Dynamic Source Routing in Ad Hoc Wireless Networks

David B. Johnson
David A. Maltz

Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891

dbj@cs.cmu.edu

Abstract

An *ad hoc* network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. In such an environment, it may be necessary for one mobile host to enlist the aid of other hosts in forwarding a packet to its destination, due to the limited range of each mobile host's wireless transmissions. This paper presents a protocol for routing in ad hoc networks that uses *dynamic source routing*. The protocol adapts quickly to routing changes when host movement is frequent, yet requires little or no overhead during periods in which hosts move less frequently. Based on results from a packet-level simulation of mobile hosts operating in an ad hoc network, the protocol performs well over a variety of environmental conditions such as host density and movement rates. For all but the highest rates of host movement simulated, the overhead of the protocol is quite low, falling to just 1% of total data packets transmitted for moderate movement rates in a network of 24 mobile hosts. In all cases, the difference in length between the routes used and the optimal route lengths is negligible, and in most cases, route lengths are on average within a factor of 1.01 of optimal.

1. Introduction

Mobile hosts and wireless networking hardware are becoming widely available, and extensive work has been done recently in integrating these elements into traditional networks such as the Internet. Oftentimes, however, mobile users will want to communicate in situations in which no fixed wired infrastructure such as this is available, either because it may not be economically practical or physically possible to provide the necessary infrastructure or because the expediency of the situation does not permit its installation. For example, a class of students may need to interact during a lecture, friends or business associates may run into each other in an airport terminal and wish to share files, or a group of emergency rescue workers may need to be quickly deployed after an earthquake or flood. In such situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of wireless network is known as an *ad hoc network*.

A version of this paper will appear as a chapter in the book *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Kluwer Academic Publishers, 1996.

If only two hosts, located closely together, are involved in the ad hoc network, no real routing protocol or routing decisions are necessary. In many ad hoc networks, though, two hosts that want to communicate may not be within wireless transmission range of each other, but could communicate if other hosts between them also participating in the ad hoc network are willing to forward packets for them. For example, in the network illustrated in Figure 1, mobile host C is not within the range of host A's wireless transmitter (indicated by the circle around A) and host A is not within the range of host C's wireless transmitter. If A and C wish to exchange packets, they may in this case enlist the services of host B to forward packets for them, since B is within the overlap between A's range and C's range. Indeed, the routing problem in a real ad hoc network may be more complicated than this example suggests, due to the inherent nonuniform propagation characteristics of wireless transmissions and due to the possibility that any or all of the hosts involved may move at any time.

Routing protocols in conventional wired networks generally use either *distance vector* or *link state* routing algorithms, both of which require periodic routing advertisements to be broadcast by each router. In distance vector routing [9, 17, 26, 27, 29], each router broadcasts to each of its neighbor routers its view of the distance to all hosts, and each router computes the shortest path to each host based on the information advertised by each of its neighbors. In link state routing [10, 16, 18], each router instead broadcasts to all other routers in the network its view of the status of each of its adjacent network links, and each router then computes the shortest distance to each host based on the complete picture of the network formed from the most recent link information from all routers. In addition to its use in wired networks, the basic distance vector algorithm has also been adapted for routing in wireless ad hoc networks, essentially treating each mobile host as a router [11, 19, 25].

This paper describes the design and performance of a routing protocol for ad hoc networks that instead uses *dynamic source routing* of packets between hosts that want to communicate. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet; the sender explicitly lists this route in the packet's header, identifying each forwarding "hop" by the address of the next node to which to transmit the packet on its way to the destination host. Source routing has been used in a number of contexts for routing in wired networks, using either statically defined or dynamically constructed source routes [4, 5, 12, 20, 22, 28], and has been used with statically configured routes in the Tucson Amateur Packet Radio (TAPR) work for routing in a wireless network [14]. The protocol presented here is explicitly designed for use in the wireless environment of an ad hoc network. There are no periodic router advertisements in the protocol. Instead, when a host needs a route to another host, it dynamically determines one based on cached information and on the results of a *route discovery* protocol.

We believe our dynamic source routing protocol offers a number of potential advantages over conventional routing protocols such as distance vector in an ad hoc network. First, unlike conventional routing protocols, our protocol uses no periodic routing advertisement messages, thereby reducing network bandwidth overhead, particularly during periods when little or no significant host movement is taking place. Battery power is also conserved on the mobile hosts, both by not sending the advertisements and by not needing to receive them (since a host could otherwise reduce its power usage by putting itself into "sleep" or "standby" mode when not busy with other tasks). Distance vector and link state routing, on the other hand, must continue to send advertisements even when nothing changes, so that other mobile hosts will continue to consider those routes or network links as valid. In addition, many of the "links" between routers seen by the routing algorithm may be redundant [11]. Wired networks are usually explicitly configured to have only one (or a small number) of routers connecting any two networks, but there are no explicit links in an ad hoc network, and all communication is by broadcast transmissions. The redundant paths in a wireless environment unnecessarily increase the size of routing updates that must be sent over the network, and increase the CPU overhead required to process each update and to compute new routes.

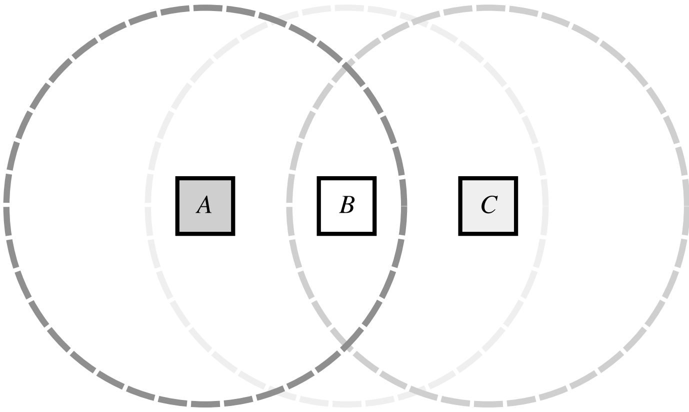

Figure 1 A simple ad hoc network of three wireless mobile hosts

In addition, conventional routing protocols based on link state or distance vector algorithms may compute some routes that do not work. In a wireless environment, network transmission between two hosts does not necessarily work equally well in both directions, due to differing propagation or interference patterns around the two hosts [1, 15]. For example, with distance vector routing, even though a host may receive a routing advertisement from another mobile host, packets it might then transmit back to that host for forwarding may not be able to reach it. Our protocol does not require transmissions between hosts to work bidirectionally, although we do make use of it when afforded, for example, by MAC-level protocols such as MACA [13] or MACAW [2] that ensure it.

Finally, conventional routing protocols are not designed for the type of dynamic topology changes that may be present in ad hoc networks. In conventional networks, links between routers occasionally go down or come up, and sometimes the cost of a link may change due to congestion, but routers do not generally move around dynamically. In an environment with mobile hosts as routers, though, convergence to new, stable routes after such dynamic changes in network topology may be slow, particularly with distance vector algorithms [20]. Our dynamic source routing protocol is able to adapt quickly to changes such as host movement, yet requires no routing protocol overhead during periods in which such changes do not occur.

Section 2 of this paper details our assumptions about the network and the mobile hosts. The basic operation of our dynamic source routing protocol is described in Section 3, and optimizations to this basic operation are described in Section 4. In Section 5, we present a preliminary evaluation of the performance of our protocol, based on a packet-level simulation. In Section 6, we discuss related protocols for wireless network routing and for source routing, and in Section 7, we present conclusions and future work.

2. Assumptions

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

We refer to the number of hops necessary for a packet to reach from any host located at one extreme edge of the network to another host located at the opposite extreme, as the *diameter* of the network. For example, the diameter of the ad hoc network depicted in Figure 1 is two. We assume that the diameter of an ad hoc network will be small but may often be greater than one.

Hosts within the ad hoc network may move at any time without notice, but we assume that the speed with which hosts move is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. In particular, we assume that hosts do not continuously move so rapidly as to make the flooding of every packet the only possible routing protocol.

We assume that hosts can enable a *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 destination address. Although we do not require this facility, it is common in current LAN hardware for broadcast media including wireless, and some of our optimizations take advantage of it. 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 believe that portions of the protocol are also suitable for implementation directly in hardware or within a programmable network interface unit to avoid this overhead on the CPU.

3. Basic Operation

3.1. Overview

To send a packet to another host, the sender constructs a *source route* in the packet's header, giving the address of each host in the network through which the packet should be forwarded in order to reach the destination host. The sender then transmits the packet over its wireless network interface to the first hop identified in the source route. When a host receives a packet, if this host is not the final destination of the packet, it simply transmits the packet to the next hop identified in the source route in the packet's header. Once the packet reaches its final destination, the packet is delivered to the network layer software on that host.

Each mobile host participating in the ad hoc network maintains a *route cache* in which it caches source routes that it has learned. When one host sends a packet to another host, the sender first checks its route cache for a source route to the destination. If a route is found, the sender uses this route to transmit the packet. If no route is found, the sender may attempt to discover one using the *route discovery* protocol. While waiting for the route discovery to complete, the host may continue normal processing and may send and receive packets with other hosts. The host may buffer the original packet in order to transmit it once the route is learned from route discovery, or it may discard the packet, relying on higher-layer protocol software to retransmit the packet if needed. Each entry in the route cache has associated with it an expiration period, after which the entry is deleted from the cache.

While a host is using any source route, it monitors the continued correct operation of that route. For example, if the sender, the destination, or any of the other hosts named as hops along a route move out of wireless transmission range of the next or previous hop along the route, the route can no longer be used to reach the destination. A route will also no longer work if any of the hosts along the route should fail or be powered off. This monitoring of the correct operation of a route in use we call *route maintenance*. When route maintenance detects a problem with a route in use, route discovery may be used again to discover a new, correct route to the destination.

This section describes the basic operation of route discovery and route maintenance. Optimizations to this basic operation of the protocol are then described in Section 4.

3.2. Route Discovery

Route discovery allows any host in the ad hoc network to dynamically discover a route to any other host in the ad hoc network, whether directly reachable within wireless transmission range or reachable through one or more intermediate network hops through other hosts. A host initiating a route discovery broadcasts a *route request* packet which may be received by those hosts within wireless transmission range of it. The route request packet identifies the host, referred to as the *target* of the route discovery, for which the route is requested. If the route discovery is successful the initiating host receives a *route reply* packet listing a sequence of network hops through which it may reach the target.

In addition to the address of the original initiator of the request and the target of the request, each route request packet contains a *route record*, in which is accumulated a record of the sequence of hops taken by the route request packet as it is propagated through the ad hoc network during this route discovery. Each route request packet also contains a unique *request id*, set by the initiator from a locally-maintained sequence number. In order to detect duplicate route requests received, each host in the ad hoc network maintains a list of the $\langle$initiator address, request id$\rangle$ pairs that it has recently received on any route request.

When any host receives a route request packet, it processes the request according to the following steps:

1. If the pair $\langle$initiator address, request id$\rangle$ for this route request is found in this host's list of recently seen requests, then discard the route request packet and do not process it further.

2. Otherwise, if this host's address is already listed in the route record in the request, then discard the route request packet and do not process it further.

3. Otherwise, if the target of the request matches this host's own address, then the route record in the packet contains the route by which the request reached this host from the initiator of the route request. Return a copy of this route in a *route reply* packet to the initiator.

4. Otherwise, append this host's own address to the route record in the route request packet, and re-broadcast the request.

The route request thus propagates through the ad hoc network until it reaches the target host, which then replies to the initiator. The original route request packet is received only by those hosts within wireless transmission range of the initiating host, and each of these hosts propagates the request if it is not the target and if the request does not appear to this host to be redundant. Discarding the request because the host's address is already listed in the route record guarantees that no single copy of the request can propagate around a loop. Also discarding the request when the host has recently seen one with the same $\langle$initiator address, request id$\rangle$ removes later copies of the request that arrive at this host by a different route.

In order to return the *route reply* packet to the initiator of the route discovery, the target host must have a route to the initiator. If the target has an entry for this destination in its route cache, then it may send the route reply packet using this route in the same way as is used in sending any other packet (Section 3.1). Otherwise, the target may reverse the route in the route record from the route request packet, and use this route to send the route reply packet. This, however, requires the wireless network communication between each of these pairs of hosts to work equally well in both directions, which may not be true in some environments or with some MAC-level protocols. An alternative approach, and the one we have currently adopted, is for this host to piggyback the route reply packet on a route request targeted at the initiator of the route discovery to which it is replying. This use of piggybacking is described in Section 4.2.

3.3. Route Maintenance

Conventional routing protocols integrate route discovery with route maintenance by continuously sending periodic routing updates. If the status of a link or router changes, the periodic updates will eventually reflect the changes to all other routers, presumably resulting in the computation of new routes. However, using route discovery, there are no periodic messages of any kind from any of the mobile hosts. Instead, while a route is in use, the route maintenance procedure monitors the operation of the route and informs the sender of any routing errors.

Since wireless networks are inherently less reliable than wired networks [1], many wireless networks utilize a hop-by-hop acknowledgement at the data link level in order to provide early detection and retransmission of lost or corrupted packets. In these networks, route maintenance can be easily provided, since at each hop, the host transmitting the packet for that hop can determine if that hop of the route is still working. If the data link level reports a transmission problem for which it cannot recover (for example, because the maximum number of retransmissions it is willing to attempt has been exceeded), this host sends a *route error* packet to the original sender of the packet encountering the error. The route error packet contains the addresses of the hosts at both ends of the hop in error: the host that detected the error and the host to which it was attempting to transmit the packet on this hop. When a route error packet is received, the hop in error is removed from this host's route cache, and all routes which contain this hop must be truncated at that point.

If the wireless network does not support such lower-level acknowledgements, an equivalent acknowledgement signal may be available in many environments. After sending a packet to the next hop mobile host, the sender may be able to hear that host transmitting the packet again, on its way further along the path, if it can operate its wireless network interface in promiscuous mode. For example, in Figure 1, host *A* may be able to hear *B*'s transmission of the packet on to *C*. This type of acknowledgement is known as a *passive acknowledgement* [11]. In addition, existing transport or application level replies or acknowledgements from the original destination could also be used as an acknowledgement that the route (or that hop of the route) is still working. As a last resort, a bit in the packet header could be included to allow a host transmitting a packet to request an explicit acknowledgement from the next-hop receiver. If no other acknowledgement signal has been received in some time from the next hop on some route, the host could use this bit to inexpensively probe the status of this hop on the route.

As with the return of a route reply packet, a host must have a route to the sender of the original packet in order to return a route error packet to it. If this host has an entry for the original sender in its route cache, it may send the route error packet using that route. Otherwise, this host may reverse the route from the packet in error (the route by which the packet reached this host) or may use piggybacking as in the case of a route reply packet (Section 4.2). Another option in the case of returning a route error packet is for this host to save the route error packet locally in a buffer, perform a route discovery for the original sender, and then send the route error packet using that route when it receives the route reply for this route discovery. This option cannot be used for returning a route reply packet, however, since then neither host would ever be able to complete a route discovery for the other, if neither initially had a route cache entry for the other.

Route maintenance can also be performed using end-to-end acknowledgements rather than the hop-by-hop acknowledgements described above, if the particular wireless network interfaces or the environment in which they are used are such that wireless transmissions between two hosts do not work equally well in both directions. As long as some route exists by which the two end hosts can communicate (perhaps different routes in each direction), route maintenance is possible. In this case, existing transport or application level replies or acknowledgements from the original destination, or explicitly requested network level acknowledgements, may be used to indicate the status of this host's route to the other host. With hop-by-hop acknowledgements, the particular hop in error is indicated in the route error packet, but with end-to-end acknowledgements, the sender may only assume that the last hop of the route to this destination is in error.

4. Optimizations

A number of optimizations are possible to the basic operation of route discovery and route maintenance as described in Section 3.2, that can reduce the number of overhead packets and can improve the average efficiency of the routes used on data packets. This section discusses some of those optimizations.

4.1. Full Use of the Route Cache

The data in a host's route cache may be stored in any format, but the active routes in its cache in effect form a tree of routes, rooted at this host, to other hosts in the ad hoc network. For example, Figure 2 shows an ad hoc network of five mobile hosts, in which mobile host *A* has earlier completed a route discovery for mobile host *D* and has cached the route to *D* through *B* and *C*. Since hosts *B* and *C* are on the route to *D*, host *A* also learns the route to both of these hosts from its route discovery for *D*. If *A* later performs a route discovery and learns the route to *E* through *B* and *C*, it can represent this in its route cache with the addition of the single new hop from *C* to *E*. If *A* then learns it can reach *C* in a single hop (without needing to go through *B*), *A* can use this new route to *C* to also shorten the routes to *D* and *E* in its route cache.

A host can add entries to its route cache any time it learns a new route. In particular, when a host forwards a data packet as an intermediate hop on the route in that packet, the forwarding host is able to observe the entire route in the packet. Thus, for example, when host *B* forwards packets from *A* to *D*, *B* can add the route information from that packet to its own route cache. If a host forwards a route reply packet, it can also add the route information from the route record being returned in that route reply, to its own route cache. Finally, since all wireless network transmissions are inherently broadcast, a host may be able configure its network interface into promiscuous receive mode, and can then add to its route cache the route information from any data or route reply packet it can overhear.

A host may use its route cache to avoid propagating a route request packet received from another host. In particular, suppose a host receives a route request packet for which it is not the target and is not already listed in the route record in the packet, and for which the pair $\langle$initiator address, request id$\rangle$ is not found in its list of recently seen requests; if the host has a route cache entry for the target of the request, it may append this cached route to the accumulated route record in the packet, and may return this route in a route reply packet to the initiator without propagating (re-broadcasting) the route request. Thus, for example, if mobile host *F* needs to send a packet to mobile host *D*, it will initiate a route discovery and broadcast a route request packet. If this broadcast is received by *A*, *A* can simply return a route reply packet to *F* containing the complete route to *D* consisting of the sequence of hops *A*, *B*, *C*, and *D*.

A particular problem can occur, however, when several mobile hosts receive the initiator's broadcast of the route request packet, and all reply based on routes found in their route caches. In Figure 2, for example, if both *A* and *B* receive *F*'s route request broadcast, they will both be able to reply from their route caches, and will both send their replies at about the same time since they both received the broadcast at about the same time. Particularly when more than the two mobile hosts in this example are involved, these simultaneous replies from the mobile hosts receiving the broadcast may create packet collisions among some or all of these replies and may cause local congestion in the wireless network. In addition, it will often be the case

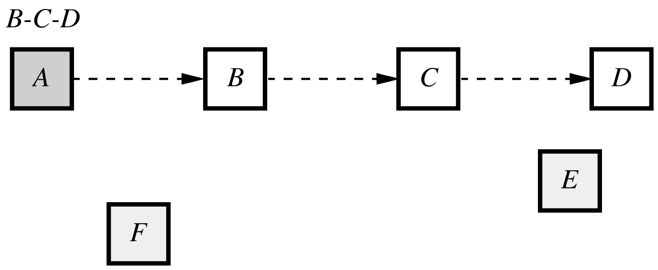

Figure 2 An example ad hoc network illustrating use of the route cache

that the different replies will indicate routes of different lengths. For example, *A*'s reply will indicate a route to *D* that is one hop longer than that in *B*'s reply.

We avoid the problems of many simultaneous replies and attempt to eliminate replies indicating routes longer than the shortest reply, by causing each mobile host to delay slightly before replying from its cache. Before replying from its route cache, a host performs the following actions:

1. Pick a delay 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 host's reply, r is a random number between 0 and 1, and H is a small constant delay to be introduced per hop.

2. Delay transmitting the route reply from this host for a period of d.

3. Within this delay period, promiscuously receive all packets at this host. If a packet is received by this host during the delay period addressed to the target of this route discovery (the target is the final destination address for the packet, through any sequence of intermediate hops), and if the length of the route on this packet is less than h, then cancel the delay and do not transmit the route reply from this host; this host may infer that the initiator of this route discovery has already received a route reply, giving an equal or better route.

Another problem that can occur when hosts reply to route requests from their cache, is the formation of a loop in the route sent in the route reply packet. The route record in the route request cannot contain a loop, and no entry in a route cache ever is set to a route containing a loop, yet the concatenation of the route record and the entry from the replying host's route cache for the target may contain a loop. For example, in Figure 2, if host *B* does not have a route cache entry for *D*, it will need to initiate a route discovery before sending a packet to *D*. In this case, *A* could immediately reply from its route cache with the route to *D* through *B* and *C*. This, however, would form the concatenated route of *A-B-C-D* for *B* to then use in sending packets to *D*, creating a loop from *B* to *A* and then back to *B*. In order to avoid this problem, if a host receives a route request and is not the target of the request but could reply from its cache, the host instead discards the request if the route in its reply would contain a loop; this restriction also implies that a host will only reply from its cache with a route in which the host itself is on the route, at the end of the route recorded in the route request packet and at the beginning of the path obtained from the host's route cache.

As a last optimization involving full use of the route cache, we have added the ability for the initiator of a route request to specify in the request packet, the maximum number of hops over which the packet may be propagated. If another host near the initiator has a cache entry for the target of a route request, the propagation of many redundant copies of the route request can be avoided if the initiator can explicitly limit the request's propagation when it is originally sent. Currently, we use this ability during route discovery as follows:

1. To perform a route discovery, initially send the route request with a hop limit of one. We refer to this as a nonpropagating route request.

2. If no route reply is received from this route request after a small timeout period, send a new route request with the hop limit set to a predefined "maximum" value for which it is assumed that all useful routes in the ad hoc network are less than this limit (currently 10).

This procedure uses the hop limit on the route request packet to inexpensively check if the target is currently within wireless transmitter range of the initiator or if another host within range has a route cache entry for this target (effectively using the caches of this host's neighbors as an extension of its own cache). Since the initial request is limited to one network hop, the timeout period before sending the propagating request can be quite small. This mechanism could also be used to implement an "expanding ring" search for the target, in which the hop limit is gradually increased in subsequent retransmissions of the route request for this target, but we have not yet experimented with this approach.

4.2. Piggybacking on Route Discoveries

As described in Section 3.2, when one host needs to send a packet to another, if the sender does not have a route cached to the destination host, it must initiate a separate route discovery, either buffering the original packet until the route reply is returned, or discarding it and relying on a higher-layer protocol to retransmit it if needed. The delay for route discovery and the total number of packets transmitted can be reduced by allowing data to be piggybacked on route request packets. Since some route requests may be propagated widely within the ad hoc network, though, the amount of data piggybacked must be limited. We currently use piggybacking when sending a route reply or a route error packet, since both are naturally small in size; small data packets such as the initial SYN packet opening a TCP connection [23] could also easily be piggybacked, but we have not yet experimented with this option.

One problem, however, arises when piggybacking on route request packets. If the route request is received by some host and is replied to based on the host's route cache without propagating the request (Section 4.1), the piggybacked data would be lost when the host discards the route request. In this case, before discarding the packet, the host must construct a new packet containing the piggybacked data from the route request packet, setting the route in this packet from the route being returned in the route reply. The route should appear as if the new packet had been sent by the initiator of the route discovery and had been forwarded normally to this host: the first portion of the route is taken from the accumulated route record in the route request packet, and the remainder of the route is taken from this host's route cache. The sender address in the packet should also be set to the initiator of the route discovery.

4.3. Reflecting Shorter Routes

While two hosts are communicating with each other using cached routes, it is desirable for the hosts to begin using shorter routes if the hosts move sufficiently closer together. In many cases, the basic route maintenance procedure is sufficient to accomplish this, since if one of the hosts moves enough to allow the route to be shortened, it will likely also move out of transmission range of the first hop on the existing route.

An improvement to this method of reflecting shorter routes is possible if hosts operate their network interfaces in promiscuous receive mode. Suppose somewhere in the forwarding of a packet, mobile host *B* transmits a packet to *C*, with *D* being the next hop after *C* in the route in the packet, as illustrated in Figure 3. If *D* receives this packet, it can examine the packet header to see that the packet reached it from *B* in one hop rather than two as intended by the route in the packet. In this case, *D* may infer that route may be shortened to exclude the intermediate hop through *C*. *D* then sends an unsolicited route reply packet to the original sender of the packet, informing it that it can now reach *D* in one hop from *B*. As with other route reply packets, other hosts which also receive this route reply (in particular, other hosts also operating in promiscuous receive mode) may also incorporate this change into their route caches. We believe that this method will ensure that the shortest routes will be used, although we have not yet added this method to our simulator.

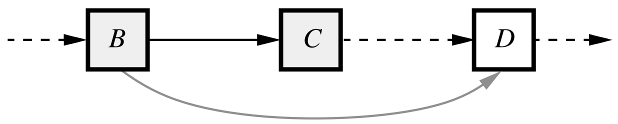

Figure 3 Mobile host *D* notices that the route can be shortened

4.4. Improved Handling of Errors

One common error condition that must be handled in an ad hoc network is the case in which the network effectively becomes partitioned. That is, two hosts that wish to communicate are not within transmission range of each other, and there are not enough other mobile hosts between them to form a sequence of hops through which they can forward packets. If a new route discovery were to be initiated for each packet sent by a host in this situation, a large number of unproductive route request packets would be propagated throughout the subset of the ad hoc network reachable from this host. In order to reduce the overhead from such route discoveries, we use exponential backoff to limit the rate at which new route discoveries may be initiated from any host for the same target. If the host attempts to send additional data packets to this same host more frequently than this limit, the subsequent packets may be buffered until a route reply is received, but they do 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 hosts to limit the rate at which ARP requests are sent to any single IP address [3].

An additional optimization possible to improve the handling of errors is to use promiscuous receive mode to allow hosts to eavesdrop on route error packets being sent to other hosts. For example, Figure 4 shows the return of a route error packet to mobile host *A* from host *B*. If hosts *C*, *D*, and *E* are operating in promiscuous receive mode, they will be able to receive the route error packet. Since a route error packet names both ends of the route hop causing the error, any host receiving the error packet can update its route cache to reflect the fact that the two hosts indicated in the packet can no longer directly communicate. A host receiving a route error packet can simply search its route cache for any routes using this hop, and for each such route found, the route is truncated at this hop. All hosts on the route before this hop are still reachable on this route, but subsequent hosts are not.

It is possible, however, that a route error packet being returned to the original sender of a data packet may take a different route to the sender than the data packet took to the point at which the error was encountered. For example, in a network environment in which radio transmission between two hosts does not work equally well in both directions, a route discovery used by the host returning the route error packet may discover a different route back to the original sender. Thus, some hosts that may have cached the route in error may not be able to receive the route error packet, even using promiscuous receive mode. This situation can be improved by extending the handling of route error packets such that, once the route error packet reaches the original sender of the data packet, that original sender also retransmits the route error packet back to the point of error along the path originally used for the data packet, if this path differs from the one along which it received the route error packet.

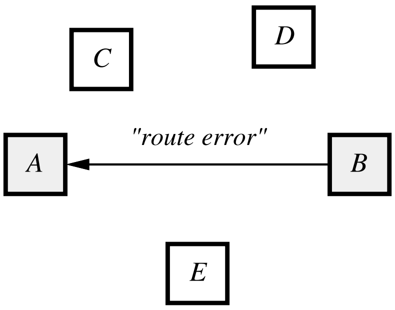

Figure 4 Mobile host *B* is returning a route error packet to *A*

A last optimization to improve the handling of errors is to support the caching of "negative" information in a host's route cache. Suppose, in Figure 4, that none of these optimizations for handling errors are in use. When *A* receives *B*'s route error packet, it may initiate a route discovery in order to find a new route to the same target. However, if hosts *C*, *D*, or *E* have an entry in their route cache for this target, they will likely reply to *A* from their cache with a cached copy of the same route that *A* just removed from its cache. If instead, *A* could enter into its cache when it receives *B*'s route error packet, an indication that this hop is not currently working (rather than simply removing this hop from any routes currently in its cache), then *A* could ignore future replies from *C*, *D*, and *E* that include this hop from their caches. A short expiration period must be placed on this negative cached information, since while this entry is in its cache, *A* will otherwise refuse to allow this hop in any route entries in its cache, even if this hop begins working again.

We have not currently included this caching of negative information in our simulation, due to the difficulty of picking a suitable expiration period, and since it appears to not be necessary in most cases, if hosts also promiscuously receive route error packets. For example, in Figure 4, if *C*, *D*, and *E* also receive *B*'s route error packet, they will have removed this hop from their caches before *A*'s new route discovery is initiated, thus avoiding the problem.

5. Performance Evaluation

5.1. The Simulation

To evaluate the performance of our dynamic source routing protocol, we constructed a packet-level simulator which allowed us to observe and measure the protocol's performance under a variety of conditions. In addition to a number of parameter choices in the protocol, the simulator allowed us to vary certain environmental factors such as the number of mobile hosts, the pattern and speed of host movement, and the distribution of the hosts in space. The simulator implements the basic protocol along with all optimizations described in Section 4 with the exceptions of reflecting shorter routes (Section 4.3) and the caching of negative information (Section 4.4).

The basic simulation parameters were chosen to model an ad hoc network consisting of a collection of mobile hosts moving around in a medium-sized room. The area in which the hosts move is square, 9 meters on a side. Each host moves with a velocity between 0.3 and 0.7 meters per second (somewhere between a slow walk and a quick stroll), and each transceiver has a range of 3 meters. These parameters could represent, for example, a group of hosts using diffuse infrared transceivers, or since the parameters are related and can be scaled linearly, they could also represent a number of cars with radio-equipped portables driving (very quickly) around an area of 1 square kilometer.

Each host is initially placed at a random position within in the simulation area. As the simulation progresses, each host pauses at its current location for a period, which we call the *pause time*, and then randomly chooses a new location to move to and a velocity between 0.3 and 0.7 meters per second at which to move there. Each host continues this behavior, alternately pausing and moving to a new location, for the duration of the simulation. Using this model, hosts appear to wander through the room with their restlessness determined by the pause time constant.

Whenever a host transmits a packet, some method must be used to determine which of the surrounding hosts will receive a copy of the packet. While our simulation's transmission model is admittedly simple, it still allows us to estimate the basic performance of the protocol. In the simulation, each host can be the originator of up to 3 conversations at a time, with the other participant in the conversation chosen randomly from among the other hosts. In actual use, we would expect hosts to communicate mostly with a small common subset of the available hosts (such as servers), which would reduce the number of route discoveries required. While a host is the initiator of less than three conversations, it initiates a new conversation with a randomly chosen partner after an average period of 15 seconds (using an exponential distribution).

Each conversation lasts for a predetermined number of packets, the number being chosen from a geometric distribution with an average of 1000 packets. Again, in actual use, we would expect some (or all) of the conversations to be of longer duration than this, depending on the mix of network application programs in use. Short conversations, however, give a more conservative measure of the performance of the protocol, since more route discoveries are required as each host more frequently changes which other hosts it is communicating with.

During a conversation, packets are sent to the partner at exponentially distributed sending times with a uniformly chosen average rate between 2 and 5 packets per second. Packet lengths are chosen with a distribution such that 70% of the packets are long packets (1000 bytes) and the remainder are short packets (32 bytes). To give the appearance of a two way conversation, the partner sends a return packet back to the originator for every packet it receives. These return packets have the same length distribution as the originator's packets.

Packet transmission in the simulator is based on a network using link level acknowledgements at each hop. Transmissions to a host that is out of range always fail while transmissions to a host in range fail with a probability of 5%. The data link layer in the simulator will make up to 2 retransmission attempts if the first transmission fails. Each time a sender transmits a packet, the other hosts in range of the sender each have a 95% chance of overhearing the packet. The bandwidth for transmitting data is 100 Kbytes per second.

There are a number of parameters such as timeouts and holdoff periods which must be configured within the protocol. The values used in the simulator are shown in Table 1. Although we have attempted to choose reasonable values for these parameters and believe that our results are not particularly sensitive to these choices, we have not yet experimented with the effects of possible alternate choices.

The simulator does not attempt to model channel contention. Given a suitable data link layer protocol such as MACAW [2], the chance of data being lost to collision during periods of channel contention grows small, although at the expense of more delay in packet sending and receipt as each host waits for its turn to use the bandwidth. Therefore, while our model should accurately show the numbers of packets that will be received, we cannot completely evaluate the latency of any individual packet exchange.

The simulator also does not model possible one-way links between hosts. This choice is necessarily implied by our desire to implement the protocol on top of a data-link layer with link level acknowledgements. However, as described above, the protocol will still work in the absence of link layer acknowledgements.

A final minor limitation of our simulation is that our simulated environment is assumed to be devoid of obstacles to transmission or movement. Further, transmission failures are assumed to be uniformly distributed and independent, which does not take into account spatially localized failures due to sources such as microwave ovens, in the case of radio, or windows and reflections, in the case of infrared.

Table 1 Parameter values used in the simulation

Parameter	Value
Period between nonpropagating route requests	5 sec.
Nonpropagating route request time out	100 msec.
Route request time out	500 msec.
Route request slot length	500 msec.
Maximum route request period	10 sec.
Route reply holdoff per-hop delay	4 msec.

5.2. Results

We executed 20 runs of the simulator for each of a number of different movement rates and numbers of mobile hosts in the simulated ad hoc network. Each run simulated 4000 seconds of execution (just over one hour), with each mobile host moving and communicating as described in Section 5.1. The movement rate of the mobile hosts was determined by the pause time described above, with pause times ranging from 0 to 4000. With a pause time of 0, all hosts are constantly in motion, whereas with a pause time of 4000, hosts do not move during the run from their initial randomly chosen locations. The ad hoc network in each run consisted of either 6, 12, 18, or 24 mobile hosts. We present here the average over the 20 runs of the simulator for each of these cases; standard deviation for all cases was within 7% and in general was 2% or less of the average value for each case.

Figure 5 shows the total number of packet transmissions performed relative to the optimal number of transmissions for the data packets sent during the simulation. The optimal number of transmissions is the number of hops for each data packet needed to get from the sender of a packet to the intended receiver, if perfect routing decisions are made for each packet and if no transmission errors occur. The total number of packets actually transmitted includes the number of hops for each data packet based on the source route used by the sender, plus all packet transmissions used for route request, route reply, and route error packets. This ratio shows the work efficiency of the protocol: a value of 1.0 indicates a perfectly efficient protocol with no overhead packets present.

For all but the shortest pause times in the simulated environment, the total number of packets transmitted by the protocol is very close to optimal, and falls to an overhead of about 1% (a ratio of 1.01) for pause times greater than 1000 with 24 mobile hosts, as shown in Figure 5. For very short pause times, representing very frequent host movement, the protocol overhead is higher, reaching a maximum ratio of 2.6 for a pause time of 0, representing all hosts in constant motion. In such situations, source routes become invalid quickly after they are discovered, and additional overhead is spent discovering new routes. However, because the route maintenance procedure can quickly detect when a route in use is no longer working, nearly all data packets can be successfully delivered even in peri-

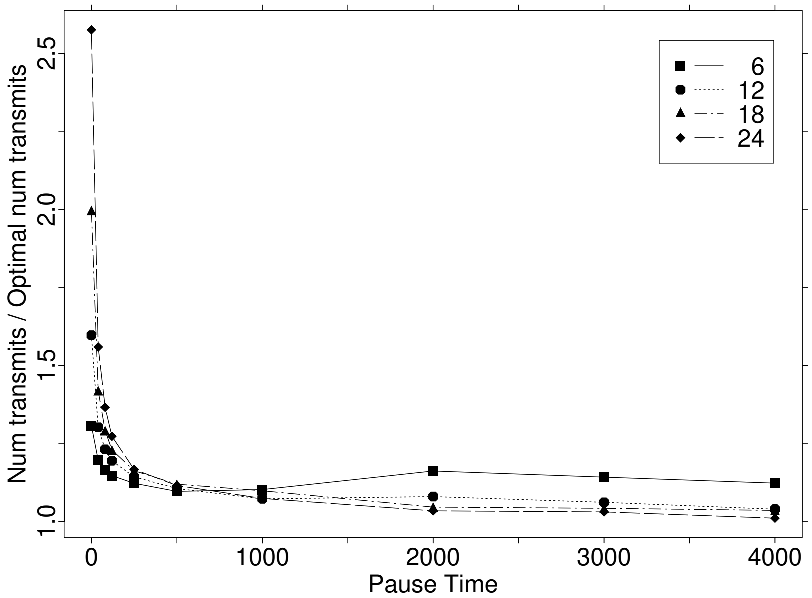

Figure 5 Average total number of transmissions performed relative to optimal (20 runs)

ods of such extreme host movement. Distance vector protocols, on the other hand, would be unable to keep up with the rate of change in the network and would be unable to deliver most data packets.

The performance results for protocol overhead presented here are affected by the occurrence of disconnected components of the mobile hosts within the area of the ad hoc network. When placing a number of mobile hosts at random locations within the simulation area, there is a chance that some groups of hosts will be unable to communicate with other groups of hosts since all hosts in each group are out of wireless transmission range of all hosts in other groups. This effectively forms a partition of the ad hoc network. For example, Figure 6 illustrates a typical placement of 24 mobile hosts, which happened to form two disconnected components. With fewer mobile hosts spread over the same area, we have observed similar (but worse) occurrences of disconnected components.

For each data packet sent with the receiver outside the sender's disconnected component, the basic protocol would initiate a route discovery, although we have included an optimization to limit the rate of new discoveries using an exponential backoff, as described in Section 4.4. The remaining extra route discoveries still performed in such situations show in increased protocol overhead, such as in the higher overhead values for 6 and 12 hosts shown in Figure 5, although the number of such extra route discoveries due disconnected components is greatly reduced by this optimization.

Figure 7 shows the average length of a source route used in sending a data packet relative to the optimal route length for the packets sent during the simulation. The optimal route length here is the number of hops needed to reach the destination if perfect routing information were available. In Figure 7, a different scale has been used on the y-axis than was used in Figure 5 in order to show the relevant detail in the shape of each graph. The ratio of the length of routes used relative to optimal shows the degree to which the protocol finds and maintains optimal routes as the mobile hosts move about. As shown here, the protocol finds and uses close to optimal routes, even without the route shortening optimization using promiscuous receive mode described in Section 4.3. The difference in length be-

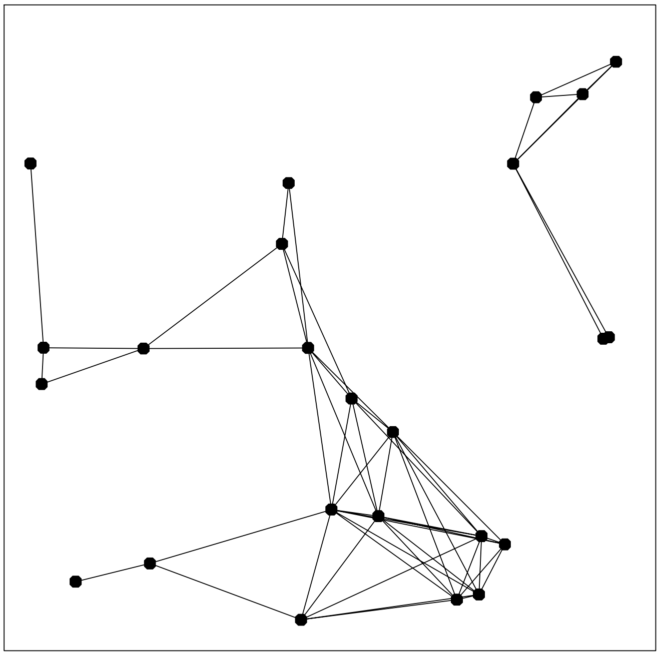

Figure 6 Example of disconnected clusters with 24 hosts

tween the routes used and the optimal route lengths is negligible. In most cases, the route lengths used are on average within a factor of 1.01 of optimal, and in all cases are within a factor of 1.09 of optimal.

6. Related Work

Research on packet routing in wireless networks of mobile hosts dates back at least to 1973 when the U.S. Defense Advanced Research Projects Agency began the DARPA Packet Radio Network (PRNET) project [11] and its successor the Survivable Adaptive Networks (SURAN) project [15]. PRNET supports the automatic set up and maintenance of packet switched communication routes in a network of moderately mobile hosts communicating via radios. The PRNET routing protocol uses a form of distance vector routing, with each node broadcasting a routing update packet (called a packet-radio organization packet, or PROP, in PRNET) every 7.5 seconds. The header of each data packet contains the source and destination node addresses, the number of hops taken so far from the source, and the number of hops remaining to reach the destination (based on the sender's routing table). Nodes promiscuously receive all packets and may update their routing tables based on this header information. The data link protocol uses hop-by-hop acknowledgements, using either explicit (active) acknowledgements or passive acknowledgements from these promiscuously received packets.

The amateur radio community has also worked extensively with routing in wireless networks of (sometimes) mobile hosts [14], holding 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 [7]. 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 [6, 8]. NET/ROM also allows updating of its routing table based on the source address information in the headers of packets that it receives.

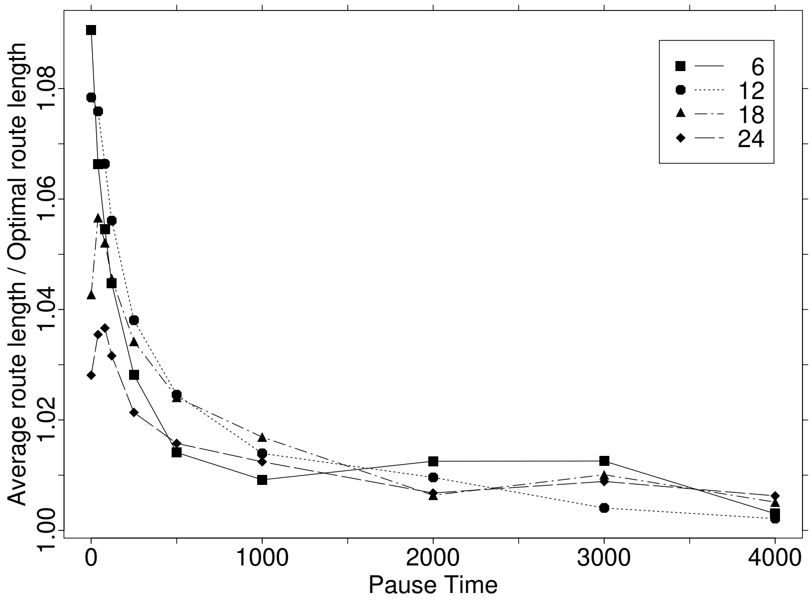

Figure 7 Average route length used relative to optimal (20 runs)

A particular problem with the use of distance vector routing protocols in networks with hosts that move, is the possibility of forming routing loops. In order to eliminate this possibility, Perkins and Bhagwat have recently proposed adding sequence numbers to routing updates in their Destination-Sequenced Distance Vector (DSDV) protocol [19]. These sequence numbers are used to compare the age of information in a routing update, and allow each node to preferentially select routes based on the freshest information. DSDV also uses triggered routing updates to speed route convergence. In order to damp route fluctuation and reduce congestion from large numbers of triggered updates after a route changes, each node in DSDV maintains information about the frequency with which it sees route changes and may delay some routing updates.

Our dynamic source routing protocol is similar in approach to some source routing protocols used in wired networks, such as in the IEEE 802 SRT bridge standard [20], in FLIP [12], and in SDRP [5, 28]. Our route request packet serves essentially the same role in route discovery as an "all paths explorer" packet. However, in wired networks, a bridge can copy an all paths explorer from one network interface onto each of its other interfaces and be sure that the explorer will flood the network in an orderly and complete way. Our protocol includes optimizations such as caching $\langle$initiator address, request id$\rangle$ pairs to efficiently flood explorers through a wireless network. We also make extensive use of caching and can effectively make use of promiscuous receive mode in the network interface to optimize route discovery.

The Internet Address Resolution Protocol (ARP) [21] is used to find the MAC address of a host on the same LAN as the sender. ARP is somewhat similar to our nonpropagating route request packets, except that a mobile host may answer the route request from its cache whereas ARP requests are normally only answered by the target host itself. In cases in which several LANs have been bridged together, the bridge may run "proxy" ARP [24], which allows the bridge to answer an ARP request on behalf of another host. In this sense, our nonpropagating route requests are also similar to proxy ARP in that they expand the effective size of a single host's route cache by allowing it to cheaply make use of the caches of neighboring hosts to reduce the need for propagating request packets.

One tradeoff between source routing and distance vector routing is in the handling of partioned networks, as mentioned in Section 5.2. Under dynamic source routing, if a host wishes to communicate with an unreachable host, the rate at which route requests are made will be reduced by a back off mechanism; the protocol, however, continues to make periodic efforts to find a route to the unreachable host, consuming some network resources. Under distance vector routing, with the assumption that routes have had time to converge once the host became unreachable, no network resources are spent trying to send packets to an unreachable host, as none of the hosts in the sender's partition of the network have a routing table entry for the destination. In practice, the problem of attempting to route packets to unreachable hosts is less significant, as connection-oriented protocols typically give up after a certain timeout or number of attempts, and human users will likewise give up eventually on any attempt to reach an unreachable host. In our simulation, however, a host continues to attempt sending packets to the destination host for the entire duration of each simulated conversation.

In general, when hosts move quickly enough and frequently enough, the best strategy that any routing protocol can use is to flood data packets throughout the network in hopes that the moving host will run into one of the many copies. On the other hand, when host movement is very slow or infrequent, ideally no overhead should be required for routing, since none of the routing information changes. Dynamic source routing moves easily between these two regimes, driven by the rate at which route requests are initiated. If a host with which another host wants to communicate is moving very quickly, source routing floods the network with a route request and then sends a data packet as soon as the destination is found and a route reply returned. Hosts uninterested in talking to this quickly moving host may see some of the available bandwidth consumed by the route requests but will have their own routes left unaffected. Likewise, if host motion slows, the rate at which routes cease working will slow and routing overhead will thus be reduced due to less frequent need for new route discoveries.

7. Conclusion

This paper has presented a protocol for routing packets between wireless mobile hosts in an ad hoc network. Unlike routing protocols using distance vector or link state algorithms, our protocol uses dynamic source routing which adapts quickly to routing changes when host movement is frequent, yet requires little or no overhead during periods in which hosts move less frequently. Based on results from a packet-level simulation of mobile hosts operating in an ad hoc network, the protocol performs well over a variety of environmental conditions such as host density and movement rates. For all but the highest rates of host movement simulated, the overhead of the protocol is quite low, falling to just 1% of total data packets transmitted for moderate movement rates in a network of 24 mobile hosts. In all cases, the difference in length between the routes used and the optimal route lengths is negligible, and in most cases, route lengths are on average within a factor of 1.02 of optimal.

We are currently expanding our simulations to incorporate some additional optimizations and to quantify the effects of the individual optimizations on the behavior and performance of the protocol. We are also continuing to study other routing protocols for use in ad hoc networks, including those based on distance vector or link state routing, as well as the interconnection of an ad hoc network with a wide-area network such as the Internet, reachable by some but not all of the ad hoc network nodes. Although this paper does not address the security concerns inherent in wireless networks or packet routing, we are currently examining these issues with respect to attacks on privacy and denial of service in the routing protocol. Finally, we are beginning implementation of the protocol on notebook computers for use by students in an academic environment.

Acknowledgements

This research was supported in part by the Wireless Initiative of the Information Networking Institute at Carnegie Mellon University, and by the National Science Foundation under CAREER Award NCR-9502725. David Maltz was also supported in part by an IBM Cooperative Fellowship.

References

[1] David F. Bantz and Frédéric J. Bauchot. Wireless LAN design alternatives. *IEEE Network*, 8(2):43–53, March/April 1994.

[2] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless LAN's. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 212–225, August 1994.

[3] Robert T. Braden, editor. Requirements for Internet hosts — communication layers. Internet Request For Comments RFC 1122, October 1989.

[4] Roy C. Dixon and Daniel A. Pitt. Addressing, bridging, and source routing. *IEEE Network*, 2(1):25–32, January 1988.

[5] Deborah Estrin, Daniel Zappala, Tony Li, Yakov Rekhter, and Kannan Varadhan. Source Demand Routing: Packet format and forwarding specification (version 1). Internet Draft, January 1995. Work in progress.

[6] Daniel M. Frank. Transmission of IP datagrams over NET/ROM networks. In *ARRL Amateur Radio 7th Computer Networking Conference*, pages 65–70, October 1988.

[7] Bdale Garbee. Thoughts on the issues of address resolution and routing in amateur packet radio TCP/IP networks. In *ARRL Amateur Radio 6th Computer Networking Conference*, pages 56–58, August 1987.

[8] James Geier, Martin DeSimio, and Byron Welsh. Network routing techniques and their relevance to packet radio networks. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 105–117, September 1990.

[9] C. Hedrick. Routing Information Protocol. Internet Request For Comments RFC 1058, June 1988.

[10] International Standards Organization. Intermediate system to intermediate system intra-domain routing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473). ISO DP 10589, February 1990.

[11] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.

[12] M. Frans Kaashoek, Robbert van Renesse, Hans van Staveren, and Andrew S. Tanenbaum. FLIP: An internetwork protocol for supporting distributed systems. *ACM Transactions on Computer Systems*, 11(1):73–106, February 1993.

[13] Phil Karn. MACA — A new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–140, September 1990.

[14] Philip R. Karn, Harold E. Price, and Robert J. Diersing. Packet radio in the amateur service. *IEEE Journal on Selected Areas in Communications*, SAC-3(3):431–439, May 1985.

[15] Gregory S. Lauer. Packet-radio routing. In *Routing in Communications Networks*, edited by Martha E. SteenStrup, chapter 11, pages 55–76. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[16] John M. McQuillan, Ira Richer, and Eric C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[17] John M. McQuillan and David C. Walden. The ARPA network design decisions. *Computer Networks*, 1(5):243–289, August 1977.

[18] J. Moy. OSPF version 2. Internet Request For Comments RFC 1247, July 1991.

[19] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[20] Radia Perlman. *Interconnections: Bridges and Routers*. Addison-Wesley, Reading, Massachusetts, 1992.

[21] David C. Plummer. An Ethernet address resolution protocol: Or converting network protocol addresses to 48.bit Ethernet addresses for transmission on Ethernet hardware. Internet Request For Comments RFC 826, November 1982.

[22] J. B. Postel, editor. Internet Protocol. Internet Request For Comments RFC 791, September 1981.

[23] J. B. Postel, editor. Transmission Control Protocol. Internet Request For Comments RFC 793, September 1981.

[24] J. B. Postel. Multi-LAN address resolution. Internet Request For Comments RFC 925, October 1984.

[25] Nachum Shacham and Jil Westcott. Future directions in packet radio architectures and protocols. *Proceedings of the IEEE*, 75(1):83–99, January 1987.

[26] Gursharan S. Sidhu, Richard F. Andrews, and Alan B. Oppenheimer. *Inside AppleTalk*. Addison Wesley, Reading, Massachusetts, 1990.

[27] Paul Turner. NetWare communications processes. *NetWare Application Notes*, Novell Research, pages 25–81, September 1990.

[28] Kannan Varadhan, Deborah Estrin, Steve Hotz, and Yakov Rekhter. SDRP route construction. Internet Draft, July 1994. Work in progress.

[29] Xerox Corporation. Internet transport protocols. Xerox System Integration Standard 028112, December 1981.