A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols

Josh Broch David A. Maltz David B. Johnson Yih-Chun Hu Jorjeta Jetcheva

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

http://www.monarch.cs.cmu.edu/

Abstract

An ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. Due to the limited transmission range of wireless network interfaces, multiple network "hops" may be needed for one node to exchange data with another across the network. In recent years, a variety of new routing protocols targeted specifically at this environment have been developed, but little performance information on each protocol and no realistic performance comparison between them is available. This paper presents the results of a detailed packet-level simulation comparing four multi-hop wireless ad hoc network routing protocols that cover a range of design choices: DSDV, TORA, DSR, and AODV. We have extended the *ns*-2 network simulator to accurately model the MAC and physical-layer behavior of the IEEE 802.11 wireless LAN standard, including a realistic wireless transmission channel model, and present the results of simulations of networks of 50 mobile nodes.

1 Introduction

In areas in which there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use, wireless mobile users may still be able to communicate through the formation of an *ad hoc network*. In such a network, each mobile node operates not only as a host but also as a router, forwarding packets for other mobile nodes in the network that may not be within direct wireless transmission range of each other. Each node participates in an ad hoc routing protocol that allows it to discover "multi-hop" paths through the network to any other node. The idea of ad hoc networking is sometimes also called *infrastructureless networking* [13], since the mobile nodes in the network dynamically establish routing among themselves to form their own network "on the fly." Some examples of the possible uses of ad hoc networking include students using laptop computers to participate in an interactive lecture, business associates sharing information during a meeting, soldiers relaying information for situational awareness on the battlefield [12, 21], and emergency disaster relief personnel coordinating efforts after a hurricane or earthquake.

This work was supported in part by the National Science Foundation (NSF) under CAREER Award NCR-9502725, by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061, and by the AT&T Foundation under a Special Purpose Grant in Science and Engineering. David Maltz was also supported under an IBM Cooperative Fellowship, and Yih-Chun Hu was also supported by an NSF Graduate 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 NSF, AFMC, DARPA, the AT&T Foundation, IBM, Carnegie Mellon University, or the U.S. Government.

To appear in *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'98), October 25–30, 1998, Dallas, Texas, USA.

Many different protocols have been proposed to solve the multi-hop routing problem in ad hoc networks, each based on different assumptions and intuitions. However, little is known about the actual performance of these protocols, and no attempt has previously been made to directly compare them in a realistic manner.

This paper is the first to provide a realistic, quantitative analysis comparing the performance of a variety of multi-hop wireless ad hoc network routing protocols. We present results of detailed simulations showing the relative performance of four recently proposed ad hoc routing protocols: DSDV [18], TORA [14, 15], DSR [9, 10, 2], and AODV [17]. To enable these simulations, we extended the *ns*-2 network simulator [6] to include:

- *Node mobility*.
- A *realistic physical layer* including a radio propagation model supporting propagation delay, capture effects, and carrier sense [20].
- *Radio network interfaces* with properties such as transmission power, antenna gain, and receiver sensitivity.
- The *IEEE 802.11 Medium Access Control (MAC) protocol* using the Distributed Coordination Function (DCF) [8].

Our results in this paper are based on simulations of an ad hoc network of 50 wireless mobile nodes moving about and communicating with each other. We analyze the performance of each protocol and explain the design choices that account for their performance.

2 Simulation Environment

ns is a discrete event simulator developed by the University of California at Berkeley and the VINT project [6]. While it provides substantial support for simulating TCP and other protocols over conventional networks, it provides no support for accurately simulating the physical aspects of multi-hop wireless networks or the MAC protocols needed in such environments. Berkeley has recently released *ns* code that provides some support for modeling wireless LANs, but this code cannot be used for studying multi-hop ad hoc networks as it does not support the notion of node position; there is no spatial diversity (all nodes are in the same collision domain), and it can only model directly connected nodes.

In this section, we describe some of the modifications we made to *ns* to allow accurate simulation of mobile wireless networks.

2.1 Physical and Data Link Layer Model

To accurately model the attenuation of radio waves between antennas close to the ground, radio engineers typically use a model that attenuates the power of a signal as $1/r^2$ at short distances (r is the distance between the antennas), and as $1/r^4$ at longer distances. The crossover point is called the *reference distance*, and is typically around 100 meters for outdoor low-gain antennas 1.5m above the ground plane operating in the 1–2GHz band [20]. Following this practice, our signal propagation model combines both a free space propagation model and a two-ray ground reflection model. When a transmitter is within the reference distance of the receiver, we use the free space model where the signal attenuates as $1/r^2$. Outside of this distance, we use the ground reflection model where the signal falls off as $1/r^4$.

Each mobile node has a position and a velocity and moves around on a topography that is specified using either a digital elevation map or a flat grid. The position of a mobile node can be calculated as a function of time, and is used by the radio propagation model to calculate the propagation delay from one node to another and to determine the power level of a received signal at each mobile node.

Each mobile node has one or more wireless network interfaces, with all interfaces of the same type (on all mobile nodes) linked together by a single physical channel. When a network interface transmits a packet, it passes the packet to the appropriate physical channel object. This object then computes the propagation delay from the sender to every other interface on the channel and schedules a "packet reception" event for each. This event notifies the receiving interface that the first bit of a new packet has arrived. At this time, the power level at which the packet was received is compared to two different values: the carrier sense threshold and the receive threshold. If the power level falls below the carrier sense threshold, the packet is discarded as noise. If the received power level is above the carrier sense threshold but below the receive threshold, the packet is marked as a packet in error before being passed to the MAC layer. Otherwise, the packet is simply handed up to the MAC layer.

Once the MAC layer receives a packet, it checks to insure that its receive state is presently "idle." If the receiver is not idle, one of two things can happen. If the power level of the packet already being received is at least 10 dB greater than the received power level of the new packet, we assume capture, discard the new packet, and allow the receiving interface to continue with its current receive operation. Otherwise, a collision occurs and both packets are dropped.

If the MAC layer is idle when an incoming packet is passed up from the network interface, it simply computes the transmission time of the packet and schedules a "packet reception complete" event for itself. When this event occurs, the MAC layer verifies that the packet is error-free, performs destination address filtering, and passes the packet up the protocol stack.

2.2 Medium Access Control

The link layer of our simulator implements the complete IEEE 802.11 standard [8] Medium Access Control (MAC) protocol Distributed Coordination Function (DCF) in order to accurately model the contention of nodes for the wireless medium. DCF is similar to MACA [11] and MACAW [1] and is designed to use both *physical carrier sense* and *virtual carrier sense* mechanisms to reduce the probability of collisions due to hidden terminals. The transmission of each unicast packet is preceded by a Request-to-Send/Clear-to-Send (RTS/CTS) exchange that reserves the wireless channel for transmission of a data packet. Each correctly received unicast packet is followed by an Acknowledgment (ACK) to the sender, which retransmits the packet a limited number of times until this ACK is received. Broadcast packets are sent only when virtual and physical carrier sense indicate that the medium is clear, but they are not preceded by an RTS/CTS and are not acknowledged by their recipients.

2.3 Address Resolution

Since the routing protocols all operate at the network layer using IP addresses, an implementation of ARP [19], modeled after the BSD Unix implementation [23], was included in the simulation and used to resolve IP addresses to link layer addresses. The broadcast nature of an ARP REQUEST packet (Section 6.3) and the interaction of ARP with on-demand protocols (Section 6.4) make ARP an important detail of the simulation.

2.4 Packet Buffering

Each node has a queue for packets awaiting transmission by the network interface that holds up to 50 packets and is managed in a drop-tail fashion. Each on-demand routing protocol (i.e., TORA, DSR, or AODV), can buffer separately an additional 50 packets that that are awaiting discovery of a route through the network.

3 Ad Hoc Network Routing Protocols Studied

In this section, we briefly describe the key features of the DSDV, TORA, DSR, and AODV protocols studied in our simulations. We also describe the particular parameters that we chose when implementing each protocol.

The protocols were carefully implemented according to their specifications published as of April 1998 and based on clarifications of some issues from the designers of each protocol and on our own experimentation with them. In particular, during the process of implementing each protocol and analyzing the results from early simulation runs, we discovered some modifications for each protocol that improved its performance. The key improvements to each protocol are highlighted in the respective protocol descriptions below. We also made the following improvements to all of the protocols:

- To prevent synchronization, periodic broadcasts and packets sent in response to the reception of a broadcast packet were jittered using a random delay uniformly distributed between 0 and 10 milliseconds.
- To insure that routing information propagated through the network in a timely fashion, routing packets being sent were queued for transmission at the head of the network interface transmit queue, whereas all other packets (ARP and data) were inserted at the end of the interface transmit queue.
- Each of the protocols used link breakage detection feedback from the 802.11 MAC layer that indicated when a packet could not be forwarded to its next hop, with the exception of DSDV as explained in Section 3.1.2.

3.1 Destination-Sequenced Distance Vector (DSDV)

DSDV [18] is a hop-by-hop distance vector routing protocol requiring each node to periodically broadcast routing updates. The key advantage of DSDV over traditional distance vector protocols is that it guarantees loop-freedom.

3.1.1 Basic Mechanisms

Each DSDV node maintains a routing table listing the "next hop" for each reachable destination. DSDV tags each route with a sequence number and considers a route R more favorable than R' if R has a greater sequence number, or if the two routes have equal sequence numbers but R has a lower metric. Each node in the network advertises a monotonically increasing even sequence number for itself. When a node **B** decides that its route to a destination **D** has broken, it advertises the route to **D** with an infinite metric and a sequence number one greater than its sequence number for the route that has broken (making an odd sequence number). This causes any node **A** routing packets through **B** to incorporate the infinite-metric route into its routing table until node **A** hears a route to **D** with a higher sequence number.

3.1.2 Implementation Decisions

We did not use link layer breakage detection from the 802.11 MAC protocol in obtaining the DSDV data presented in this paper, because after implementing the protocol both with and without it, we found the performance significantly worse *with* the link layer breakage detection. The reason is that if a neighbor **N** of a node **A** detects that its link to **A** is broken, it will broadcast a triggered route update containing an infinite metric for **A**. The sequence number in this triggered update will be one greater than the last sequence number broadcast by **A**, and therefore is the highest sequence number existing anywhere in the network for **A**. Each node that hears this update will record an infinite metric for destination **A** and will propagate the

Table I Constants used in the DSDV-SQ simulation.

Periodic route update interval	15 s
Periodic updates missed before link declared broken	3
Initial triggered update weighted settling time	6 s
Weighted settling time weighting factor	7/8
Route advertisement aggregation time	1 s
Maximum packets buffered per node per destination	5

information further. This renders node **A** unreachable from all nodes in the network until **A** broadcasts a newer sequence number in a periodic update. **A** will send this update as soon as it learns of the infinite metric being propagated for it, but large numbers of packets can be dropped in the meantime.

Our implementation uses both full and incremental updates as required by the protocol's description. However, the published description of DSDV [18] is ambiguous about specifying when triggered updates should be sent. One interpretation is that the receipt of a new sequence number for a destination should cause a triggered update. We call this approach DSDV-SQ (sequence number). The advantage of this approach is that broken links will be detected and routed around as new sequence numbers propagate around the broken link and create alternate routes. The second interpretation, which we call simply DSDV, is that only the receipt of a new metric should cause a triggered update, and that the receipt of a new sequence number is not sufficiently important to incur the overhead of propagating a triggered update.

We implemented both DSDV-SQ and DSDV and found that while DSDV-SQ is much more expensive in terms of overhead, it provides a much better packet delivery ratio in most cases. The second scheme (DSDV) is much more conservative in terms of routing overhead, but because link breakages are not detected as quickly, more data packets are dropped. All of the results in this paper use DSDV-SQ, with the exception of Section 6.2, which compares this with DSDV.

Table I lists the constants used in our DSDV-SQ simulation.

3.2 Temporally-Ordered Routing Algorithm (TORA)

TORA [14, 15] is a distributed routing protocol based on a "link reversal" algorithm. It is designed to discover routes on demand, provide multiple routes to a destination, establish routes quickly, and minimize communication overhead by localizing algorithmic reaction to topological changes when possible. Route optimality (shortest-path routing) is considered of secondary importance, and longer routes are often used to avoid the overhead of discovering newer routes.

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

3.2.1 Basic Mechanisms

At each node in the network, a logically separate copy of TORA is run for each destination. When a node needs a route to a particular destination, it broadcasts a QUERY packet containing the address of the destination for which it requires a route. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination. The recipient of the QUERY then broadcasts an UPDATE packet listing its height with respect to the destination. As this packet propagates through the network, each node that receives the UPDATE sets its height to a value greater than the height of the neighbor from which the UPDATE was received. This has the effect of creating a series of directed links from the original sender of the QUERY to the node that initially generated the UPDATE.

When a node discovers that a route to a destination is no longer valid, it adjusts its height so that it is a local maximum with respect to its neighbors and transmits an UPDATE packet. If the node has no neighbors of finite height with respect to this destination, then the node instead attempts to discover a new route as described above. When a node detects a network partition, it generates a CLEAR packet that resets routing state and removes invalid routes from the network.

TORA is layered on top of IMEP, the Internet MANET Encapsulation Protocol [5], which is required to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, plus notification to the routing protocol whenever a link to one of its neighbors is created or broken. To reduce overhead, IMEP attempts to aggregate many TORA and IMEP control messages (which IMEP refers to as *objects*) together into a single packet (as an *object block*) before transmission. Each block carries a sequence number and a response list of other nodes from which an ACK has not yet been received, and only those nodes ACK the block when receiving it; IMEP retransmits each block with some period, and continues to retransmit it if needed for some maximum total period, after which time, the link to each unacknowledged node is declared down and TORA is notified. IMEP can also provide network layer address resolution, but we did not use this service, as we used ARP [19] with all four routing protocols. For link status sensing and maintaining a list of a node's neighbors, each IMEP node periodically transmits a BEACON (or "BEACON-equivalent") packet, which is answered by each node hearing it with a HELLO (or "HELLO-equivalent") packet.

3.2.2 Implementation Decisions

IMEP must queue objects for some period of time to allow possible aggregation with other objects, but the IMEP specification [5] does not define this time period, and we discovered that the overall performance of the protocol was very sensitive to the choice of this value. After significant experimentation, we chose as the best balance between packet overhead and routing protocol convergence, to aggregate HELLO and ACK packets for a time uniformly chosen between 150 ms and 250 ms, and to not delay TORA routing messages for aggregation. The reason for not delaying these messages is that the TORA link reversal process creates short-lived routing loops that exist from the time that the link-reversal starts until the time that all nodes that need to be aware of the reversal receive the corresponding UPDATE (Section 5.2). Delaying the transmission of TORA routing messages for aggregation, coupled with any queuing delay at the network interface, allows these routing loops to last long enough that significant numbers of data packets are dropped.

The TORA and IMEP specifications [15, 5] do not define the precise semantics of reliable object delivery required by TORA, but experimentation showed that very strong semantics must be provided in order to prevent the creation of long-lived routing loops. In particular, all TORA objects must be delivered reliably and in order, without any duplication. Additionally, all neighboring nodes in the ad hoc network must have a consistent picture of the network with regard to each destination. This implies that anytime a node **A** decides its link to a neighbor **B** has gone down, **B** *must* also decide that the link to **A** has gone down.

We have implemented IMEP to provide this functionality, although the retransmission timeout and total number of attempts are not specified by IMEP [5]. We chose a retransmission period of 500 ms and a total timeout of 1500 ms, although based upon our observations, an adaptive retransmission timer should be added to the protocol. In-order delivery is enforced by, at each receiver node **B**, only passing

Table II Constants used in the TORA simulation.

BEACON period	1 s
Time after which a link is declared down if no BEACON or HELLO packets were exchanged	3 s
Time after which an object block is retransmitted if no acknowledgment is received	500 ms
Time after which an object block is not retransmitted and the link to the destination is declared down	1500 ms
Min HELLO and ACK aggregation delay	150 ms
Max HELLO and ACK aggregation delay	250 ms

an object block from some node **A** to TORA if the block has the sequence number that IMEP at **B** next expects from **A**. Blocks with lower sequence numbers may generate another ACK but are otherwise dropped. Blocks with higher sequence numbers are queued until the missing blocks arrive or until the maximum 1500 ms total timeout expires, at which point **B** can be certain the object will never be retransmitted. By this point, **A** will have declared its link to **B** down, since it will not have received an ACK for the missing packet. To give the routing protocol at **B** a picture consistent with that seen by the protocol at **A**, the IMEP layer at **B** notifies its routing protocol that the link to **A** is down, then notifies it the link is back up, and then processes the queued packets.

Finally, we improved IMEP's method of link status sensing by reducing it to a point that functions with minimum overhead yet still maintains all of the required link status information. During our experimentation with IMEP, we found that nodes need only send BEACON messages when they are disconnected from all other nodes. Suppose two nodes **A** and **B**, both of which have neighbors, transmit a single HELLO listing each of their neighbors once per BEACON period. If a bi-directional link exists between **A** and **B**, both nodes will overhear each other's HELLOs and all other transmissions, causing each node to create a link to the other with "incoming" status. In subsequent HELLO messages, **A** and **B** will list each other, confirming that a bi-directional link exists between them.

Table II lists the constants used in our TORA simulation.

3.3 Dynamic Source Routing (DSR)

DSR [9, 10, 2] uses source routing rather than hop-by-hop routing, with each packet to be routed carrying in its header the complete, ordered list of nodes through which the packet must pass. The key advantage of source routing is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, coupled with the on-demand nature of the protocol, eliminates the need for the periodic route advertisement and neighbor detection packets present in other protocols.

3.3.1 Basic Mechanisms

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism by which a node **S** wishing to send a packet to a destination **D** obtains a source route to **D**. To perform a Route Discovery, the source node **S** broadcasts a ROUTE REQUEST packet that is flooded through the network in a controlled manner and is answered by a ROUTE REPLY packet from either the destination node or another node that knows a route to the destination. To reduce the cost of Route Discovery, each node maintains a cache of source routes it has learned or overheard, which it aggressively uses to limit the frequency and propagation of ROUTE REQUESTs.

Route Maintenance is the mechanism by which a packet's sender **S** detects if the network topology has changed such that it can no longer use its route to the destination **D** because two nodes listed in the route have moved out of range of each other. When Route Maintenance indicates a source route is broken, **S** is notified with a ROUTE ERROR packet. The sender **S** can then attempt to use any other route to **D** already in its cache or can invoke Route Discovery again to find a new route.

Table III Constants used in the DSR simulation.

Time between retransmitted ROUTE REQUESTs (exponentially backed off)	500 ms
Size of source route header carrying n addresses	$4n + 4$ bytes
Timeout for nonpropagating search	30 ms
Time to hold packets awaiting routes	30 s
Max rate for sending gratuitous REPLYs for a route	1/s

3.3.2 Implementation Decisions

Using the suggestions from the published description of DSR [10], we have optimized our implementation in a number of ways.

Although the DSR protocol supports unidirectional routes, IEEE 802.11 requires an RTS/CTS/Data/ACK exchange for all unicast packets, limiting the routing protocol to using only bidirectional links in delivering data packets. We implemented DSR to discover only routes composed of bidirectional links by requiring that a node return all ROUTE REPLY messages to the requestor by reversing the path over which the ROUTE REQUEST packet came. If the path taken by a ROUTE REQUEST contained unidirectional links, then the corresponding ROUTE REPLY will not reach the requestor, preventing it from learning the unidirectional link route.

In Route Discovery, a node first sends a ROUTE REQUEST with the maximum propagation limit (hop limit) set to zero, prohibiting its neighbors from rebroadcasting it. At the cost of a single broadcast packet, this mechanism allows a node to query the route caches of all its neighbors for a route and also optimizes the case in which the destination node is adjacent to the source. If this nonpropagating search times out, a propagating ROUTE REQUEST is sent.

Nodes operate their network interfaces in *promiscuous* mode, disabling the interface's address filtering and causing the network protocol to receive all packets that the interface overhears. These packets are scanned for useful source routes or ROUTE ERROR messages and then discarded. This optimization allows nodes to learn potentially useful information, while causing no additional overhead on the limited network bandwidth.

Furthermore, when a node overhears a packet not addressed to itself, it checks the unprocessed portion of the source route in the packet's header. If the node's own address is present, it knows that this source route could bypass the unprocessed hops preceding it in the route. The node then sends a gratuitous ROUTE REPLY message to the packet's source, giving it the shorter route without these hops.

Finally, when an intermediate node forwarding a packet discovers that the next hop in the source route is unreachable, it examines its route cache for another route to the destination. If a route exists, the node replaces the broken source route on the packet with the route from its cache and retransmits the packet. If a route does not exist in its cache, the node drops the packet and does not begin a new Route Discovery of its own.

Table III lists the constants used in our DSR simulation.

3.4 Ad Hoc On-Demand Distance Vector (AODV)

AODV [17] is essentially a combination of both DSR and DSDV. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV.

3.4.1 Basic Mechanisms

When a node **S** needs a route to some destination **D**, it broadcasts a ROUTE REQUEST message to its neighbors, including the last known sequence number for that destination. The ROUTE REQUEST is flooded in a controlled manner through the network until it reaches a node that has a route to the destination. Each node that forwards the ROUTE REQUEST creates a *reverse route* for itself back to node **S**.

When the ROUTE REQUEST reaches a node with a route to **D**, that node generates a ROUTE REPLY that contains the number of hops necessary to reach **D** and the sequence number for **D** most recently seen by the node generating the REPLY. Each node that participates in forwarding this REPLY back toward the originator of the ROUTE REQUEST (node **S**), creates a *forward route* to **D**. The state created in each node along the path from **S** to **D** is hop-by-hop state; that is, each node remembers only the next hop and not the entire route, as would be done in source routing.

In order to maintain routes, AODV normally requires that each node periodically transmit a HELLO message, with a default rate of once per second. Failure to receive three consecutive HELLO messages from a neighbor is taken as an indication that the link to the neighbor in question is down. Alternatively, the AODV specification briefly suggests that a node may use physical layer or link layer methods to detect link breakages to nodes that it considers neighbors [17].

When a link goes down, any upstream node that has recently forwarded packets to a destination using that link is notified via an UNSOLICITED ROUTE REPLY containing an infinite metric for that destination. Upon receipt of such a ROUTE REPLY, a node must acquire a new route to the destination using Route Discovery as described above.

3.4.2 Implementation Decisions

We initially implemented AODV using periodic HELLO messages for link breakage detection as described in the AODV specification [17]. For comparison, we also implemented a version of AODV that we call AODV-LL (link layer), instead using *only* link layer feedback from 802.11 as in DSR, completely eliminating the standard AODV HELLO mechanism. Such an approach saves the overhead of the periodic HELLO messages, but does somewhat change the fundamental nature of the protocol; for example, all link breakage detection in AODV-LL is only on-demand, and thus a broken link cannot be detected until a packet needs to be sent over the link, whereas the periodic HELLO messages in standard AODV may allow broken links to be detected before a packet must be forwarded. Nevertheless, we found our alternate version AODV-LL to perform significantly better than standard AODV, and so we report measurements from that version here.

In addition, we also changed our AODV implementation to use a shorter timeout of 6 seconds before retrying a ROUTE REQUEST for which no ROUTE REPLY has been received (RREP_WAIT_TIME). The value given in the AODV specification was 120 seconds, based on the other constants specified there for AODV. However, a ROUTE REPLY can only be returned if each node along the discovered route still has a reverse route along which to return it, saved from when the ROUTE REQUEST was propagated. Since the specified timeout for this reverse route information in each node is only 3 seconds, the original ROUTE REPLY timeout value of 120 seconds unnecessarily limited the protocol's ability to recover from a dropped ROUTE REQUEST or ROUTE REPLY packet.

Table IV lists the constants used in our AODV-LL simulation.

Table IV Constants used in the AODV-LL simulation.

Time for which a route is considered active	300 s
Lifetime on a ROUTE REPLY sent by destination node	600 s
Number of times a ROUTE REQUEST is retried	3
Time before a ROUTE REQUEST is retried	6 s
Time for which the broadcast id for a forwarded ROUTE REQUEST is kept	3 s
Time for which reverse route information for a ROUTE REPLY is kept	3 s
Time before broken link is deleted from routing table	3 s
MAC layer link breakage detection	yes

4 Methodology

The overall goal of our experiments was to measure the ability of the routing protocols to react to network topology change while continuing to successfully deliver data packets to their destinations. To measure this ability, our basic methodology was to apply to a simulated network a variety of workloads, in effect testing with each data packet originated by some sender whether the routing protocol can at that time route to the destination of that packet. We were not attempting to measure the protocols' performance on a particular workload taken from real life, but rather to measure the protocols' performance under a range of conditions.

Our protocol evaluations are based on the simulation of 50 wireless nodes forming an ad hoc network, moving about over a rectangular (1500m $\times$ 300m) flat space for 900 seconds of simulated time. We chose a rectangular space in order to force the use of longer routes between nodes than would occur in a square space with equal node density. The physical radio characteristics of each mobile node's network interface, such as the antenna gain, transmit power, and receiver sensitivity, were chosen to approximate the Lucent WaveLAN [22] direct sequence spread spectrum radio.

In order to enable direct, fair comparisons between the protocols, it was critical to challenge the protocols with identical loads and environmental conditions. Each run of the simulator accepts as input a *scenario file* that describes the exact motion of each node and the exact sequence of packets originated by each node, together with the exact time at which each change in motion or packet origination is to occur. We pre-generated 210 different scenario files with varying movement patterns and traffic loads, and then ran all four routing protocols against each of these scenario files. Since each protocol was challenged in an identical fashion, we can directly compare the performance results of the four protocols.

4.1 Movement Model

Nodes in the simulation move according to a model that we call the "random waypoint" model [10]. The movement scenario files we used for each simulation are characterized by a *pause time*. Each node begins the simulation by remaining stationary for *pause time* seconds. It then selects a random destination in the 1500m $\times$ 300m space and moves to that destination at a speed distributed uniformly between 0 and some maximum speed. Upon reaching the destination, the node pauses again for *pause time* seconds, selects another destination, and proceeds there as previously described, repeating this behavior for the duration of the simulation. Each simulation ran for 900 seconds of simulated time.

We ran our simulations with movement patterns generated for 7 different pause times: 0, 30, 60, 120, 300, 600, and 900 seconds. A pause time of 0 seconds corresponds to continuous motion, and a pause time of 900 (the length of the simulation) corresponds to no motion.

Because the performance of the protocols is very sensitive to movement pattern, we generated scenario files with 70 different movement patterns, 10 for each value of pause time. All four routing protocols were run on the same 70 movement patterns.

We experimented with two different maximum speeds of node movement. We primarily report in this paper data from simulations using a maximum node speed of 20 meters per second (average speed 10 meters per second), but also compare this to simulations using a maximum speed of 1 meter per second.

4.2 Communication Model

As the goal of our simulation was to compare the performance of each routing protocol, we chose our traffic sources to be constant bit rate (CBR) sources. When defining the parameters of the communication model, we experimented with sending rates of 1, 4, and 8 packets

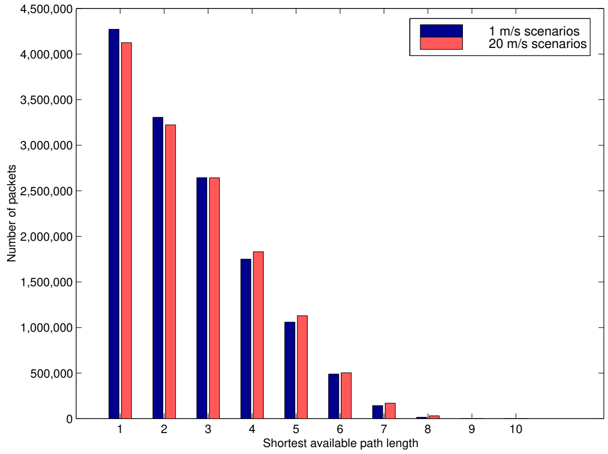


Figure 1 Distribution of the shortest path available to each application packet originated over all scenarios.

per second, networks containing 10, 20, and 30 CBR sources, and packet sizes of 64 and 1024 bytes.

Varying the number of CBR sources was approximately equivalent to varying the sending rate. Hence, for these simulations we chose to fix the sending rate at 4 packets per second, and used three different communication patterns corresponding to 10, 20, and 30 sources.

When using 1024-byte packets, we found that congestion, due to lack of spatial diversity, became a problem for all protocols and one or two nodes would drop most of the packets that they received for forwarding. As none of the studied protocols performs load balancing, and the goal of our analysis was to determine if the routing protocols could consistently track changes in topology, we attempted to factor out congestive effects by reducing the packet size to 64 bytes. This smaller packet size still provides a good test of the routing protocols, since we are still testing their ability to determine routes to a destination with the same frequency (a total of 40, 80, or 120 times per second).

All communication patterns were peer-to-peer, and connections were started at times uniformly distributed between 0 and 180 seconds. The three communication patterns (10, 20, and 30 sources), taken in conjunction with the 70 movement patterns, provide a total of 210 different scenario files for each maximum node movement speed (1 m/s and 20 m/s) with which we compared the four routing protocols.

We did not use TCP sources because TCP offers a conforming load to the network, meaning that it changes the times at which it sends packets based on its perception of the network's ability to carry packets. As a result, both the time at which each data packet is originated by its sender and the position of the node when sending the packet would differ between the protocols, preventing a direct comparison between them.

4.3 Scenario Characteristics

To characterize the challenge our scenarios placed on the routing protocols, we measured the lengths of the routes over which the protocols had to deliver packets, and the total number of topology changes in each scenario.

When each data packet is originated, an internal mechanism of our simulator (separate from the routing protocols) calculates the shortest path between the packet's sender and its destination. The packet is labeled with this information, which is compared with the number of hops actually taken by the packet when received by the intended destination. The shortest path is calculated based on a nominal transmission range of 250m for each radio and does not account for congestion or interference that any particular packet might see.

Table V Average number of link connectivity changes during each 900-second simulation as a function of pause time.

Pause Time	# of Connectivity Changes	
	1 m/s	20 m/s
0	898	11857
30	908	8984
60	792	7738
120	732	5390
300	512	2428
600	245	1270
900	0	0

Figure 1 shows the distribution of shortest path lengths for all packets over the 210 scenario files at 1 m/s and 20 m/s that we used. The height of each bar represents the number of packets for which the destination was the given distance away when the packet was originated. The average data packet in our simulations had to cross 2.6 hops to reach its destination, and the farthest reachable node to which the routing protocols had to deliver a packet was 8 hops away.

Table V shows the average number of link connectivity changes that occurred during each of the simulations runs for each value of pause time. We count one link connectivity change whenever a node goes into or out of direct communication range with another node. For the specific scenarios we used, the 30-second pause time scenarios at 1 m/s actually have a slightly higher average rate of link connectivity change than the 0-second pause time scenarios, due to an artifact of the random generation of the scenarios. This artifact is also visible as a slight bump at a pause time of 30 seconds in the performance graphs we present for 1 m/s in Section 5.

4.4 Validation of the Propagation Model and MAC Layer

Our propagation model uses standard equations and techniques, and it was verified by an expert in radio propagation modeling. We analyzed the power and simulated radio behavior as a function of distance between small groups of nodes to ensure that the propagation, capture, and carrier sense models were working as designed.

The 802.11 MAC implementation was studied in a variety of scenarios and independently verified by two of the authors. We experimentally tested that when all nodes are in range of each other, no data packets experience collisions (regardless of offered traffic load), and that each node is able to make progress sending packets. This verified that the carrier sense, RTS/CTS, and back-off mechanisms of 802.11 were working correctly.

4.5 Validation of the Routing Protocol Implementations

Each protocol implementation was studied and verified by at least two of the authors, and two independent implementations were made of both AODV and DSDV.

The results of each simulation were internally consistent. That is, the percentage of packets originated by the "application layer" sources that were not logged as either received or dropped at the end of the simulation was less than 0.01% (approximately 10 packets per simulation). These packets were almost certainly in transit at the end of the simulation, as the simulation originates between 40 and 120 packets per simulated second and terminates at exactly 900 seconds without a cool-down period.

The results of our simulations are, in fact, different from the few previously reported studies of some of these protocols. We explain the reasons for these differences in Sections 5, 6, and 7.

4.6 Metrics

In comparing the protocols, we chose to evaluate them according to the following three metrics:

- *Packet delivery ratio*: The ratio between the number of packets originated by the "application layer" CBR sources and the number of packets received by the CBR sink at the final destination.
- *Routing overhead*: The total number of routing packets transmitted during the simulation. For packets sent over multiple hops, *each* transmission of the packet (each hop) counts as one transmission.
- *Path optimality*: The difference between the number of hops a packet took to reach its destination and the length of the shortest path that physically existed through the network when the packet was originated.

Packet delivery ratio is important as it describes the loss rate that will be seen by the transport protocols, which in turn effects the maximum throughput that the network can support. This metric characterizes both the completeness and correctness of the routing protocol.

Routing overhead is an important metric for comparing these protocols, as it measures the scalability of a protocol, the degree to which it will function in congested or low-bandwidth environments, and its efficiency in terms of consuming node battery power. Protocols that send large numbers of routing packets can also increase the probability of packet collisions and may delay data packets in network interface transmission queues. We also evaluated the number of bytes of routing overhead caused by the source routing header required by DSR, and present those results in Section 6.1. We did not include IEEE 802.11 MAC packets or ARP packets in routing overhead, since the routing protocols could be run over a variety of different medium access or address resolution protocols, each of which would have different overhead. Our goal was to compare the routing protocols to each other, not to find the optimal performance possible in our scenarios.

In the absence of congestion or other "noise," path optimality measures the ability of the routing protocol to efficiently use network resources by selecting the shortest path from a source to a destination. We calculate it as the difference between the shortest path found internally by the simulator when the packet was originated, and the number of hops the packet actually took to reach its destination.

5 Simulation Results

As noted in Section 4.1, we conducted simulations using two different node movement speeds: a maximum speed of 20 m/s (average speed 10 m/s) and a maximum speed of 1 m/s. We first compare the four protocols based on the 20 m/s simulations, and then in Section 5.5 present data for 1 m/s for comparison. For all simulations, the communication patterns were peer-to-peer, with each run having either 10, 20, or 30 sources sending 4 packets per second.

5.1 Comparison Summary

Figures 2 and 3 highlight the relative performance of the four routing protocols on our traffic loads of 20 sources.

All of the protocols deliver a greater percentage of the originated data packets when there is little node mobility (i.e., at large pause time), converging to 100% delivery when there is no node motion. DSR and AODV-LL perform particularly well, delivering over 95% of the data packets regardless of mobility rate. In these scenarios, DSDV-SQ fails to converge at pause times less than 300 seconds.

The four routing protocols impose vastly different amounts of overhead, as shown in Figure 3. Nearly an order of magnitude separates DSR, which has the least overhead, from TORA, which has the most. The basic character of each protocol is demonstrated in the shape of its overhead curve. TORA, DSR, and AODV-LL are all on-demand protocols, and their overhead drops as the mobility rate drops. As DSDV-SQ is a largely periodic routing protocol, its overhead is nearly constant with respect to mobility rate.

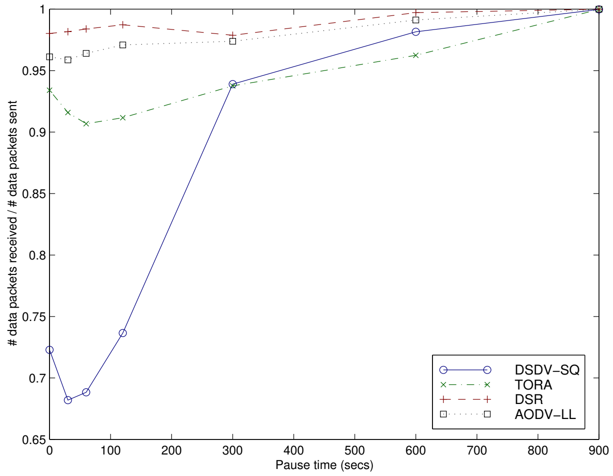


Figure 2 Comparison between the four protocols of the fraction of application data packets successfully delivered (packet delivery ratio) as a function of pause time. Pause time 0 represents constant mobility.

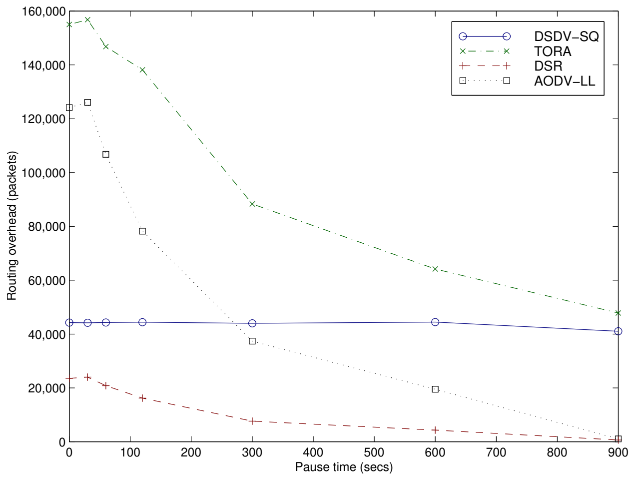


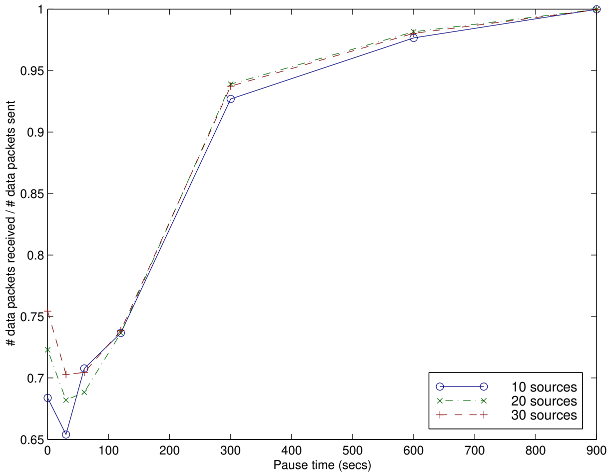
Figure 3 Comparison between the four protocols of the number of routing packets sent (routing overhead) as a function of pause time. Pause time 0 represents constant mobility.

The TORA results shown in Figures 2 and 3 at pause time 600 are the average of only 9 scenarios, as the overhead for the tenth scenario was much higher than the others due to significant congestion caused by the routing protocol. The complete results are included below and explained in Section 5.3.

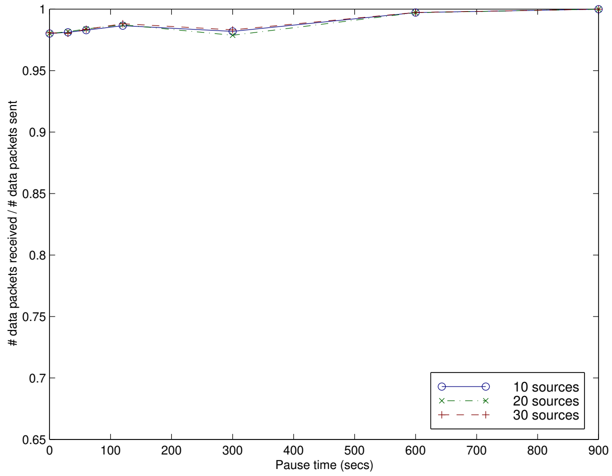
5.2 Packet Delivery Ratio Details

Figure 4 shows the fraction of the originated application data packets each protocol was able to deliver, as a function of both node mobility rate (pause time) and network load (number of sources). For DSR and AODV-LL, packet delivery ratio is independent of offered traffic load, with both protocols delivering between 95% and 100% of the packets in all cases.

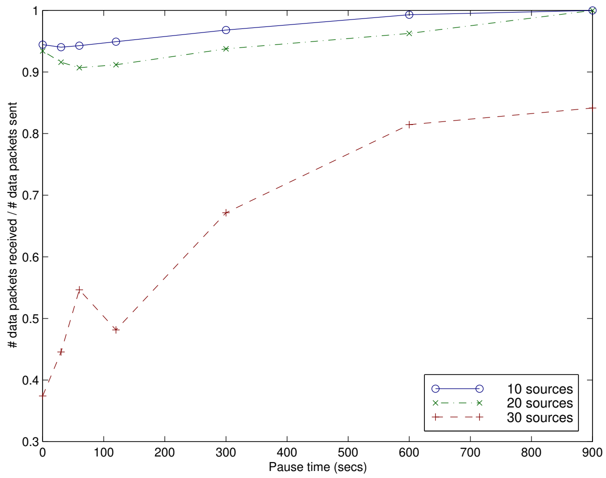
DSDV-SQ fails to converge below pause time 300, where it delivers about 92% of its packets. At higher rates of mobility (lower pause times), DSDV-SQ does poorly, dropping to a 70% packet delivery ratio. Nearly all of the dropped packets are lost because a stale routing table entry directed them to be forwarded over a broken link. As described in Section 3.1.2, DSDV-SQ maintains only one route per destination and consequently, each packet that the MAC layer is unable to deliver is dropped since there are no alternate routes.



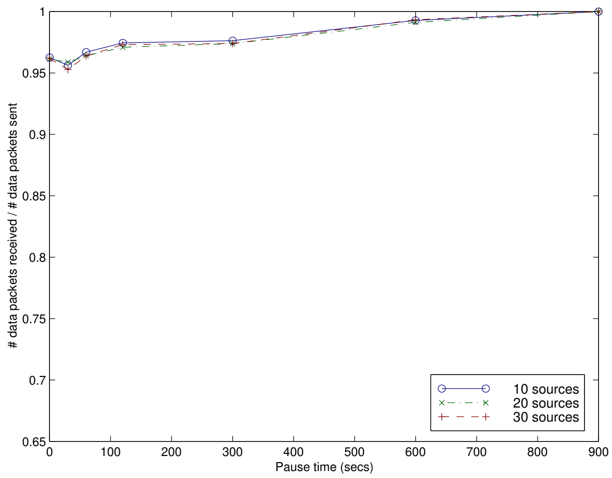
(a) DSDV-SQ



(b) DSR



(c) TORA



(d) AODV-LL

Figure 4 Packet delivery ratio as a function of pause time. TORA is shown on a different vertical scale for clarity (see Figure 2).

TORA does well with 10 or 20 sources, delivering between 90% and 95% of originated data packets even at the highest rate of node mobility (pause time 0). The majority of the packet drops are due to the creation of short-lived routing loops that are a natural part of its link-reversal process. Consider a node **A** routing packets to **C** via **B**. If **B**'s link to **C** breaks, **B** will reverse its link to **A**, transmit an UPDATE to notify its neighbors it has done this, and begin routing packets to **C** via **A**. Until **A** receives the UPDATE, data packets to **C** will loop between **A** and **B**. Our implementation of TORA detects when the next-hop of a packet is the same as the previous-hop and drops the data packet, since experiments showed that allowing these packets to loop until their TTL expires or the loop resolves causes more packets to be dropped overall, as the looping data packets interfere with the ability of other nearby nodes to successfully exchange the broadcast UPDATE packet that will resolve their routing loop.

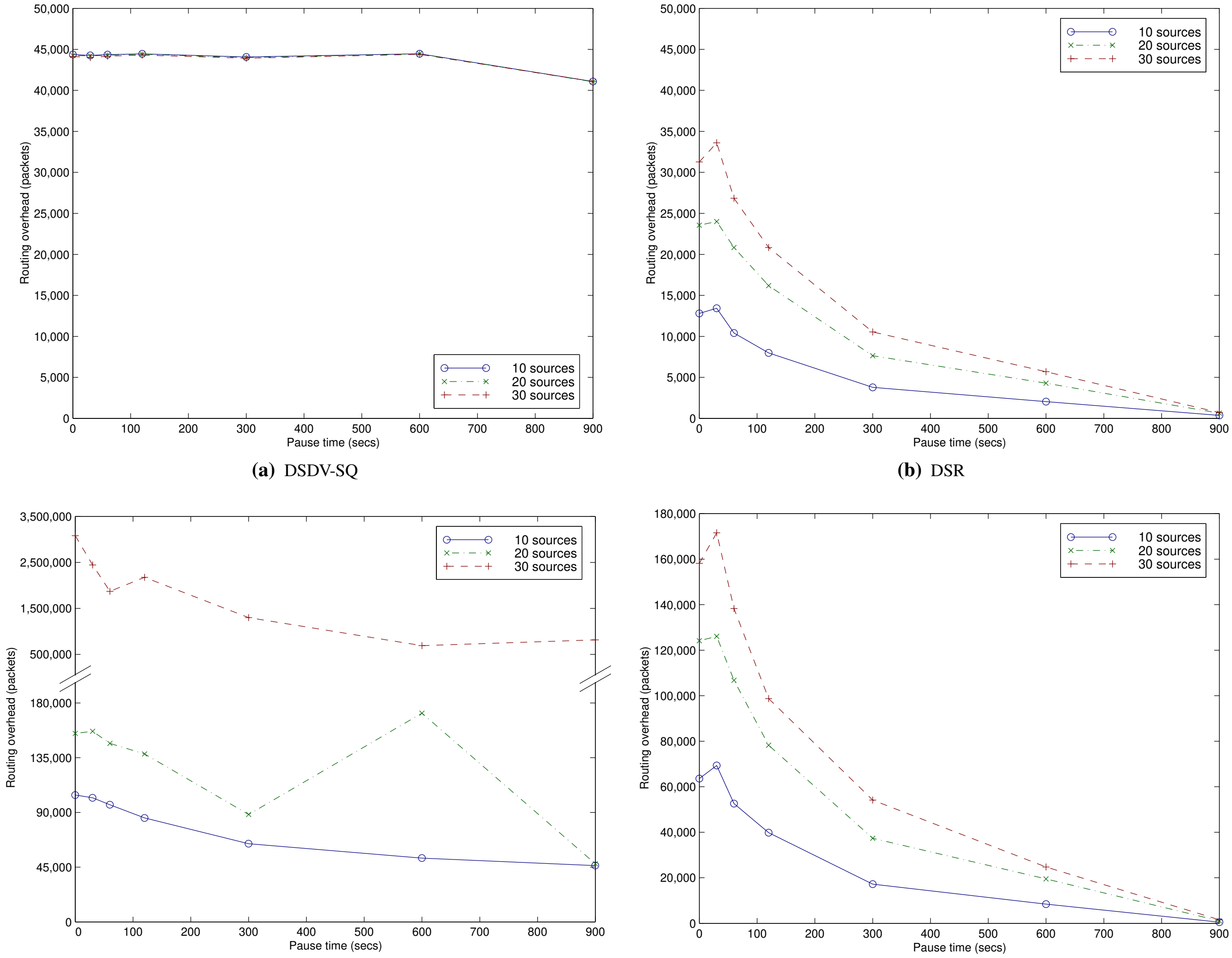
With 30 sources, TORA's average packet delivery ratio drops to 40% at pause time 0, although upon examination of the data we found that variability was extremely large, with packet delivery ratios ranging from 8% to 91%. In most of these scenarios, TORA fails to converge because of increased congestion, as explained in Section 5.3. A very recently released revision to the IMEP specification [4] claims to improve the reliable control message delivery semantics provided by IMEP, which might eliminate some of the behaviors seen here. However, these new mechanisms add more packet overhead to TORA/IMEP which, as shown in Section 5.3, is already higher than the other protocols studied here.

5.3 Routing Overhead Details

Figure 5 shows the number of routing protocol packets sent by each protocol in obtaining the delivery ratios shown in Figure 4. DSR and DSDV-SQ are plotted on a the same scale as each other, but AODV-LL and TORA are each plotted on different scales to best show the effect of pause time and offered load on overhead. TORA, DSR, and AODV-LL are on-demand routing protocols, so as the number of sources increases, we expect the number of routing packets sent to increase because there are more destinations to which the network must maintain working routes.

DSR and AODV-LL, which use *only* on-demand packets and very similar basic mechanisms, have almost identically shaped curves. Both protocols exhibit the desirable property that the incremental cost of additional sources decreases as sources are added, since the protocol can use information learned from one route discovery to complete a subsequent route discovery.

However, the absolute overhead required by DSR and AODV-LL are very different, with AODV-LL requiring about 5 times the overhead of DSR when there is constant node motion (pause time 0). This dramatic increase in AODV-LL's overhead occurs because each of its route discoveries typically propagates to every node in the ad hoc network. For example, at pause time 0 with 30 sources, AODV-LL



(a) DSDV-SQ **(b)** DSR **(c)** TORA **(d)** AODV-LL

Figure 5 Routing overhead as a function of pause time. TORA and AODV-LL are shown on different vertical scales for clarity (see Figure 3).

initiates about 2200 route discoveries per 900-second simulation run, resulting in around 110,000 ROUTE REQUEST transmissions. In contrast, DSR limits the scope and overhead of ROUTE REQUEST packets by using caching from forwarded and promiscuously overheard packets and using non-propagating ROUTE REQUESTs as described in Section 3.3.2, which results in DSR sending only 950 non-propagating requests and 300 propagating requests per simulation run.

TORA's overhead is the sum of a constant mobility-independent overhead and a variable mobility-dependent overhead. The constant overhead is the result of IMEP's neighbor discovery mechanism, which requires each node to transmit at least 1 HELLO packet per BEACON period (1 second). For 900-second simulations with 50 nodes, this results in a minimum overhead of 45,000 packets. The variable part of the overhead consists of the routing packets TORA uses to create and maintain routes, multiplied by the number of retransmission and acknowledgment packets IMEP uses to ensure their reliable, in-order delivery.

In many of our scenarios with 30 sources, TORA essentially underwent congestive collapse. A positive feedback loop developed in TORA/IMEP wherein the number of routing packets sent caused numerous MAC-layer collisions, which in turn caused data, ACK, and HELLO packets to be lost. The loss of these packets caused IMEP to erroneously believe that links to its neighbors were breaking, even in the pause time 900 scenarios when all nodes were stationary. TORA reacted to the perceived link breakages by sending more UPDATEs, which closed the feedback loop by directly causing more congestion. More importantly, each UPDATE sent required reliable delivery, which increased the system's exposure to additional erroneous link failure detections, since the failure to receive an ACK from retransmitted UPDATES was treated as a link failure indication. In the worst runs, TORA generated over 10 million objects, which IMEP aggregated into 1.6 million packets requiring reliable delivery. In the few 30-source runs where congestion did not develop, the overhead varied from 639,000 packets at pause time 0 to 47,000 packets at pause time 900.

DSDV-SQ has approximately constant overhead, regardless of movement rate or offered traffic load. This constant behavior arises because each destination **D** broadcasts a periodic update with a new sequence number every 15 seconds. With 50 unsynchronized nodes in the simulation, at least one node broadcasts a periodic update during each second. DSDV-SQ considers the receipt of a new sequence number for a node to be important enough to distribute immediately (Section 3.1), so each node that receives **D**'s periodic update generates a triggered update. These triggered updates flood the network, as each node receiving one learns a new sequence number and so also generates a triggered update. Each node limits the rate at which it sends triggered updates to one per second, but since there is at least

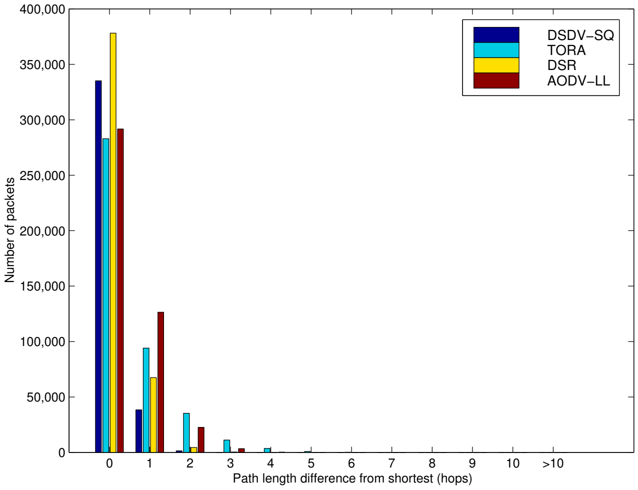


Figure 6 Difference between the number of hops each packet took to reach its destination and the optimal number of hops required. Data is for 20 sources.

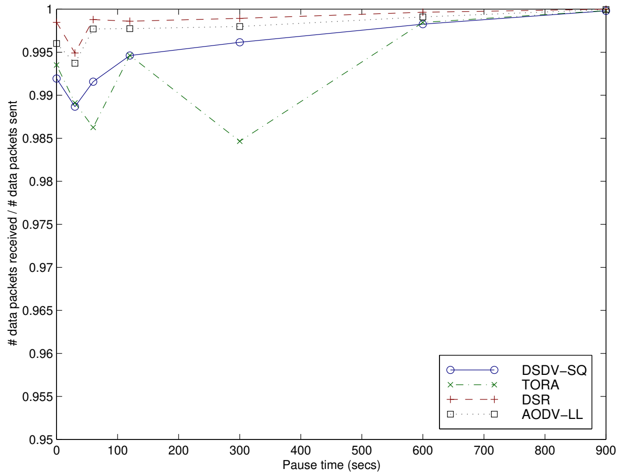


Figure 7 Comparison of the fraction of application data packets successfully delivered as a function of pause time. Speed is 1 m/s.

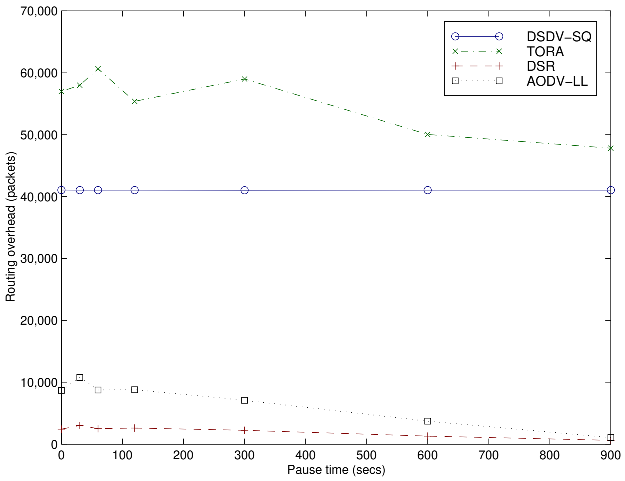


Figure 8 Comparison of the number of routing packets sent as a function of pause time. Speed is 1 m/s.

one new sequence number per second, every node transmits triggered updates at the maximum permitted rate. Therefore, although the base periodic action of DSDV-SQ is once per 15 seconds, the effective rate of a group of nodes is one update per node per second, yielding an overhead of 45,000 packets for a 900-second, 50-node simulation.

5.4 Path Optimality Details

As described in Section 4, an internal mechanism of our simulator knows the length of the shortest possible path between all nodes in the network at any time and labels all packets with this path length when they are originated. Figure 6 shows the difference between this shortest path length and the length of the paths actually taken by data packets. A difference of 0 means the packet took a shortest path, and a difference greater than 0 indicates the number of extra hops the packet took.

Both DSDV-SQ and DSR use routes very close to optimal. TORA and AODV-LL each have a significant tail, taking up to 4 or more hops longer than optimal for some packets, although TORA was not designed to find shortest paths. For space reasons, Figure 6 aggregates the data from all pause times into one graph. When the data are broken out by pause time, DSDV-SQ and DSR do very well regardless of pause time, with no statistically significant change in optimality of routing with respect to node mobility rate. TORA and AODV-LL, on the other hand, each show a significant difference with respect to pause time in the length of the routes they use relative to the shortest possible routes. When node mobility is very low, they use routes that are significantly closer to the shortest possible routes than when nodes are moving.

5.5 Lower Speed of Node Movement

In order to explore how the protocols scale as the rate of topology change varies, we changed the maximum node speed from 20 m/s to 1 m/s and re-evaluated all four protocols over scenario files using this lower movement speed. Figures 7 and 8 show the results of this experiment when using 20 sources. All of the protocols deliver more than 98.5% of their packets at this movement speed. Unlike in the 20 m/s scenarios, where DSDV-SQ could not converge, it delivers excellent performance in the 1 m/s scenarios.

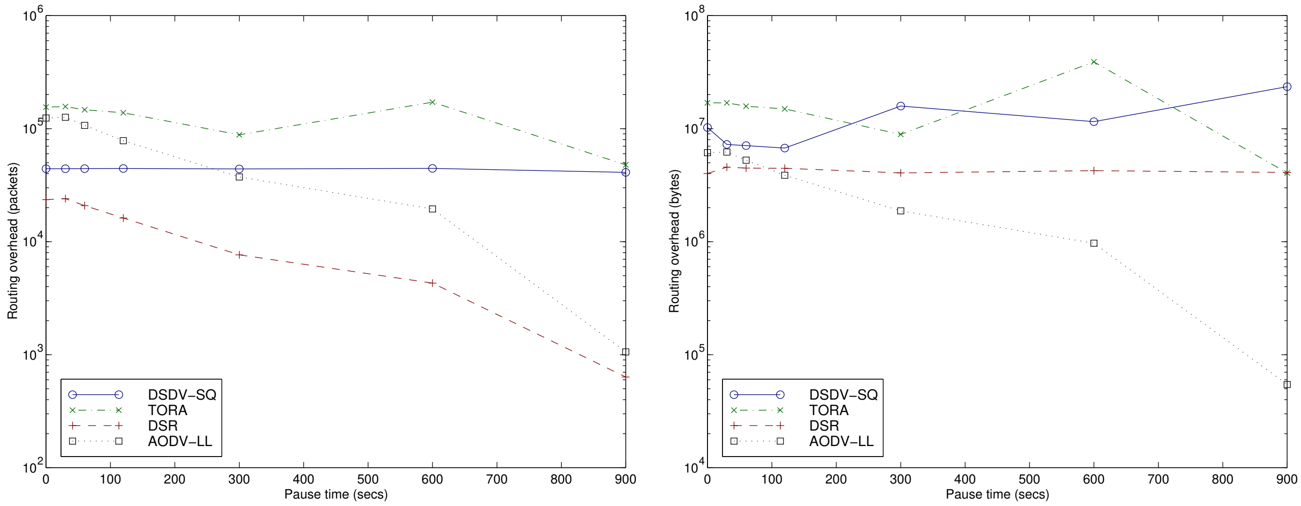
Even at this slower rate of movement, each of the routing protocols generated very different amounts of overhead. Neither DSR nor AODV-LL were seriously challenged by this set of scenarios, as the overhead increases only mildly as pause time decreases. The separation between DSR and AODV-LL, however, has grown from a factor of 5 to nearly a factor of 10 because DSR's caching is even more effective at lower speeds where the cached information goes stale more slowly.

Due to its largely periodic nature, DSDV-SQ continues to have a constant overhead of approximately 41,000 packets, while TORA's overhead is dominated by the link/status sensing mechanism of IMEP, which amounts to one packet per node per second, or a total of 45,000 packets per simulation (Section 5.3).

6 Additional Observations

6.1 Overhead in Source Routing Protocols

When comparing the number of routing overhead packets sent by each of the protocols, DSR clearly has the lowest overhead (Figure 5). The data for 20 sources is reproduced in Figure 9(a) on a semi-log axis for clarity. However, if routing overhead is measured in bytes and includes the bytes of the source route header that DSR places in each packet, DSR becomes more expensive than AODV-LL except at the highest rates of mobility, although it still transmits fewer bytes of routing overhead than does DSDV-SQ or TORA. AODV-LL uses a Route Discovery mechanism based on DSR's, but it creates hop-by-hop routing state in each node along a path in order to eliminate



(a) Routing overhead in packets.

(b) Routing overhead in bytes.

Figure 9 Contrasting routing overhead in packets and in bytes. Both graphs use semi-log axes.

the overhead of source routing from data packets. This reduction in overhead bytes is shown in Figure 9(b).

It is unclear whether this improvement in bytes of overhead is significant for real world protocol operation, because the majority of AODV-LL overhead bytes are carried in many small packets. The cost to acquire the medium to transmit a packet is significantly more expensive in terms of power and network utilization than the incremental cost of adding a few bytes to an existing packet, so the actual cost of the source route header in DSR is less than the number of bytes might indicate. A completely fair comparison based on overhead in bytes would also have to include the cost of physical layer framing and MAC protocol bytes, which we have deliberately factored out since the routing protocols could be run over many different MAC implementations, each of which would have a different overhead.

6.2 The Effect of Triggered Updates in DSDV

As noted in Section 3.1.2, DSDV can employ either of two strategies for determining when to send triggered updates. In the first strategy, DSDV-SQ, a node sends a triggered update each time it receives a new sequence number for some destination. As shown in Figure 10, DSDV-SQ delivers over 99% of its packets for all pause times when the maximum node speed is 1 m/s. In the 20 m/s case, DSDV-SQ's packet delivery ratio falls to 95% at a pause time of 300 seconds and further decreases at higher mobility rates as DSDV-SQ is unable to converge. Figure 11 shows that for both the 1 m/s and 20 m/s data, DSDV-SQ's routing overhead is approximately 45,000 packets for all pause times, as discussed in Section 5.3.

The second scheme for sending triggered updates, which we call simply DSDV, requires that they be sent only when a new metric is received for a destination. In this case, link breakages are not detected as quickly as in DSDV-SQ, generally resulting in more dropped packets.

For a movement speed of 1 m/s, DSDV delivers fewer packets than DSDV-SQ, with its packet delivery ratio decreasing to 95% at pause time 0. While DSDV's routing overhead is a factor of 4 smaller, the fact that its routing overhead is constant indicates that a movement speed of 1 m/s does not exercise the routing protocol fully. At 20 m/s, both DSDV-SQ and DSDV fail to converge, causing a large percentage of data packets to be dropped. However, the DSDV triggering scheme reduces the relative routing overhead by a factor of 4 at pause time 900 and by a factor of 2 at pause time 0.

6.3 Reliability Issues with Broadcast Packets

Because broadcast packets are not receiver directed, there is no way to reserve the wireless medium at the receivers before transmitting a broadcast packet (e.g., with an RTS/CTS exchange). Consequently, broadcast packets are inherently less reliable than unicast packets. This difference does not exist in wired networks and represents a fundamental limitation of wireless networks that must be accounted for in the design of ad hoc network routing protocols. Upon sampling a number of our scenarios, we found that over any single hop, 99.8% of unicast data packets are received successfully, while only 92.6% of broadcast packets are received, based on counting the number of receivers within transmission range of the broadcasting node. The difference between the two numbers is due to collisions. In future work, we will examine how the difference varies with the average degree of the nodes, the size of the broadcast packets, and the relative proportion of broadcast packets.

6.4 Interaction of ARP with On-Demand Protocols

When an on-demand routing protocol receives packets to a destination for which it does not have a route, the protocol typically buffers the packets in the routing layer until it can discover a route for the packets. Once the routing protocol finds a route, it sends the queued packets down to the link-layer for transmission. In the course of our early experiments, however, we observed a serious layer-integration problem that would effect any on-demand protocol running on top of an ARP implementation similar to that in BSD Unix [23].

Our ARP code, like the BSD code, buffers one packet per destination awaiting a link layer address. If a series of packets are passed by the routing layer to the ARP code with a next-hop destination whose link-layer address is unknown, all but the last of these packets will be dropped by ARP. In our simulation, we remedied this by pacing the rate at which packets are passed from the routing queue, though the implementation of ARP could be modified to buffer additional packets, or the routing protocol could be coded to check that the ARP layer has a link-layer address for the next-hop before passing it packets from the routing queue.

7 Related Work

Some simulation results for DSDV, TORA, and DSR have been presented in earlier papers, although those simulations used substantially different input parameters than ours and did not simulate the wireless network as accurately. We are not aware of any previously published performance results for AODV.

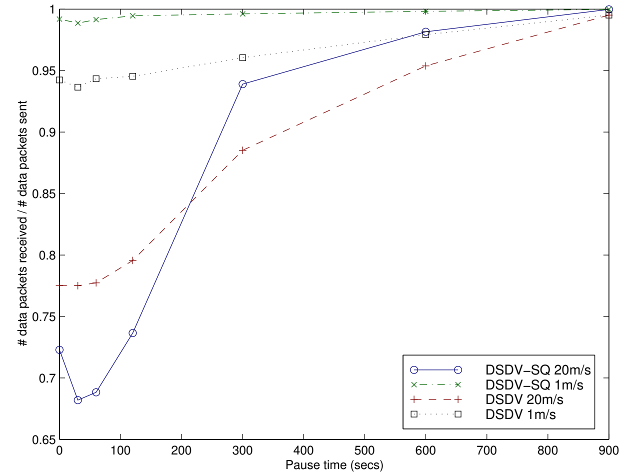


Figure 10 Fraction of originated data packets successfully delivered by DSDV-SQ and DSDV.

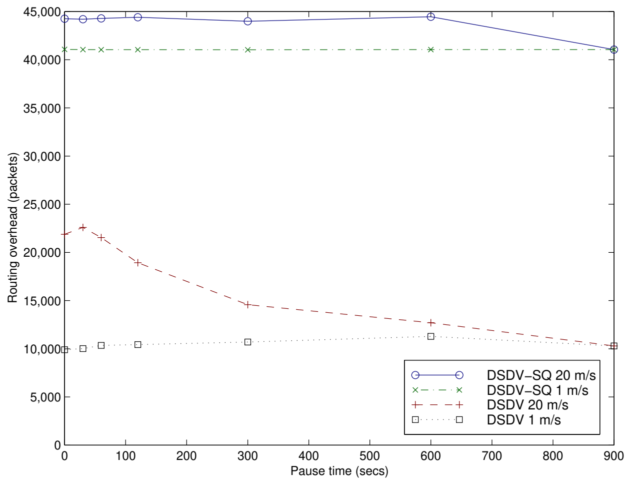


Figure 11 Routing overhead as a function of pause time for DSDV-SQ and DSDV.

7.1 Park and Corson

An earlier simulation of TORA was done by Park and Corson [16], who compared TORA to an "idealized" link state routing protocol. Their results are quite different from those presented in this paper, showing TORA delivering over 90% of its packets in all cases, but these results are incomparable to ours because of the many simplifications they made in simulating the environment. In order to avoid congestion, their simulation used a packet transmission rate of only 4, 1.5, or 0.6 packets per *minute* per node. Simulations were run for 2 hours, the mean time between failure for links was varied from 32 minutes to 1 minute, and the average network connectivity was artificially held constant at 90%, 70%, or 50%.

Their simulator modeled the network as a "densely-connected honeycomb" with constant node density. There is no notion of node mobility. Each node is connected to a fixed set of neighbors by separate links that cycle between an active and an inactive state independent of all other links. These links are error free, and no dynamics below the network layer are modeled. Radio propagation, medium access, collisions, and physical node mobility are completely ignored; it is even possible for a node to correctly receive two simultaneous transmissions. In their simulation, link transitions cause interrupts that give the protocols immediate feedback whenever a link goes up or down. In reality, though, a node can only detect that a link has broken if it is trying to use the link or if it fails to receive expected periodic beacons over it, and a node can only recognize the existence of a new neighbor when it receives a packet from that neighbor.

An earlier paper [3] presented simulation results for the protocol on which TORA is in part based, but the simulator used in that study had the same problems as the simulator used by Park and Corson, and the metrics used are incomparable to ours.

7.2 Johnson and Maltz

We have previously simulated DSR [10] using the same mobility model as in this paper. In this previous work, movement speeds ranged from 0.3 to 0.7 m/s, with the nodes moving about in a 9m × 9m space using shorter range infrared wireless transmitters with a 3-meter range.

This simulation lacked a realistic model of radio propagation and a MAC layer such as IEEE 802.11. These missing pieces greatly simplify the problem faced by the routing protocol, as propagation delay, capture effects, MAC-layer collisions, and the effects of congestion due to large packet sizes are unaccounted for. Furthermore, broadcast and unicast packets were delivered with the same probability, and, as noted in Section 6.3, this is not a realistic assumption. This earlier simulation study also did not consider ad hoc networks of more than 24 nodes.

In the earlier simulation, we characterized the path optimality of the routing protocol by reporting the *ratio* of the average route length used to the optimal route length that could have been used if the routing protocol had perfect information. However, we did not report the actual route lengths used, and since both numbers are averages, this ratio tended to blur the dynamics of the protocol. As a result, in this paper we present path optimality data as the *difference* between the optimal and actual path lengths used.

7.3 Freisleben and Jansen

DSDV and DSR were recently simulated and compared by Freisleben and Jansen [7]. Their simulations used configurations of 10 or 25 mobile nodes, with movement speeds relative to transmission range approximately 6 times faster than in our simulations, making some of their results incomparable to ours. Furthermore, although their results could be explained by many factors, such as congestion, collisions, or the failure of the routing protocols to converge, the authors do not analyze or interpret the results and it is not possible to assess the impact of these factors from the data presented.

All of their results also suffer due to a number of deficiencies in their simulation and implementation of the protocols. All events in their simulator take place in regular time steps, resulting in perfectly synchronized behavior of a number of separate mobile nodes in some cases. For example, upon receipt of a broadcast packet, all nodes who attempt to send a response packet will experience a collision on their RTS, requiring a binary exponential backoff and retransmission attempt. Furthermore, they do not buffer any packets while waiting for a ROUTE REPLY to be returned during a DSR Route Discovery, causing large numbers of packets to be dropped needlessly.

8 Conclusions

The area of ad hoc networking has been receiving increasing attention among researchers in recent years, as the available wireless networking and mobile computing hardware bases are now capable of supporting the promise of this technology. Over the past few years, a variety of new routing protocols targeted specifically at the ad hoc networking environment have been proposed, but little performance information on each protocol and no detailed performance comparison between the protocols has previously been available.

This paper makes contributions in two areas. First, we describe our modifications to the *ns* network simulator to provide an accurate simulation of the MAC and physical-layer behavior of the IEEE

802.11 wireless LAN standard, including a realistic wireless transmission channel model. This new simulation environment provides a powerful tool for evaluating ad hoc networking protocols and other wireless protocols and applications. Second, using this simulation environment, we present the results of a detailed packet-level simulation comparing four recent multi-hop wireless ad hoc network routing protocols. These protocols, DSDV, TORA, DSR, and AODV, cover a range of design choices, including periodic advertisements vs. on-demand route discovery, use of feedback from the MAC layer to indicate a failure to forward a packet to the next hop, and hop-by-hop routing vs. source routing. We simulated each protocol in ad hoc networks of 50 mobile nodes moving about and communicating with each other, and presented the results for a range of node mobility rates and movement speeds.

Each of the protocols studied performs well in some cases yet has certain drawbacks in others. DSDV performs quite predictably, delivering virtually all data packets when node mobility rate and movement speed are low, and failing to converge as node mobility increases. TORA, although the worst performer in our experiments in terms of routing packet overhead, still delivered over 90% of the packets in scenarios with 10 or 20 sources. At 30 sources, the network was unable to handle all of the traffic generated by the routing protocol and a significant fraction of data packets were dropped. The performance of DSR was very good at all mobility rates and movement speeds, although its use of source routing increases the number of routing overhead bytes required by the protocol. Finally, AODV performs almost as well as DSR at all mobility rates and movement speeds and accomplishes its goal of eliminating source routing overhead, but it still requires the transmission of many routing overhead packets and at high rates of node mobility is actually more expensive than DSR.

Acknowledgments

Many thanks are due to Dan Stancil, who reviewed our propagation model for correctness and provided insights on the details of radio propagation. We would also like to thank Pravin Bhagwat, Tracy Camp, Scott Corson, Vincent Park, and Charles Perkins for answering our questions about the details of the operation of DSDV, TORA/IMEP, and AODV.

References

[1] 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.

[2] Josh Broch, David B. Johnson, and David A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-00.txt, March 1998. Work in progress.

[3] M. Scott Corson and Anthony Ephremides. A distributed routing algorithm for mobile wireless networks. *Wireless Networks*, 1(1):61–81, February 1995.

[4] M. Scott Corson, S. Papademetriou, Philip Papadopoulos, Vincent D. Park, and Amir Qayyum. An Internet MANET Encapsulation Protocol (IMEP) Specification. Internet-Draft, draft-ietf-manet-imep-spec-01.txt, August 1998. Work in progress.

[5] M. Scott Corson and Vincent D. Park. An Internet MANET Encapsulation Protocol (IMEP) Specification. Internet-Draft, draft-ietf-manet-imep-spec-00.txt, November 1997. Work in progress.

[6] Kevin Fall and Kannan Varadhan, editors. *ns* notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from http://www-mash.cs.berkeley.edu/ns/.

[7] Bernd Freisleben and Ralph Jansen. Analysis of routing protocols for ad hoc networks of mobile comuters. In *Proceedings of the 15th IASTED International Conference on Applied Informatics*, pages 133–136, Innsbruck, Austria, February 1997. IASTED-Acta Press.

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

[9] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[10] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[11] Phil Karn. MACA — A new channel access method for packet radio. In *Proceedings of the 9th Computer Networking Conference*, pages 134–140, September 1990.

[12] Barry M. Leiner, Robert J. Ruth, and Ambatipudi R. Sastry. Goals and challenges of the DARPA GloMo program. *IEEE Personal Communications*, 3(6):34–43, December 1996.

[13] National Science Foundation. Research priorities in wireless and mobile communications and networking: Report of a workshop held March 24–26, 1997, Airlie House, Virginia. Available at http://www.cise.nsf.gov/anir/ww.html.

[14] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, pages 1405–1413, April 1997.

[15] Vincent D. Park and M. Scott Corson. Temporally-Ordered Routing Algorithm (TORA) version 1: Functional specification. Internet-Draft, draft-ietf-manet-tora-spec-00.txt, November 1997. Work in progress.

[16] Vincent D. Park and M. Scott Corson. A performance comparision of TORA and Ideal Link State routing. In *Proceedings of IEEE Symposium on Computers and Communication '98*, June 1998.

[17] Charles Perkins. Ad Hoc On Demand Distance Vector (AODV) routing. Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997. Work in progress.

[18] 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. A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps.

[19] David 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.

[20] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, New Jersey, 1996.

[21] Neil Siegel, Dave Hall, Clint Walker, and Rene Rubio. The Tactical Internet Graybeard Panel briefings. U.S. Army Digitization Office. Available at http://www.ado.army.mil/Briefings/Tact%20Internet/index.htm, October 1997.

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

[23] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley, Reading, Massachusetts, 1995.