet is a function for upper-layer protocols such as TCP. For sending such a retransmission or other packets to this same destination **E**, if **A** has in its Route Cache another route to **E** (for example, from additional ROUTE REPLYs from its earlier Route Discovery or from having overheard sufficient routing information from other packets), it can send the packet using the new route immediately. Otherwise, it may perform a new Route Discovery for this target (subject to the exponential backoff described in Section 5.2.1).

5.2.3 Additional Route Discovery Features

Caching Overheard Routing Information

A node forwarding or otherwise overhearing any packet may add the routing information from that packet to its own Route Cache. In particular, the source route used in a data packet, the accumulated route record in a ROUTE REQUEST, or the route being returned in a ROUTE REPLY may all be cached by any node. Routing information from any of these packets received may be cached whether the packet was addressed to this node, sent to a broadcast (or multicast) MAC address, or received while the node's network interface was in promiscuous mode.

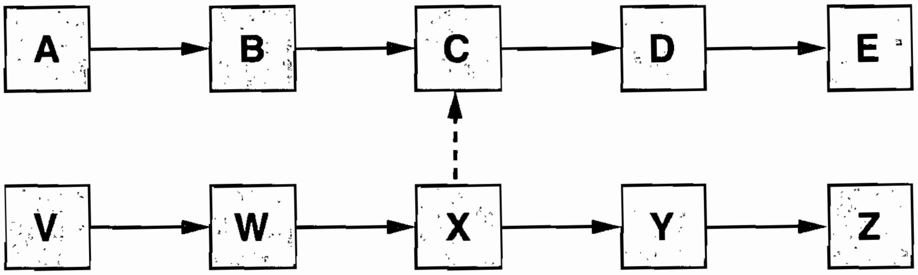


Figure 5.3. Limitations on Caching Overheard Routing Information
Node **C** is forwarding to **E** and overhears packets from **X**.

However, one limitation on caching of such overheard routing information is the possible presence of unidirectional links in the ad hoc network (Section 5.1). For example, Figure 5.3 illustrates a situation in which node **A** is using a source route to communicate with node **E**. As node **C** forwards a data packet along the route from **A** to **E**, it can always add to its cache the presence of the "forward" direction links, which it learns from the headers of these packets, from itself to **D** and from **D** to **E**. However, the "reverse" direction of the links identified in the packet headers, from **C** back to **B** and from **B** to **A**, may not work because these links might be unidirectional. If **C** knows that the links are in fact bidirectional—for example, because of the MAC protocol in use—it can cache them but otherwise should not.

Likewise, node **V** in Figure 5.3 is using a different source route to communicate with node **Z**. If node **C** overhears node **X** transmitting a data packet to forward it to **Y** (from **V**), **C** should consider whether or not the links involved can be known to be bidirectional before caching them. If the link from **X** to **C** (over which this data packet was received) can be known to be bidirectional, **C** can cache the link from itself to **X**, the link from **X** to **Y**, and the link from **Y** to **Z**. If all links can be assumed to be bidirectional, **C** can also cache the links from **X** to **W** and from **W** to **V**. Similar considerations apply to the routing information that might be learned from forwarded or otherwise overheard ROUTE REQUEST or ROUTE REPLY packets.

Replying to Route Requests Using Cached Routes

A node receiving a ROUTE REQUEST for which it is not the target searches its own Route Cache for a route to the REQUEST target. If a route is found, the node generally returns a ROUTE REPLY to the initiator itself rather than forwarding the ROUTE REQUEST. In the ROUTE REPLY, it sets the route record to list the sequence of hops over which this copy of the ROUTE REQUEST was forwarded to it, concatenated with its own idea of the route from itself to the target from its Route Cache.

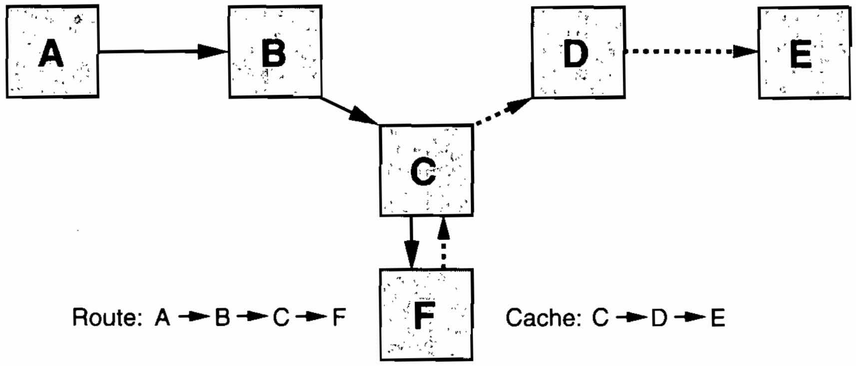


Figure 5.4. A Possible Duplication of Route Hops Avoided by the Route Discovery Limitation on Replying to ROUTE REQUESTs from the Route Cache

However, before transmitting a ROUTE REPLY packet that was generated using information from its Route Cache in this way, a node must verify that the resulting route being returned in the ROUTE REPLY, after this concatenation, contains no duplicate nodes listed in the route record. For example, Figure 5.4 illustrates a case in which a ROUTE REQUEST for target **E** has been received by node **F** and node **F** already has in its Route Cache a route from itself to **E**. The concatenation of the accumulated route from the ROUTE REQUEST and the cached route from **F**'s Route Cache includes a duplicate node in passing from **C** to **F** and back to **C**.

Node **F** in this case *could* attempt to edit the route to eliminate the duplication, resulting in a route from **A** to **B** to **C** to **D** and on to **E**, but in this case node **F** is not on the route that it returned in its own ROUTE REPLY. DSR Route Discovery prohibits node **F** from returning such a ROUTE REPLY from its cache for two reasons. First, this limitation increases the probability that the resulting route is valid because **F** in this case should have received a ROUTE ERROR if the route had stopped working. Second, this limitation means that a ROUTE ERROR traversing the route is very likely to pass through any node that sent the ROUTE REPLY for the route (including **F**), which helps to ensure that stale data is removed from caches (such as at **F**) in a timely manner. Otherwise, the next Route Discovery initiated by **A** might also be contaminated by a ROUTE REPLY from **F** containing the same stale route. If the ROUTE REQUEST does not meet these restrictions, the node (**F** in this example) discards the ROUTE REQUEST rather than replying to or propagating it.

Preventing Route Reply Storms

The ability of nodes to reply to a ROUTE REQUEST on the basis of information in their Route Caches, as described in the previous section, can

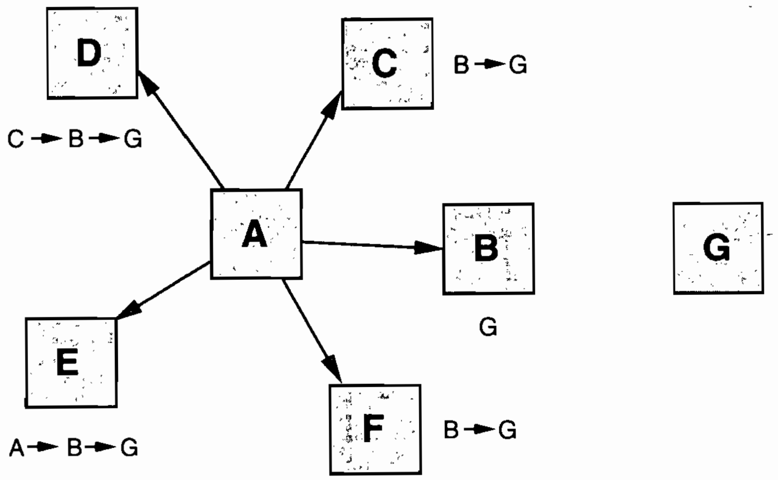


Figure 5.5. A ROUTE REPLY Storm

result in a possible ROUTE REPLY "storm" in some cases. In particular, if a node broadcasts a ROUTE REQUEST for a target node for which the node's neighbors have a route in their Route Caches, each neighbor may attempt to send a ROUTE REPLY, thereby wasting bandwidth and possibly increasing the number of network collisions in the area.

For example, in the situation shown in Figure 5.5, nodes **B**, **C**, **D**, **E**, and **F** all receive **A**'s ROUTE REQUEST for target **G**, and each has the indicated route cached for this target. Normally, they all attempt to reply from their own Route Caches and all send their REPLYs at about the same time because they all received the broadcast ROUTE REQUEST at about the same time. Such simultaneous replies from different nodes all receiving the ROUTE REQUEST may create packet collisions among some or all of these REPLYS and may cause local congestion in the wireless network. In addition, it is often the case that the different replies indicate routes of different lengths, as shown in this example.

If a node can put its network interface into promiscuous receive mode, it should delay sending its own ROUTE REPLY for a short period and listen to see if the initiating node begins using a shorter route first. That is, this node should delay sending its own ROUTE REPLY for a random period $d = H \times (h - 1 + r)$, where h is the length in number of network hops for the route to be returned in this node's ROUTE REPLY, r is a random number between 0 and 1, and H is a small constant delay (at least twice the maximum wireless link propagation delay) to be introduced per hop. This delay effectively randomizes the time at which each node sends its ROUTE REPLY; all nodes sending ROUTE REPLYs giving routes of length less than h send their REPLYs before this node, and all nodes sending ROUTE REPLYs giving routes of length greater than h send their REPLYs after this node. Within the delay period, this node promiscuously receives all packets, looking for data packets from the initiator of this Route Discovery destined for the target of the Discovery. If such a data packet received by this node during the delay period uses a source route of length less than or equal to h, the node may infer that the initiator of the Route Discovery has already received a ROUTE REPLY giving an equally good or better route. In this case, this node cancels its delay timer and does *not* send its ROUTE REPLY for this Route Discovery.

Route Request Hop Limits

Each ROUTE REQUEST message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the ROUTE REQUEST. As the REQUEST is forwarded, this limit is decremented, and the REQUEST packet is discarded if the limit reaches zero before finding the target. We use this mechanism to send a *nonpropagating* ROUTE REQUEST (i.e., with hop limit 0) as an inexpensive method of determining if the target is currently a neighbor of the initiator or if a neighbor node has a route to the target cached (effectively using the neighbor's cache as an extension of the initiator's own cache). If no ROUTE REPLY is received after a short time-out, a *propagating* ROUTE REQUEST (i.e., with no hop limit) is sent.

We have also considered using this mechanism to implement an *expanding ring* search for the target [Johnson+ 1996a]. For example, a node can send an initial nonpropagating ROUTE REQUEST as described above; if no ROUTE REPLY is received for it, the node can initiate another ROUTE REQUEST with a hop limit of 1. For each ROUTE REQUEST initiated, if no ROUTE REPLY is received for it, the node can double the hop limit used on the previous attempt to progressively explore for the target node without allowing the ROUTE REQUEST to propagate over the entire network. However, this expanding ring search approach can increase the average latency of Route Discovery, as multiple Discovery attempts and time-outs may be needed before a route to the target node is found.

5.2.4 Additional Route Maintenance Features

Packet Salvaging

After sending a ROUTE ERROR message as part of Route Maintenance, as described in Section 5.2.2, a node may attempt to *salvage* the data packet that caused the ROUTE ERROR rather than discard it. To salvage a packet, the node sending a ROUTE ERROR searches its own Route Cache for a route from itself to the destination of the packet causing the ERROR.

If such a route is found, the node may salvage the packet after returning the ROUTE ERROR, by replacing the original source route on the packet with the route from its Route Cache. It then forwards the packet to the next node indicated along this source route. For example, in Figure 5.2, if node **C** has another route cached to node **E**, it can salvage the packet by applying this route to it rather than discard it.

When salvaged in this way, the packet is also marked as having been salvaged to prevent a single packet being salvaged multiple times. Otherwise, it is possible for the packet to enter a routing loop as different nodes repeatedly salvage it and replace the source route on it with routes to each other. An alternative salvaging mechanism that we have considered is to replace only the unused suffix of the original route (the portion in advance of this node) with the new route from this node's Route Cache, forming a new route whose prefix is the original route and whose suffix is the route from the Cache. In this case, the normal rules for avoiding the listing of duplicate nodes in a source route are sufficient to avoid routing loops. However, this mechanism prevents the new route from "backtracking" from this node to an earlier node already traversed by this packet and then being forwarded along a different remaining sequence of hops to the destination. Our current salvaging mechanism allows backtracking but prevents a packet from being salvaged more than once.

Automatic Route Shortening

Source routes may be automatically shortened if one or more of their intermediate hops become unnecessary. This mechanism of automatically shortening routes is somewhat similar to the use of passive acknowledgments. In particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in promiscuous receive mode), it examines the route's unused portion. If this node is not the intended next hop for the packet but is named in the later unused portion of the packet's source route, it can infer that the intermediate nodes before itself in the source route are no longer needed. Figure 5.6 illustrates an example in which node **C** has overheard a data packet being transmitted from **A** to **B** for later forwarding to **C**; the arrow pointing to one node in the source route

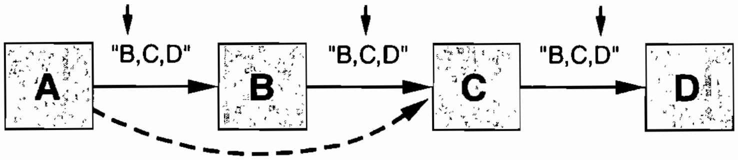


Figure 5.6. An Example of Automatic Route Shortening

in each packet indicates the intended next receiver of the packet along the route.

In this case, this node (**C**) returns a *gratuitous* ROUTE REPLY message to the original sender of the packet (**A**). The ROUTE REPLY gives the shorter route as the concatenation of the portion of the original source route up through the node that transmitted the overheard packet plus the suffix of the original source route beginning with the node returning the gratuitous ROUTE REPLY. In this example, the route returned in the gratuitous ROUTE REPLY message sent from **C** to **A** gives the new route as the sequence of hops from **A** to **C** to **D**.

Increased Spreading of Route Error Messages

When a source node receives a ROUTE ERROR for a data packet that it originated, it propagates it to its neighbors by piggybacking it on its next ROUTE REQUEST. In this way, stale information in the caches of nodes around this source node will not generate ROUTE REPLYs that contain the same invalid link for which this source node received the ROUTE ERROR.

For example, in the situation shown in Figure 5.2, node **A** learns from the ROUTE ERROR message from **C** that the link from **C** to **D** is currently broken. It thus removes this link from its own Route Cache and initiates a new Route Discovery (if it doesn't have another route to **E** in its Route Cache). On the ROUTE REQUEST packet initiating this Route Discovery, node **A** piggybacks a copy of this ROUTE ERROR message, ensuring that it spreads well to other nodes and guaranteeing that any ROUTE REPLY that it receives (including those from other nodes' Route Caches) in response to this ROUTE REQUEST does not contain a route that assumes the existence of this broken link.

We have also considered, but not simulated, a further improvement to Route Maintenance in which a node that receives a ROUTE ERROR, such as **A** in Figure 5.4, forwards it along the same source route that resulted in it. This will almost guarantee that the ROUTE ERROR reaches the node that generated the ROUTE REPLY containing the broken link, which prevents that node from contaminating a future Route Discovery with the same broken link.

Caching Negative Information

In some cases, DSR can potentially benefit from nodes caching "negative" information in their Route Caches. For example, in Figure 5.2, if node **A** caches the fact that the link from **C** to **D** is currently broken (rather than simply removing this hop from its Route Cache), it can guarantee that no ROUTE REPLY that utilizes this broken link, which it receives in response to its new Route Discovery, will be accepted. A short expiration period must be placed on this negative cached information because, while this entry is in its Route Cache, **A** will otherwise refuse to allow this link in its cache even if this link begins working again.

Another case in which caching negative information in a node's Route Cache might be useful is one in which a link is providing highly variable service, sometimes working correctly but often not. This situation can occur, for example, when the link is near the limit of the sending node's wireless transmission range and there are significant sources of interference (e.g., multipath) near the receiving node on this link. In this case, by caching the negative information that this link is broken, a node can avoid adding this problematic link back to its Route Cache during the brief periods in which it is working correctly.

We have not included this caching of negative information in our simulations or implementation of DSR, although we have found situations in our testbed implementation (Section 5.3.2) where it could improve the performance of Route Discovery [Maltz+ 1999b]. A challenge in implementing the caching of negative information that we are researching is the difficulty of picking a suitable expiration period for such cache entries.

5.2.5 Support for Heterogeneous Networks and Mobile IP

In configuring and deploying an ad hoc network, in many cases all nodes will be equipped with the same type of wireless network interfaces, allowing simple routing between nodes over arbitrary sequences of network hops. However, a more flexible configuration might be to also equip a subset of the nodes with a second network interface consisting of a longer-range (and thus generally lower-speed) wireless network interface. For example, in a military setting a group of soldiers might use short-range radios to communicate among themselves while relaying through truck-mounted higher-power radios to communicate with other groups.

This general type of network configuration is the ad hoc equivalent of wireless *overlay networks* [Katz+ 1996]. Because of the high degree of locality likely to be present among directly cooperating nodes communicating with each other, such a network configuration allows high-speed communication among such cooperating nodes and at the same time allows communication with other nodes further away without requiring very large numbers of network hops. The longer-range radios might also allow gaps between different groups of nodes to be spanned, reducing the probability of network partition. A simple example of such an ad hoc network configuration is shown in Figure 5.7, where nodes **A**, **B**, and **C** each have both short-range and long-range radio interfaces and all other nodes in the ad hoc network have only short-range interfaces. Node **X** is using a source route to node **Y** that employs a sequence of both short-range and long-range hops.

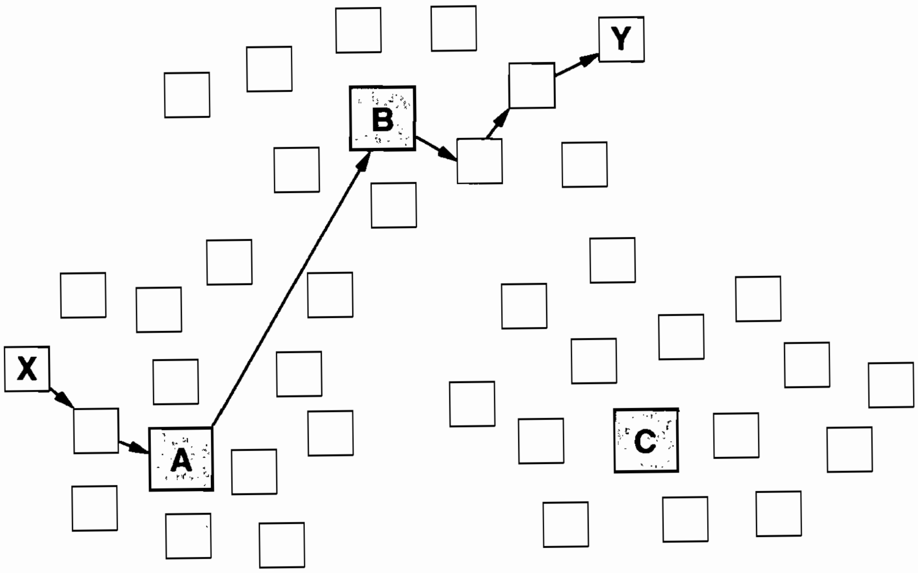


Figure 5.7. A Heterogeneous Ad Hoc Network

Use of Interface Indices in DSR

DSR supports automatic, seamless routing in these (see Figure 5.7) and other heterogeneous configurations through its logical addressing model [Broch+ 1999b]. Using conventional IP addressing, each ad hoc network node configures a different IP address for its possibly many individual network interfaces, but, as noted in Section 5.1, each node using DSR chooses *one* of these as its *home address* to use for all communication while in the ad hoc network. This use of a single IP address per node gives DSR the ability to treat the overall network as single routing domain. To then distinguish between the different network interfaces on a node, each node independently assigns a locally unique *interface index* to each of its own network interfaces.

The interface index for any network interface on a node is an *opaque* value assigned by the node itself. The particular value chosen must be unique among the network interfaces on that individual node, but it need have no other significance and need not be coordinated with any other nodes in the choice of their own interface indices. On many operating systems, a unique value to identify each network interface is already available and can be used for this purpose; for example, the `if_index` field in the `ifnet` structure for a network interface in BSD UNIX-based networking stacks [Wright+ 1995] can be used directly by a node for the interface index for that network interface.

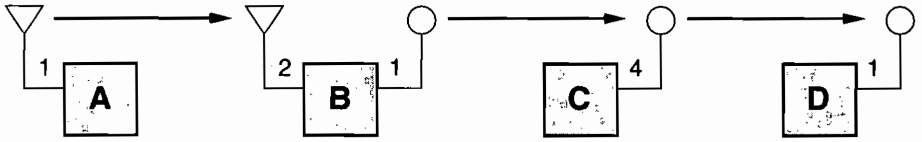


Figure 5.8. An Ad Hoc Network of Nodes with Heterogeneous Network Interfaces

As an example, Figure 5.8 illustrates a simple ad hoc network of four nodes, in which node **A** is using one type of network interface (represented by the triangles), node **C** and node **D** are using a different type (represented by the circles), and node **B** is configured with both types and can forward packets between the two. The number labeling each network interface indicates the interface index chosen by the interface's corresponding node. Because the interface indices are chosen independently by each node, it is possible, for example, that nodes **B** and **D** each choose index 1 for their circle network interfaces but that node **C** chooses index 4.

The interface index is used as part of each hop in each source route discovered and used by DSR. Specifically, a path through the ad hoc network from a source node $\mathbf{N}_0$ to a destination node $\mathbf{N}_m$ is fully represented as a series of hops $\mathbf{N}_0/i_0 \rightarrow \mathbf{N}_1/i_1 \rightarrow \mathbf{N}_2/i_2 \rightarrow \ldots \rightarrow \mathbf{N}_m$, where $\mathbf{N}_k/i_k$ indicates that node $\mathbf{N}_k$ must transmit the packet using its network interface i_k in order to deliver the packet over the next hop to node $\mathbf{N}_{k+1}$.

In forwarding a ROUTE REQUEST, a node adds to the route record in it not only its own address (Section 5.2.1) but also the interface index of its own network interface on which it forwards the packet. To allow the reversing of a sequence of hops for a reverse route back to the originating node (when, for example, the existence of bidirectional links can be assumed on the basis of the underlying MAC protocol), the node forwarding the ROUTE REQUEST may also add to the route record in it the interface index of its own network interface on which it *received* the ROUTE REQUEST packet. For example, the source route shown in Figure 5.8 is $\mathbf{A}/1 \rightarrow \mathbf{B}/1 \rightarrow \mathbf{C}/4 \rightarrow \mathbf{D}$. The corresponding reversed route is $\mathbf{D}/1 \rightarrow \mathbf{C}/4 \rightarrow \mathbf{B}/2 \rightarrow \mathbf{A}$. The interface indices to represent a route are carried in the ROUTE REQUEST, the ROUTE REPLY, and the source route in the headers of the data packets.

Internet Interconnection and Mobile IP

DSR supports the seamless interoperation between an ad hoc network and the Internet, allowing packets to be routed transparently from the ad hoc network to nodes in the Internet and from the Internet to nodes in the ad hoc network [Broch+ 1999b]. To enable this interoperation, one node (or more) in the ad hoc network must be connected to the Internet, such that it participates in the ad hoc network through DSR and also participates in the Internet through standard IP routing. We call such a node a *gateway* between the ad hoc network and the Internet. In this way, DSR allows the coverage range around a wireless Internet base station, for example, to be dynamically enlarged through multiple "hops" between nodes through the ad hoc network. It is also possible for such a gateway node to operate as a Mobile IP home agent or foreign agent [Johnson 1995], allowing nodes to visit the ad hoc network as a Mobile IP foreign network and allowing nodes whose home network is the ad hoc network to visit other networks using Mobile IP.

This Internet interconnection is implemented through two special reserved interface index values, used by gateway nodes to identify their interconnection to the Internet. If the node has a separate physical network interface other than the network interface(s) that it uses for participation in the ad hoc network, by which it connects to the Internet, the reserved interface index is used to identify that interface. However, it is also possible for a node to use a single network interface both for ad hoc network participation and for Internet connection through standard IP routing. In this case, the reserved interface index identifies the logically separate functionality of this interface for its Internet connection, and the node uses another (locally assigned) interface index value to identify this interface in its separate logical participation in the ad hoc network.

If the gateway node is acting as a Mobile IP home agent or foreign agent (termed a *mobility agent*) on this network interface, it uses the reserved interface index value `IF_INDEX_MA`. Otherwise, the gateway node uses the reserved value `IF_INDEX_ROUTER`. The distinction between the reserved index values for mobility agents and for routers allows mobility agents to advertise their existence (as needed for Mobile IP) at no cost. A node in the ad hoc network that processes a routing header listing the interface index `IF_INDEX_MA` can then send a unicast Mobile IP AGENT SOLICITATION [Perkins 1996] to the corresponding address in the routing header to obtain complete information about the Mobile IP services provided.

In processing a received ROUTE REQUEST, a gateway node generates a ROUTE REPLY, giving its reserved interface index value, if it believes that it may be able to reach the target node through its Internet connection. Thus, the originator of the Route Discovery may receive REPLYs both from the gateway and from the node itself if the node is present in the ad hoc network. When later sending packets to this destination, the sender should prefer cached routes that do not traverse a hop with an interface index of `IF_INDEX_MA` or `IF_INDEX_ROUTER`, as these will favor routes that lead directly to the destination node within the ad hoc network.

5.2.6 Multicast Routing with DSR

DSR does not currently support true multicast routing, but it does support an approximation that is sufficient in many network contexts. Through an extension of the Route Discovery mechanism, DSR supports the controlled flooding of a data packet to all nodes in the ad hoc network that are within some specified number of hops of the originator. These nodes may then apply destination address filtering (e.g., in software) to limit the packet to those nodes subscribed to the packet's indicated multicast destination address. Even though this mechanism does not support pruning of the broadcast tree to conserve network resources, it can be used to distribute information to all nodes in the ad hoc network subscribed to the destination multicast address. This mechanism may also be useful for sending application-level packets to all nodes in a limited range around the sender.

In this form of multicasting, an application on a DSR node sends a packet to a multicast destination address and DSR piggybacks the data from the packet inside a ROUTE REQUEST targeted at the multicast address. The normal ROUTE REQUEST propagation scheme described in Section 5.2.1 results in this packet being efficiently distributed to all nodes in the network within the specified hop count (TTL) of the originator. After forwarding the packet as defined for Route Discovery, each receiving node then individually examines its destination address and discards the packet if it is destined for a multicast address to which this node is not subscribed.

5.2.7 Location of DSR Functions in the ISO Network Reference Model

In our design of DSR, we had to determine the layer within the protocol hierarchy at which to implement ad hoc network routing. We considered two options: the *link layer* (ISO layer 2) and the *network layer* (ISO layer 3). Originally, we opted for the link layer for several reasons:

- Pragmatically, running the DSR protocol at the link layer maximizes the number of mobile nodes that can participate in ad hoc networks. For example, the protocol can route equally well between IPv4 [Postel 1981a], IPv6 [Deering+ 1998], and IPX [Turner 1990] nodes.
- Historically [Johnson 1994, Johnson+ 1996a], as described more fully in Section 5.4, DSR grew from our contemplation of a multihop propagating version of the Internet Address Resolution Protocol (ARP) [Plummer 1982] as well as from the routing mechanism used in IEEE 802 source routing bridges [Perlman 1992]. These are layer-2 protocols.
- Technically, we designed DSR to be simple enough to be implemented directly in the firmware inside wireless network interface cards [Johnson 1994, Johnson+ 1996a], well below the layer-3 software within a mo-

bile node. We see great potential in this for DSR running inside a cloud of mobile nodes around a fixed base station, where it would transparently extend the coverage range to these nodes. Mobile nodes that would otherwise be unable to communicate with the base station because of factors such as distance, fading, or local interference could then reach the base station through their peers.

Ultimately, however, we decided to specify [Broch+ 1999a] and to implement [Maltz+ 1999b] DSR as a layer-3 protocol because this is the only layer at which we could realistically support nodes with multiple network interfaces of different types, as described in Section 5.2.5.

5.3 DSR EVALUATION

This section summarizes some of our experiences in evaluating DSR through detailed studies using discrete event simulation and through implementation and actual operation and experience with the protocol in an ad hoc networking testbed environment. Complete details of this evaluation can be found in other publications [Broch+ 1998, Maltz+ 1999a, Maltz+ 1999b].

5.3.1 Simulation Summary

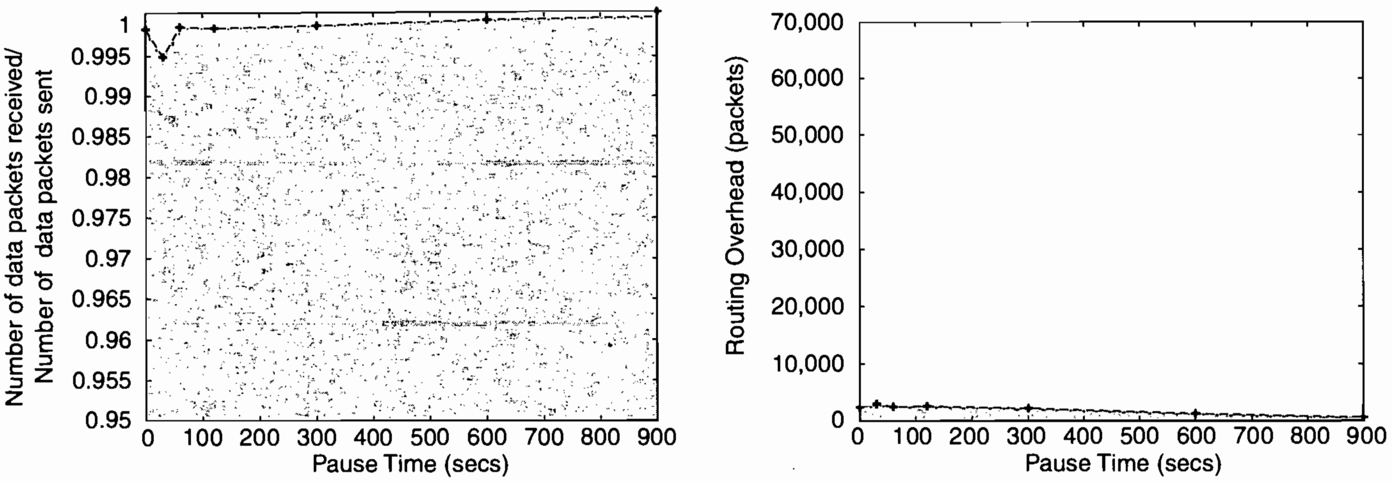
Our simulation environment consists of a set of wireless and mobile networking extensions that we created [Broch+ 1998]. Based on the publicly available *ns-2* network simulator from the University of California at Berkeley and the VINT Project [Fall+ 1997], these extensions provide a detailed model of the physical and link layer behavior of a wireless network and allow arbitrary movement of nodes within it. At the physical layer, we provide realistic modeling of factors such as free space and ground reflection propagation, transmission power, antenna gain, receiver sensitivity, propagation delay, carrier sense, and capture effect [Rappaport 1996]. At the link layer, we model the complete Distributed Coordination Function (DCF) MAC protocol of the IEEE 802.11 wireless LAN protocol standard [IEEE 1997], along with the standard Internet ARP [Plummer 1982]. These wireless and mobile networking extensions are available from the Carnegie-Mellon University Monarch Project web pages [Monarch] and have been widely used by other researchers; a version has also been adopted as a part of the standard VINT release of *ns-2*.

We conducted a number of simulation studies with this environment, analyzing the behavior and performance of DSR and comparing it to other proposed routing protocols for ad hoc networks [Broch+ 1998, Maltz+ 1999a]. Here we summarize only some of the basic results that indicate DSR's excellent performance. All simulations were run in ad hoc networks of

50 mobile nodes moving according to the *random waypoint* mobility model [Johnson+ 1996a] within a flat rectangular (1500 m × 300 m) area; all simulations were run for 15 minutes (900 seconds) of simulated time. Data traffic was generated using constant bit rate (CBR) UDP traffic sources, with either 10, 20, or 30 mobile nodes acting as traffic sources generating 4 packets/second each. We show here the results for 20 sources, although the results for 10 and 30 sources are similar. All movement and application layer communication were generated in advance and captured in a *scenario file*, allowing us to rerun DSR or other ad hoc network routing protocols on the *identical* workloads. The physical radio characteristics of each mobile node's network interface, such as antenna gain, transmit power, and receiver sensitivity, were chosen to approximate the Lucent WaveLAN [Tuch 1993] direct sequence spread-spectrum radio.

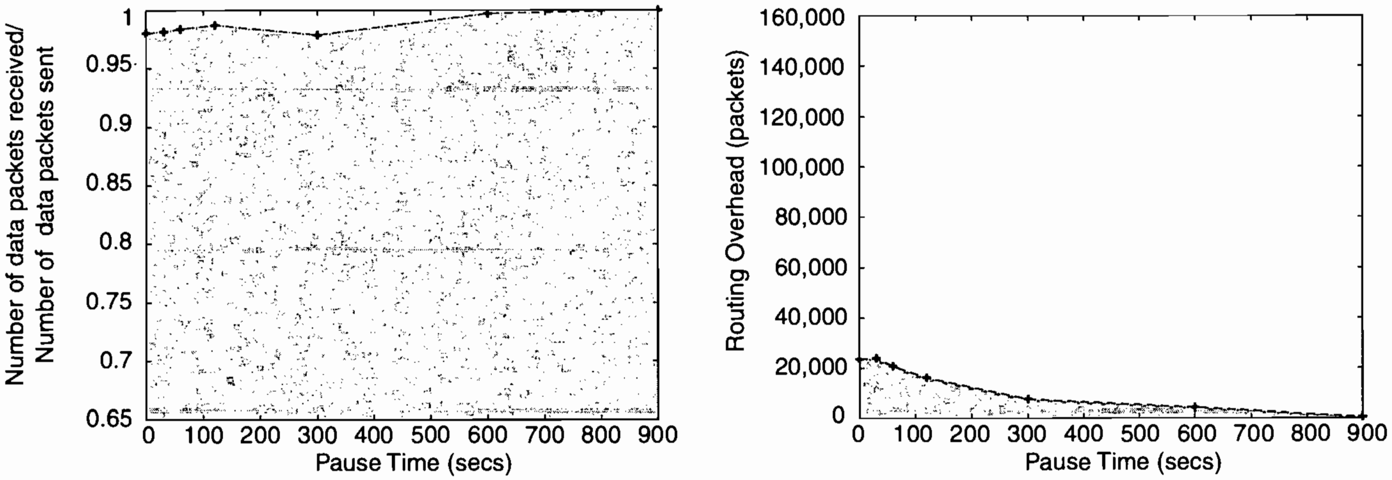
In the random waypoint mobility model [Johnson+ 1996a], each mobile node begins at a random location and and moves independently during the simulation. It remains stationary for a specified period that we call the *pause time* and then moves in a straight line to some new randomly chosen location at a randomly chosen speed up to some maximum speed. Once it reaches that new location, the node again remains stationary for the pause time and then chooses a new random location to proceed to at some new randomly chosen speed; it continues to repeat this behavior throughout the simulation run. We have found that this model can produce large amounts of relative node movement and network topology change, and thus it provides a good movement model with which to stress DSR or any other ad hoc network routing protocol.

Figure 5.9 summarizes the performance of DSR as a function of pause time for two different maximum node movement speeds: Figures 5.9(a) and (b) show the performance for 1 meter/second (about 2 miles/hour), and Figures 5.9(c) and (d) show the performance for 20 meters/second (about 45 miles/hour). For the two respective node movement speeds, the packet delivery ratio—see Figures 5.9(a) and (c)—is the overall percentage of the UDP data packets originated by nodes that were successfully delivered by DSR, and the routing overhead—see Figures 5.9(b) and (d)—is the number of routing overhead packets generated by DSR to achieve this level of data packet delivery. Each point in the graphs represents the average of 10 random movement and communication scenarios for the given pause time. At a pause time of 0 (on the left of each graph), all nodes in the network are in constant motion; as the pause time increases from left to right, the average node movement rate in the network decreases. At a pause time of 900 (on the right of each graph), all nodes are stationary because each simulation was run for 900 simulated seconds of operation of the ad hoc network. The vertical scales on the graphs for 1 meter/second and for 20 meters/second differ in order to make the detail visible.



(a) Packet delivery ratio (1 m/sec)

(b) Routing overhead (1 m/sec)



(c) Packet delivery ratio (20 m/sec)

(d) Routing overhead (20 m/sec)

Figure 5.9. Summary of DSR Performance as a Function of Pause Time

At both movement speeds, DSR delivers almost all data packets, regardless of pause time, with the packet delivery ratio rising to equal 100% at pause time 900 (a stationary network). Similarly, at all pause times routing overhead is low—essentially 0 at pause time 900, rising only slowly as pause time decreases (as the average node mobility rate in the network increases). At the lower movement speed of 1 meter/second, DSR is able to deliver greater than 99.5% of all packets, with most cases delivering greater than 99.8%; the slight decrease at pause time 30 is due to the random generation of the scenarios that we used in the simulations. At the higher movement speed of 20 meters/second, DSR is able to deliver greater than 98% of all packets, even at pause time 0.

5.3.2 DSR Implementation and Testbed Summary

To study the behavior of DSR in a real network, we have implemented DSR in the FreeBSD version of UNIX [FreeBSD] and have experimented with this implementation extensively in an outdoor testbed constructed in Pittsburgh, where Carnegie-Mellon University is located [Maltz+ 1999b]. This has allowed us to experience the full variability and dynamics of real radio propagation, to evaluate user perceptions of applications running over the protocols, and to confirm the results from our simulations.

All of the code implementing DSR resides in the kernel in a module that straddles the IP layer. Conceptually, however, DSR can be thought of as a virtual network interface (*dsr0*) residing below the IP layer. Like other protocol implementation efforts that have used virtual interfaces to hide mobility from the normal network stack [Cheshire+ 1996], the *dsr0* interface accepts packets from the normal IP stack just as any other network interface does, but uses its own mechanisms to arrange for their delivery via the actual physical network interfaces.

To allow multiple types of DSR information to be combined in a single packet, and to allow DSR information to be piggybacked on existing packets, we used a packet format modeled after the *extension header* and *option* format used by IPv6 [Deering+ 1998, Hinden 1996]. In particular, ROUTE REQUESTs, ROUTE REPLYs, and ROUTE ERRORs are each encoded as an option within either a hop-by-hop or an end-to-end extension header, and a DSR source route on a packet is encoded as a separate extension header.

We have experimented extensively with this implementation of DSR in our actual ad hoc networking testbed [Maltz+ 1999b]. Over a period of four months between December 1998 and March 1999, we operated this ad hoc networking testbed daily. Figure 5.10 shows a map of the testbed site and illustrates the layout of the nodes and the mobility in the network. We describe here the movement and communication behavior that we utilized for many of our experiments.

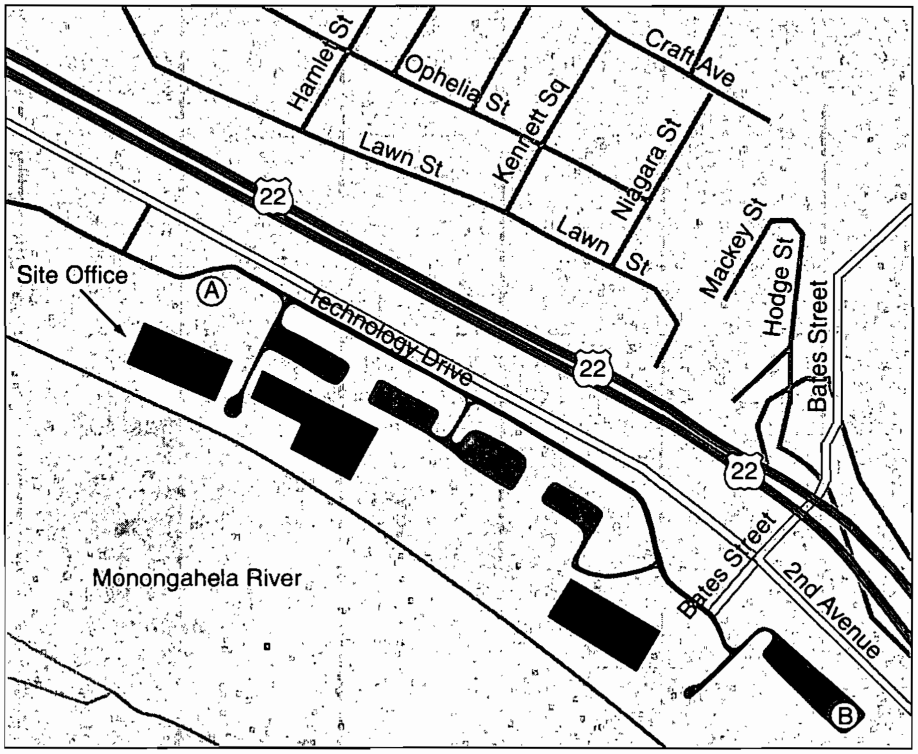


Figure 5.10. Map of the Carnegie-Mellon University Monarch Project DSR Ad Hoc Networking Testbed Site

The testbed consisted of five mobile nodes implemented as cars driving at about 25 MPH (about 10 meters/second), plus two stationary nodes (labeled **A** and **B** in Figure 5.10) separated by a distance of about 700 meters (typically about three radio hops). The mobile node cars moved continuously in a loop along a path starting in the rectangular parking area near **A** (in front of the Site Office building and the building next to it) and then along the shaded roadway to the parking area at **B**; then they turned there and returned to the first parking area, and repeated this route. All cars typically moved along this loop, in nearly constant motion, throughout a run of the testbed. As the cars moved, the route between the two stationary nodes **A** and **B** constantly changed, as did the route between any car and any other car as the cars moved relative to one another. The area used for the testbed was open to general vehicle traffic and had several stop signs, so the actual speed of each node also varied over time, just as it would in any real, deployed network. All of the routes within the ad hoc network were dynamically found and maintained through our DSR ad hoc network routing protocol.

In each car, a laptop computer implemented the DSR routing protocol, served as an endpoint in different higher-layer protocol connections and applications, and allowed local logging of network events on its hard disk. The wireless network interfaces used to form the ad hoc network were WaveLAN PCMCIA PC card radios, operating at 900 MHz, from Lucent Technologies [Tuch 1993]. Each car was also outfitted with a highly accurate global positioning system (GPS) receiver operating in real time kinematic (RTK) mode, providing each node with its own current position to centimeter-level accuracy. During different runs of the testbed, we were thus able to have each mobile node log its own current GPS position as well as the source, destination, and contents of each packet sent or received, along with all significant DSR state transition events. To facilitate additional position logging, the sender's current GPS position was piggybacked on each packet sent, which was logged along with the data of the packet on receipt. The signal strength and signal quality for each received packet (as reported by the WaveLAN hardware) were also logged. Logging this data allowed us to determine whether the protocol was working as intended and helped us diagnose any problems encountered. We have begun attempting to use this data to help with a detailed validation of our simulation models and results [Johnson 1999].

In operating the testbed [Maltz+ 1999b], we experimented with a wide variety of simultaneous data traffic types and network loads, including bulk file transfer, telnet, constant bit rate UDP streams similar to voice or video loading the network, and realtime position and status reporting packets. All realtime GPS "correction" data, required for the RTK GPS operation, was also sent once per second to each node over the ad hoc network from a GPS reference station located on top of the Site Office building shown in the map in Figure 5.10. This system was successfully demonstrated in February and March 1999 to a number of the sponsors and partners in our research, including the DARPA Global Mobile Information Systems Program (GloMo), Lucent Technologies, Bell Atlantic Mobile, and Caterpillar Corporation. In these demonstrations, the mobile node cars were in constant motion, as described above, with the network successfully carrying a large volume of all of these types of traffic. The demonstrations also included interconnection of the ad hoc network to the Internet and integration with Mobile IP, as described in Section 5.2.3.

5.4 RELATED WORK

Research on routing in multihop wireless ad hoc networks dates back at least to 1973, when the U.S. Defense Advanced Research Projects Agency

(DARPA) began the Packet Radio Network (PRNet) project [Jubin+ 1987]. PRNet and its successor, the Survivable Adaptive Networks (SURAN) project [Lauer 1995], generated a substantial number of fundamental results in this area. With the increasing capabilities and decreasing costs of small, portable computers—such as laptops and personal digital assistants (PDAs)—and with the increasing availability of inexpensive wireless network interface devices—such as wireless LAN interfaces packaged as PCMCIA PC cards—a growing number of other research projects in ad hoc networking have developed, some of which are described in other chapters of this book. In our discussion of related work here, we concentrate on research specifically related to the DSR protocol.

The initial design of DSR, including our basic Route Maintenance and Route Discovery mechanisms, was first published in December 1994, with significant additional design details and initial simulation results published in early 1996 [Johnson 1994, Johnson+ 1996a]. As noted at the beginning of this chapter, the design specification for DSR has also been submitted to the IETF MANET (Mobile Ad Hoc Networks) Working Group to help in their efforts to standardize a protocol for routing IP packets in an ad hoc network [Broch+ 1999a, MANET].

The original motivation in the design of DSR came from the ARP [Plummer 1982] used in the TCP/IP suite of protocols in the Internet. ARP is used on Ethernets and other types of networks to find the link layer MAC address of a node on the same subnet as the sender. A node sending a packet to a local IP address, for which it does not yet have the MAC address cached, broadcasts an ARP REQUEST packet on the local subnet link, giving the IP address of the node it is looking for. That node responds with an ARP REPLY packet, giving its MAC address, and all other nodes ignore the REQUEST. If all nodes in an ad hoc network are within wireless transmission range of each other, this is the only routing protocol needed for the ad hoc network. DSR extends this basic ARP behavior by allowing the REQUEST packet (the ROUTE REQUEST rather than an ARP REQUEST) to be propagated multiple hops away through forwarding by neighbor nodes, with the ultimate ROUTE REPLY being returned over multiple hops back to the initiator of the REQUEST.

DSR's nonpropagating ROUTE REQUEST packets are indeed quite similar to the basic ARP REQUEST behavior, except that a mobile node may answer the ROUTE REQUEST from its cache, whereas ARP REQUESTs are normally answered only by the target node itself. With ARP, in cases in which several LANs have been bridged together, the bridge may run "proxy" ARP [Postel 1984], which allows it to answer an ARP REQUEST on behalf of another node (behind the bridge). In this sense, our nonpropagating ROUTE REQUESTs are also similar to proxy ARP; they expand the effective size of a single node's Route Cache by allowing it to make cheap use of the caches of neighboring nodes to reduce the need for propagating ROUTE REQUESTs. Our original implementation of DSR in 1997 also was structured as an extension of ARP, integrated into the existing ARP implementation in the FreeBSD UNIX kernel [FreeBSD] using an extension of the ARP REQUEST and ARP REPLY packet formats. As described in Sections 5.2.5 and 5.3.2, however, we ultimately decided to operate DSR at the network layer rather than at the link layer to allow routing between different heterogeneous networks all forming a single ad hoc network.

DSR is also similar in approach to the source routing discovery mechanism used in the IEEE 802 SRT bridge standard [Perlman 1992]; related mechanisms have also been used in other systems, including FLIP [Kaashoek+ 1993] and SDRP [Estrin+ 1995]. In particular, our ROUTE REQUEST packet serves essentially the same role in Route Discovery as an "all paths explorer" packet does in IEEE 802 source routing bridges. However, in wired networks a bridge can copy such an explorer packet from one network interface onto each of its other interfaces (i.e., to each other link to which this bridge is attached) and be sure that the explorer packet will flood the network in an orderly and complete way. DSR, however, must operate in a wireless ad hoc network, in which nodes forward packets on the same wireless network interface on which they receive them, making such a flood more difficult to implement efficiently. DSR also contains many optimizations designed specifically for the problem of routing in multihop wireless ad hoc networks, and it defines the new Route Maintenance mechanism to quickly and efficiently detect broken links between nodes, allowing alternate routing paths to be taken or new paths to be discovered.

The amateur radio community has worked extensively with routing in wireless networks of (sometimes) mobile hosts [Karn+ 1985], having held an annual packet radio computer networking conference sponsored by the American Radio Relay League (ARRL) since 1981. Amateur packet radio networking originally used only source routing with explicit source routes constructed by the user, although some had considered the possibility of a more dynamic source routing scheme [Garbee 1987]. A system known as NET/ROM was also developed to allow the routing decisions to be automated, using a form of distance vector routing protocol rather than source routing [Frank 1988, Geier+ 1990]. NET/ROM also allows updating of its routing table based on the source address information in the headers of packets that it receives.

Recently, a number of other protocols have been structured around mechanisms similar to the Route Discovery and Route Maintenance mechanisms in DSR. For example, the Signal Stability-Based Adaptive (SSA) routing protocol [Dube+ 1997] and the Associativity-Based Routing (ABR) protocol [Toh 1996] discover routes on demand in a way similar to Route Discovery in DSR, but they attempt to select only long-lived links between nodes where possible; favoring long-lived links helps avoid routes breaking soon after they are discovered but may result in use of routes over a greater number of hops rather than over the shortest routes available. ABR also adds overhead for periodic beacon packets required to monitor link stability. The Ad Hoc On-Demand Distance Vector (AODV) routing protocol [Perkins+ 1999] uses mechanisms similar to DSR's Route Discovery and Route Maintenance, but it uses them to create hop-by-hop routes rather than source routes, as is done in DSR; this use of hop-by-hop routes avoids the source routing header overhead of DSR but prevents or makes difficult many of DSR's route caching and other Route Discovery optimizations and prevents AODV from supporting unidirectional links between nodes. The Zone Routing Protocol (ZRP) [Haas 1997, Haas+ 1998] defines a "routing zone" around each individual node, with a periodic (proactive) protocol such as distance-vector or link-state for routing within a zone and an on-demand protocol such as DSR for routing between zones; the use of routing zones reduces some of the overhead of the Route Discovery procedure, as in DSR, but adds the overhead of maintaining zone membership and routing information within each zone. ZRP may also fail at times to successfully deliver packets with highly mobile nodes because within a zone it does not utilize on-demand operation.

Finally, DSR has been used as a basis for further work by other researchers, including suggested improvements to the Route Discovery mechanism. For example, Ko and Vaidya [Ko+ 1998] proposed an optimization to Route Discovery, known as location-aided routing (LAR), that uses knowledge of the physical (geographical) location of the target node of the Route Discovery (e.g., from GPS) to narrow the area of the network over which the ROUTE REQUEST packets must be propagated. Castañeda and Das [Castañeda+ 1999] have proposed a similar Route Discovery optimization that uses only logical (topological), not physical, location information and thus does not require access to GPS. Holland and Vaidya [Holland+ 1999] recently studied the behavior of TCP in ad hoc networks above the routing layer, using DSR as a routing protocol; their work added explicit interaction between TCP and the Route Discovery and Route Maintenance mechanisms to allow TCP to correctly react to a route failure rather than treat it as network congestion, and to allow it to restart sending as soon as a new route to the destination is discovered.

5.5 CONCLUSION

The Dynamic Source Routing protocol provides excellent performance for routing in multihop wireless ad hoc networks. As shown in our detailed simulation studies and in our implementation of the protocol in a real ad hoc network of cars driving and routing among themselves, DSR has very low routing overhead and is able to correctly deliver almost all originated data packets, even with continuous, rapid motion of all nodes in the network.

A key reason for this good performance is that DSR operates *entirely* on demand [Johnson 1994], with *no* periodic activity of *any kind* required at *any level* within the network. For example, DSR does not use any periodic routing advertisement, link status sensing, or neighbor detection packets; nor does it rely on these functions from any underlying protocols in the network. This entirely on-demand behavior and the lack of periodic activity allow the number of routing overhead packets caused by DSR to scale to *zero,* when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR *automatically* scales to only that needed to track the routes currently in use.

In this chapter, we described the principle mechanisms of *Route Discovery* and *Route Maintenance* used by DSR, and we showed how they enable wireless mobile nodes to automatically form a completely self-organizing and self-configuring network among themselves. Our current work in the Monarch Project at Carnegie-Mellon University includes further improvements to DSR performance—for example, scaling to very large networks—and the addition of new features such as multicast routing and adaptive quality of service reservations and resource management. Our goal is to create an integrated set of protocols that allow mobile computers, and the applications running on them and communicating with them, to make the most efficient use of the best available network connections at any time, seamlessly. DSR is an important component of such a system.

Acknowledgments

The research described in this chapter has been carried out as part of the Monarch Project at Carnegie-Mellon University, whose team members include Josh Broch, Yih-Chun Hu, Jorjeta Jetcheva, David B. Johnson, Qifa Ke, and David A. Maltz (*http://www.monarch.cs.com.edu*). This work was supported in part by the National Science Foundation (NSF) under CAREER Award NCR-9502725, by the Air Force Materiel Command under DARPA contract number F19628-96-C-0061, and by Caterpillar Corporation. David Maltz was also supported under an Intel Graduate Fellowship and an IBM Cooperative Fellowship. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of the editor, the publisher, NSF, AFMC, DARPA, Caterpillar, Intel, IBM, Carnegie-Mellon University, or the U.S. government.

References

[Bantz+ 1994] D.F. Bantz and F.J. Bauchot. Wireless LAN Design Alternatives. *IEEE Network* 8(2):43–53, March/April 1994.

[Bharghavan+ 1994] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LANs. In *Proceedings of ACM SIGCOMM '94*, August 1994, 212–225.

[Braden 1989] R.T. Braden, ed. Requirements for Internet Hosts — Communication Layers. RFC 1122, October 1989.

[Broch+ 1998] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, October 1998, 85–97.

[Broch+ 1999a] J. Broch, D.B. Johnson, and D.A. Maltz. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks.* Internet draft (draft-ietf-manet-dsr-03.txt), October 1999 (work in progress).

[Broch+ 1999b] J. Broch, D.A. Maltz, and D.B. Johnson. Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the Fourth International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99), Workshop on Mobile Computing,* June 1999, 310–315.

[Castañeda+ 1999] R. Castañeda and S.R. Das. Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99),* August 1999.

[Cheshire+ 1996] S. Cheshire and M. Baker. Internet Mobility 4x4. In *Proceedings of ACM SIGCOMM '96*, August 1996, 318–329.

[Deering+ 1998] S.E. Deering and R.M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[Droms 1997] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.

[Dube+ 1997] R. Dube, C.D. Rais, K.-Y. Wang, and S.K. Tripathi. Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks. *IEEE Personal Communications* 4(1):36–45, February 1997.

[Estrin+ 1995] D. Estrin, D. Zappala, T. Li, Y. Rekhter, and K. Varadhan. *Source Demand Routing: Packet Format and Forwarding Version 1.* Internet draft, January 1995 (work in progress).

[Fall+ 1997] K. Fall and K. Varadhan, eds. *ns Notes and Documentation.* The VINT Project, University of California at Berkeley, Laurence Berkeley Laboratory, University of Southern California Information Sciences Institute, and Xerox PARC, November 1997 (available from *http://www-mash.cs.berkeley.edu/ns/*).

[Frank 1988] D.M. Frank. Transmission of IP Datagrams over NET/ROM Networks. In *Proceedings of the ARRL Amateur Radio Seventh Computer Networking Conference*, October 1988, 65–70.

[FreeBSD] *FreeBSD Project.* FreeBSD Home Page (available at *http://www.freebsd.org/*).

[Garbee 1987] B. Garbee. Thoughts on the Issues of Address Resolution and Routing in Amateur Packet Radio TCP/IP Networks. In *Proceedings of the ARRL Amateur Radio Sixth Computer Networking Conference*, August 1987, 56–58.

[Geier+ 1990] J. Geier, M. DeSimio, and B. Welsh. Network Routing Techniques and Their Relevance to Packet Radio Networks. In *Proceedings of the ARRL/CRRL Amateur Radio Ninth Computer Networking Conference*, September 1990, 105–117.

[Haas 1997] Z.J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *Proceedings of the Sixth International Conference on Universal Personal Communications*, October 1997, 562–566.

[Haas+ 1998] Z.J. Haas and M.R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. In *Proceedings of ACM SIGCOMM '98,* September 1998, 167–177.

[Hinden 1996] R.M. Hinden. IP Next Generation Overview. *Communications of the ACM* 39(6):61–71, June 1996.

[Holland+ 1999] G. Holland and N. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99),* August 1999, 219–230.

[IEEE 1997] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-1997.* The Institute of Electrical and Electronics Engineers, New York, 1997.

[Johansson+ 1999] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Routing Protocols for Mobile Ad-Hoc Networks—A Comparative Performance Analysis. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, August 1999.

[Johnson 1994] D.B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994, 158–163.

[Johnson 1995] D.B. Johnson. Scalable Support for Transparent Mobile Host Internetworking. *Wireless Networks* 1(3):311–321, October 1995.

[Johnson 1999] D.B. Johnson. Validation of Wireless and Mobile Network Models and Simulation. In *Proceedings of the DARPA/NIST Workshop on Validation of Large Scale Network Models and Simulation*, May 1999.

[Johnson+ 1996a] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, T. Imielinski and H. Korth, eds., Kluwer Academic Publishers, Norwell, Mass., 1996, 153–181.

[Johnson+ 1996b] D.B. Johnson and D.A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications* 3(1):34–42, February 1996.

[Jubin+ 1987] J. Jubin and J.D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE* 75(1):21–32, January 1987.

[Kaashoek+ 1993] M.F. Kaashoek, R. van Renesse, H. van Staveren, and A.S. Tanenbaum. FLIP: An Internetwork Protocol for Supporting Distributed Systems. *ACM Transactions on Computer Systems* 11(1):73–106, February 1993.

[Karn+ 1985] P.R. Karn, H.E. Price, and R.J. Diersing. Packet Radio in the Amateur Service. *IEEE Journal on Selected Areas of Communications* SAC-3(3):431–439, May 1985.

[Karn 1990] P. Karn. MACA—A New Channel Access Method for Packet Radio. In *Proceedings of the ARRL/CRRL Amateur Radio Ninth Computer Networking Conference*, September 1990, 134–140.

[Katz+ 1996] R.H. Katz and E.A. Brewer. The Case for Wireless Overlay Networks. In *Proceedings of the SPIE Multimedia and Networking Conference (MMNC '96)*, January 1996.

[Ko+ 1998] Y.-B. Ko and N. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, October 1998, 66–75.

[Lauer 1995] G. S. Lauer. Packet-Radio Routing. In *Routing in Communications Networks*, M. Steenstrup, ed., Prentice-Hall, Englewood Cliffs, N.J., 1995, 351–396.

[Maltz+ 1999a] D.A. Maltz, J. Broch, J. Jetcheva, and D.B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas of Communications* 17(8):1439–1453, August 1999.

[Maltz+ 1999b] D.A. Maltz, J. Broch, and D.B. Johnson. *Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed.* Technical report CMU-CS-99-116, School of Computer Science, Carnegie-Mellon University, Pittsburgh, March 1999.

[MANET] IETF MANET Working Group. *Mobile Ad Hoc Networks (MANET)* (Working Group Charter available at *http://www.ietf.org/html.charters/manet-charter.html*).

[Monarch] Carnegie-Mellon University Monarch Project. CMU Monarch Project Home Page (available at *http://www.monarch.cs.cmu.edu/*).

[Perkins 1996] C. Perkins, ed. IP Mobility Support. RFC 2002 (proposed standard), October 1996.

[Perkins+ 1999] C. Perkins and E.M. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proceedings of the Second Annual IEEE Workshop on Mobile Computing Systems and Applications*, February 1999, 90–100.

[Perlman 1992] R. Perlman. *Interconnections: Bridges and Routers.* Addison-Wesley, Reading, Mass., 1992.

[Plummer 1982] D.C. Plummer. An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48-bit Ethernet Addresses for Transmission on Ethernet Hardware. RFC 826, November 1982.

[Postel 1981a] J.B. Postel, ed. Internet Protocol. RFC 791, September 1981.

[Postel 1981b] J.B. Postel, ed. Transmission Control Protocol. RFC 793, September 1981.

[Postel 1984] J.B. Postel. Multi-LAN Address Resolution. RFC 925, October 1984.

[Rappaport 1996] T.S. Rappaport. *Wireless Communications: Principles and Practice.* Prentice-Hall, Englewood Cliffs, N.J., 1996.

[Toh 1996] C.-K. Toh. A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing. In *Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, March 1996, 480–486.

[Tuch 1993] B. Tuch. Development of WaveLAN, an ISM Band Wireless LAN. *AT&T Technical Journal* 72(4):27–33, July/August 1993.

[Turner 1990] P. Turner. NetWare Communications Processes. *NetWare Application Notes,* Novell Research, September 1990, 25–91.

[Wright+ 1995] G.R. Wright and W.R. Stevens. *TCP/IP Illustrated, Volume 2: The Implementation.* Addison-Wesley, Reading, Mass., 1995.