

---

## Abstract

This article describes our experiences building a multihop wireless ad hoc network of eight nodes driving around a 700 m by 300 m site. Each node runs the Dynamic Source Routing protocol and interfaces seamlessly with existing Internet infrastructure and the Mobile IP protocol. We present quantitative results from data collected during runs of our testbed under a composite workload including voice, bulk data, and real-time data. Based on careful analysis of our data, we highlight radio propagation issues that network protocols will need to address in the future.

# Lessons from a Full-Scale Multihop Wireless Ad Hoc Network Testbed

---

DAVID A. MALTZ AND JOSH BROCH, AON NETWORKS

DAVID B. JOHNSON, RICE UNIVERSITY

**D**uring the seven months from August 1998 to February 1999, we designed and implemented a full-scale, multihop ad hoc network testbed [1] to enable the evaluation of ad hoc network performance in the field. From February through April, the testbed was used to demonstrate our routing protocol and the potential of ad hoc networking to our sponsors, and also as a research tool to allow us to experiment with the carrying capacity and behavior of a fully deployed network. Our ad hoc network testbed abstractly models an example deployment consisting of a remote work site on which ad hoc networking is used for communication between vehicles and other equipment on the site.

Routing in the testbed is performed by the Dynamic Source Routing protocol (DSR) [2–5], and features within this protocol allow the entire testbed to be seamlessly integrated with existing Internet infrastructure. The nodes in the ad hoc network were implemented as cars, driving around a 700 m by 300 m site, with a Lucent WaveLAN wireless LAN radio mounted on the roof of each car. We implemented an extensive set of network monitoring tools, which track the precise location and protocol processing activities at each node and allow us to analyze the behavior of the network. A series of traffic generators were also developed to stress the network with a variety of different traffic loads.

This article briefly describes the results of our initial experiments on the testbed, and discusses the considerable effect that real-world radio propagation had on the protocols in the network. The quantitative numbers reported in later sections of this article are intended to serve three purposes. First, they validate the architectural decisions made in constructing the protocol implementation. Second, they provide protocol

designers with more data points on the characteristics of the outdoor wireless environment in which actual protocols for many uses must run. Finally, they point out several interesting consequences of real-world radio propagation.

## Testbed Overview

Our primary design goal for the testbed was to challenge the network protocols to the point where they were stressed, by subjecting them to higher rates of topology change than previous testbeds had explored [6, 7]. With the vehicles, radios, and site used in our testbed, we forced the protocols to operate in an environment in which all links between nodes change status at least every 220 s. Ignoring the additional factor of packet loss due to wireless errors, on average, the network topology changed every 4 s.

## Network Topology

Figure 1 shows a logical view of the ad hoc network testbed. The ad hoc network included five moving car-mounted nodes, labeled T1–T5, and two stationary nodes, labeled E1 and E2. Each of these nodes communicates using 900 MHz WaveLAN-I radios. These radios do not implement the IEEE 802.11 MAC protocol [8], since at the time the testbed was built, WaveLAN-IEEE radios were not available. The ad hoc network is connected to a *field office* using a 2.4 GHz point-to-point wireless link over a distance of about 700 m. This point-to-point link allowed us to locate the ad hoc network in an arbitrary location with respect to the field office, and it does not interfere with the 900 MHz radio interfaces on the individual ad hoc network nodes.

At the field office is a router (labeled R) that connects both the ad hoc network and an IP subnet at the field office back to the *central office* via a wide-area network. The visualizer node (labeled V) in the field office is used to monitor the status of the ad hoc network, and the Global Positioning System (GPS) reference station (labeled RS), located on the roof of the field office, is responsible for sending differential real-time kinematic (RTK) GPS corrections across the ad hoc network to the moving cars.

The central office contains the IP *home network* for a *roving node* (labeled RN) that drives between the central office and the ad hoc network. The routing node is able to partici-

---

*This work was performed while all the authors were at Carnegie Mellon University, and it was supported in part by the National Science Foundation (NSF) under CAREER Award NCR-9502725, by Caterpillar Corporation, and by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061. David Maltz was also supported under an Intel 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, Caterpillar, Intel, Carnegie Mellon University, AON Networks, Rice University, or the U.S. Government.*

Application	Rate	Protocol	Size
Voice	6/h/node	UDP	Average of 180 kbytes
Data	5/h/node	TCP	30, 60, or 90 kbytes
Location-dependent	When near E1	TCP	Average of 150 kbytes
GPS	1 packet/s multicast	UDP	150 bytes
PCTd	1 packet/s/node unicast	UDP	228 bytes

■ **Table 1.** Load offered to the network by nodes in the testbed.

pate in any of three networks: its home wireless LAN, Verizon Wireless's Mobile Cellular Digital Packet Data service (CDPD), and the ad hoc network. Another node (labeled HA) provides Mobile IP home agent services [9] for the roving node, enabling it to leave the central office and still maintain routing connectivity with all of the other nodes in the Internet.

During a typical experiment with the testbed, which we call a *run*, the drivers of each of the cars carrying an ad hoc network node follow a set course at speeds varying from 25 to 40 km/h (15–25 mi/h). Each run lasts for between 30 and 120 min. The road we use is open to general vehicle traffic and has several stop signs, so the velocity of each node varies in a complex fashion, just as it would in any real network. Likewise, the nodes are constrained to move along the paved surfaces of the site. This prevents us from testing the arbitrary topologies used in some theoretical simulations on abstract flat planes, but enables us to evaluate the performance we can expect in a real application.

During each run, the network was subjected to the composite workload shown in Table 1, consisting of synthetic voice calls, bulk data transfer, location-dependent transfers, and real-time data. The workload includes each node making one voice call to every other node once per hour, each node transferring a data file to every other node once per hour, each moving node (T1–T5) making a location-dependent transfer to E1 when located within 150 m of E1, and multicast differential RTK GPS correction packets. Finally, the workload also includes real-time situational awareness data sent by what we call the *Position and Communication Tracking daemon (PCTd)* running on each node. This data, sent once per second to the visualizer machine located at the field office, contains the current location of the node, taken from the node's GPS unit, and status information on the node, such as the number of packets it has forwarded, dropped, queued, originated, or retransmitted. The visualizer machine continuously displays on a map of the site the last known location of each node; it can graph the status information, and it logs all the data it receives, allowing a detailed replay of the run after the fact.

### Network Configuration

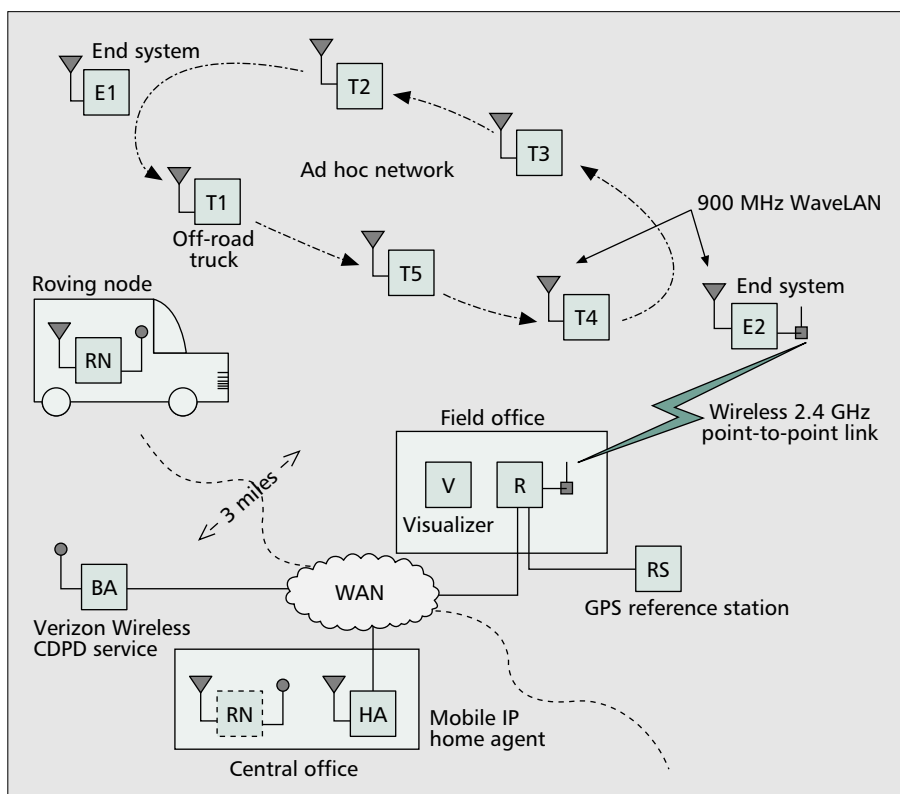
All communication among the ad hoc network nodes, T1–T5, E1, and E2, is routed by DSR. Although DSR operates at the IP layer of the network stack — open systems interconnection

(OSI) layer 3 — and permits interoperability between different physical network interfaces, our DSR implementation conceptually operates as a virtual link layer just under the normal IP layer.

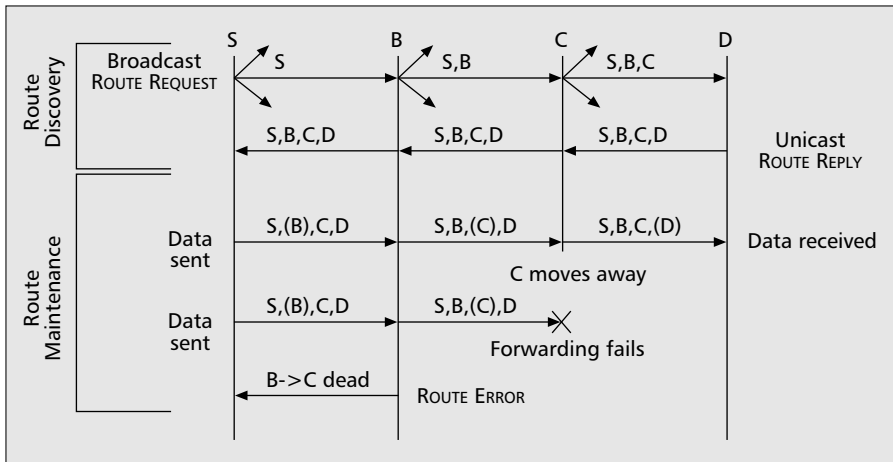
Nodes T1–T5, E1, and E2 are assigned IP addresses from a single subnet, with E2 acting as a gateway between the Internet and the ad hoc network subnet. E2 was manually configured to use the DSR protocol for communication on one network interface (the 900 MHz WaveLAN link) and use normal IP routing over the other interface (the

2.4 GHz point-to-point link to its default router, R). Packets from nodes in the Internet destined to addresses in the ad hoc subnet are routed by normal means to E2, which has a statically configured route directing them out the network interface to the ad hoc network. Once forwarded into the ad hoc network by E2, DSR takes care of routing the packets to their final destination, which often requires multiple hops inside the ad hoc network. As explained later, nodes in the ad hoc subnet (i.e., T1–T5 and E1) did not have to be configured to use E2 as a default router: when nodes in a DSR ad hoc network send packets to nodes outside the ad hoc network, the DSR protocol itself automatically routes the packets to the nearest gateway (E2, in this case), where they are forwarded into the Internet. The gateway node, E2, also provides Mobile IP foreign agent services to any Mobile IP nodes that visit the ad hoc network.

The routing node, RN, has several methods available for connecting to the Internet, and uses Mobile IP [9] to choose the best method as it drives around the city. The RN is normally within range of the WaveLAN network at the central office, and its WaveLAN network interface carries an IP address belonging to the central office subnet. When RN is roving away from the central office, it uses Mobile IP to register a care-of address with its home agent on the central office subnet. While RN has a care-of address registered with the



■ **Figure 1.** A logical overview of the testbed network.



**Figure 2.** Basic operation of the DSR protocol showing the building of a source route during the propagation of a ROUTE REQUEST, the source route's return in a ROUTE REPLY, its use in forwarding data, and the sending of a ROUTE ERROR upon forwarding failure. The next hop is indicated by the address in parentheses.

home agent, the home agent intercepts packets destined to RN, and tunnels each to the care-of address using encapsulation.

When RN cannot use its primary WaveLAN interface because it is not in range of any other WaveLAN radios, it uses its CDPD modem to connect to the CDPD service from Verizon Wireless (at the time known as Bell Atlantic Mobile), and registers its CDPD IP address with its home agent. Once RN realizes it is in range of a DSR network, it can use DSR to communicate directly with the other nodes in the ad hoc network. To enable packets from nodes outside the DSR network to reach RN, it registers itself with its home agent via the foreign agent at E2, just as in normal Mobile IP. When E2 receives a tunneled packet, it checks to see if the packet is destined to a node registered as visiting the ad hoc network. If so, E2 routes the packet to the visiting node using DSR.

## DSR Overview

The Dynamic Source Routing protocol [2-4] is based on source routing, such that the originator of each packet determines an ordered list of nodes through which the packet must pass while traveling to the destination. The key advantage of a source routing design is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packet's source has already made all of the routing decisions. This fact, coupled with the entirely *on-demand* nature of the protocol, eliminates the need for the periodic route advertisement and neighbor detection packets present in other protocols. Although DSR uses source routes, most packets do not need to incur the overhead of carrying an explicit source route header [2, 10].

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 some destination D obtains a source route to D. To reduce the cost of Route Discovery, each node maintains a Route Cache of source routes it has learned or overheard. Route Maintenance is the mechanism by which a packet's originator S detects if the network topology has changed such that it can no longer use that packet's route to the destination D because some of the nodes listed on the route have moved out of range of each other. Figure 2 shows the basic operation of DSR.

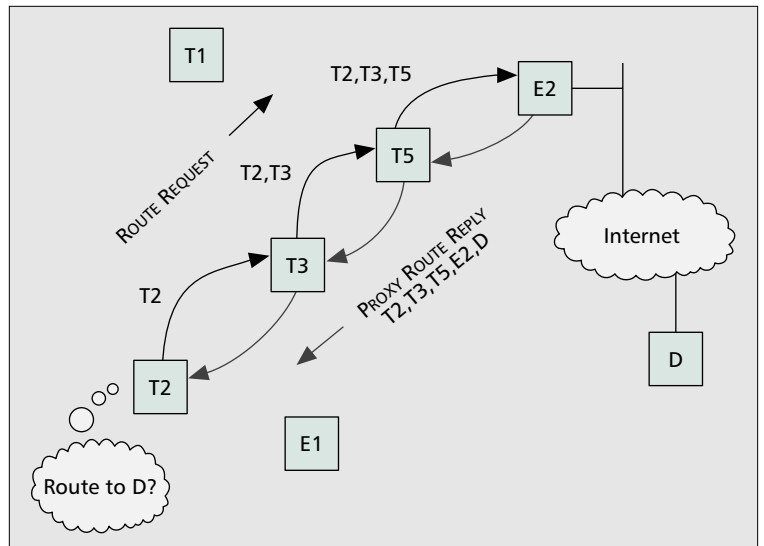
To perform Route Discovery, the source node S locally broadcasts a ROUTE REQUEST packet with the time-to-live field of the IP header initialized to 1. This type of ROUTE REQUEST is called a nonpropagating ROUTE REQUEST. It allows node S to inexpensively query the Route Caches of each of its neighbors for a route to the destination, and optimizes the case in which D is directly reachable. If no REPLY is returned within the nominal one-hop round-trip time, node S transmits a propagating ROUTE REQUEST that is flooded through the network in a controlled manner and is answered by a ROUTE REPLY packet from either node D or another node that knows a route to D.

Route Maintenance is performed only when a node is attempting to forward a packet. If the packet cannot be successfully forwarded to the next-hop indicated in the packet's source route, Route Maintenance declares that next link in the source route to be broken, and notifies the packet's originator S with a ROUTE ERROR packet. The originator S can then attempt to use any other route to D that is already in its Route Cache, or can invoke Route Discovery again to find a new route for subsequent packets.

## Integration with the Internet

We have extended the mechanisms of Route Discovery and Route Maintenance to support communication between nodes inside the ad hoc network and those outside in the greater Internet [11]. So that each node in the ad hoc network maintains a constant identity as it communicates with nodes inside and outside the network, we require that each node choose a single IP address, called its *home address*, by which it is known to all other nodes. This notion of a home address is identical to that defined by Mobile IP [9]. As in Mobile IP, each node is configured with its home address and uses this address as the IP source address for all of the packets it sends.

Figure 3 illustrates node T2 inside the ad hoc network dis-



**Figure 3.** ROUTE REQUEST for a node not in the ad hoc network being answered by the foreign agent.

covering a route to a node D outside the network. As the ROUTE REQUEST from T2 targeting D propagates, it is eventually received by the gateway node E2, which consults its routing table. If it believes D is reachable outside the ad hoc network, it sends a ROUTE REPLY listing itself as the second-to-last node in the route, and marking the reply such that T2 will recognize it as a proxy reply. If the target node D actually is inside the ad hoc network, node T2 will receive a ROUTE REPLY from both E2 and D. Since T2 can distinguish which replies are proxy replies, it can prefer the direct route when sending packets to D.

### Integration with Mobile IP

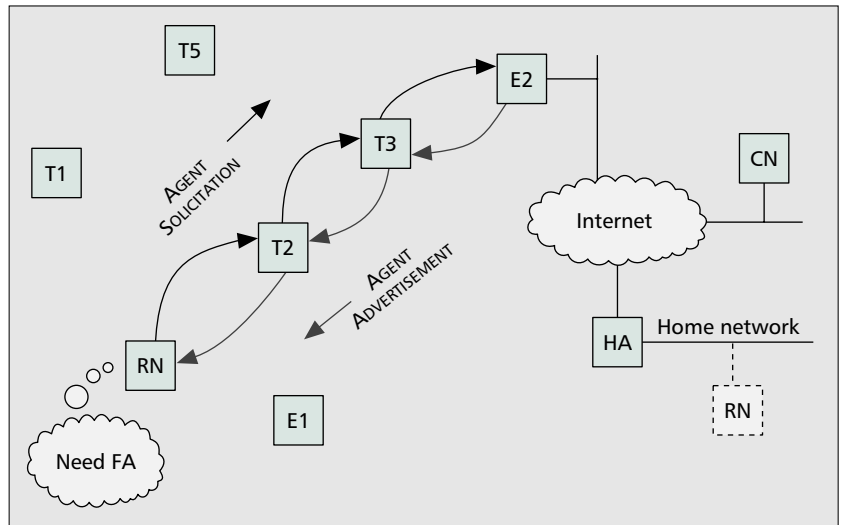
Since node RN in Fig. 1 must be able to participate in different IP subnets depending on its current location, it uses Mobile IP to connect to the Internet. Figure 4 shows how RN is homed in a subnet not belonging to the ad hoc network, but RN has wandered in range of the ad hoc network. As described earlier, node E2 provides Mobile IP foreign agent services, in addition to being configured as a gateway between the ad hoc network and the Internet.

As part of normal Mobile IP operation, RN periodically checks to verify that it is currently using the best means available to maintain connection with the Internet. We configured RN to operate in LAN mode as its top preference, to connect to the Internet via a DSR ad hoc network as its second choice, and to connect via CDPD when no other options are available. When RN receives DSR packets, such as ROUTE REQUESTS, ROUTE REPLYS, or data packets with DSR source routes on them, it knows it is within range of a DSR network and enables that option to Mobile IP.

If RN decides its best connectivity would be via the ad hoc network, it transmits a Mobile IP AGENT SOLICITATION piggybacked on a ROUTE REQUEST targeting the IP limited broadcast address (255.255.255.255). This allows the SOLICITATION to propagate over multiple hops through the ad hoc network, though gateways will not propagate it between subnets. When the foreign agent at E2 receives the SOLICITATION, it will reply with an AGENT ADVERTISEMENT, allowing RN to register itself with this foreign agent and with its home agent as a Mobile IP mobile node visiting the ad hoc network. Once the registration is complete, the mobile node's home agent will use Mobile IP to tunnel packets destined for the mobile node RN to the foreign agent at E2, and E2 will deliver the packets locally to the mobile node using DSR.

## Layer 3 Mechanisms for Acknowledgment and Retransmission

Since the WaveLAN-I radios do not provide link-layer reliability, we implemented a hop-by-hop retransmission and acknowledgment scheme within the DSR layer that provides the feedback necessary to drive DSR's Route Maintenance mechanism. One interesting aspect of our automatic repeat request (ARQ) scheme was the use of passive acknowledgments [12], which significantly reduces the number of acknowledgment packets transmitted when compared to acknowledgment schemes that acknowledge every packet (e.g., IEEE 802.11 [8]).



■ **Figure 4.** The roaming node registering with a foreign agent located on E2 in the ad hoc network.

### Implementation Overview

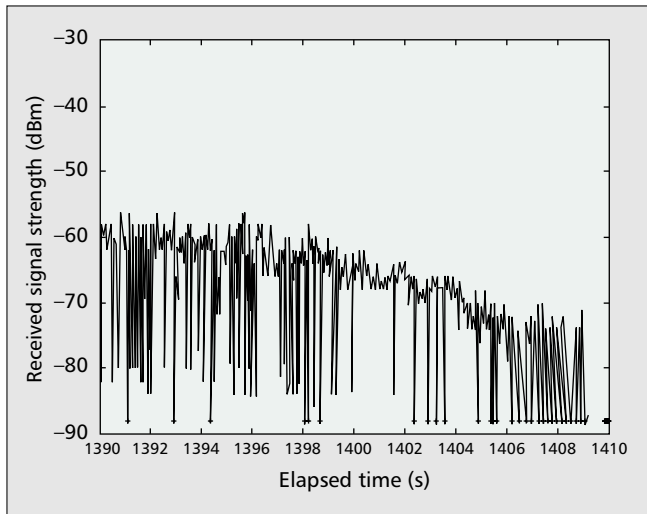
Our implementation utilizes passive acknowledgments whenever possible, meaning that if a node A originating or forwarding a packet hears the next-hop node B forward the packet, A accepts this as evidence that the packet was successfully received by B. If A fails to receive a passive acknowledgment for a particular packet it has transmitted to B, A retransmits the packet, but sets a bit in the packet's header to request an explicit acknowledgment from B. Node A also requests an explicit acknowledgment from B if B is the packet's final destination, since in this case A will not have the opportunity to receive a passive acknowledgment from B. To avoid the inefficiencies of a stop-and-wait ARQ scheme, node A uses a buffer to hold packets it has transmitted that are pending acknowledgment plus an identifier based on the IP ID field [13] to match acknowledgments with buffered packets.

This acknowledgment procedure allows A to receive acknowledgments from B even when the wireless link from A to B is unidirectional, since explicit acknowledgments can take an indirect route from B to A. During an average run, 90 percent of the acknowledgments used a direct one-hop route, and 10 percent were sent over routes with multiple hops. While this strongly suggests the presence of unidirectional links in the network, it does not support a conclusion that 10 percent of the packets travel over a unidirectional link. Once a multihop route for acknowledgments is discovered, it may continue to be used for some period of time even after the direct route begins working again.

When performing retransmissions at the DSR layer, we also found it necessary to perform duplicate detection so that when an acknowledgment is lost, a retransmitted packet is not needlessly forwarded through the network multiple times. The duplicate detection algorithm used in our implementation specified that a node should drop a received packet if an identical copy of the packet was found in a buffer awaiting either transmission or retransmission. We found that this simple form of duplicate suppression was sufficient, and that maintaining a separate history of recently seen packets was not necessary.

### Heuristics for Selecting Timeout Values

Early in the design of our retransmission mechanism, we found that contention for the wireless medium produced enough variance in the round-trip time (RTT) between neighboring nodes that using a fixed value for the retransmission timer was not practical, and *adaptive* retransmission timers were required.



■ **Figure 5.** Received signal strength of packets sent directly between two nodes during a full run of the testbed.

Our initial implementation of an adaptive retransmission timer employed the scheme used by TCP, where a smoothed RTT estimator ( $srtt$ ) and a smoothed mean deviation ( $rttvar$ ) are maintained independently for each next hop to which a node is communicating. The retransmission timeout (RTO) is then computed as

$$RTO = srtt + 4 \times rttvar$$

Unfortunately, the variance in RTT prevented this implementation from performing adequately. Frequently, the RTO would not adapt quickly enough to congestion in the network, causing packets to be retransmitted unnecessarily and creating even more congestion. It also suffered from the fact that the RTO to each next hop was computed independently, while the need to defer transmissions due to congestion is common across all neighbors accessed via the same network interface.

We found that several simple methods of reacting to increasing congestion did not work. For example, we tried an algorithm that feeds into the smoothing function for RTT estimation an RTT sample of twice the current smoothed RTT estimate whenever there is a timeout. This algorithm causes the value of the RTT estimator, and hence the retransmission timer, to tend to diverge and remain pegged at its maximum value, even after congestion has subsided.

We developed a successful retransmission timer algorithm by including a heuristic estimate of the level of local congestion so that the retransmission timer could react quickly to changes. One of the simplest ways for a node to measure congestion in the network is to look at the length of its *own* network interface transmit queue. Specifically, if more than five packets are found in the interface transmit queue — suggesting that congestion is starting to occur — we increase the value of the retransmission timer 20 ms for each packet in the queue. Assume that there are  $N$  packets in the network interface queue. For  $N \leq 5$ , the retransmission timeout is computed as before:

$$RTO = srtt + 4 \times rttvar.$$

However, for  $N > 5$ , the retransmission timeout is computed as

$$RTO = srtt + 4 \times rttvar + (N \times 20 \text{ ms}).$$

This heuristic allows the retransmission timer to increase quickly during periods of congestion and then return just as quickly to its computed value once the congestion dissipates. In 4710 measurements over several runs, approximately 75 percent of the packets trans-

mitted use the minimum retransmission timer value of 50 ms. However, for the other 25 percent, the retransmission timer adjusted itself to values between 60 and 920 ms. The wide range indicates that an adaptive retransmission scheme is required for good performance if acknowledgments are implemented above the link layer.

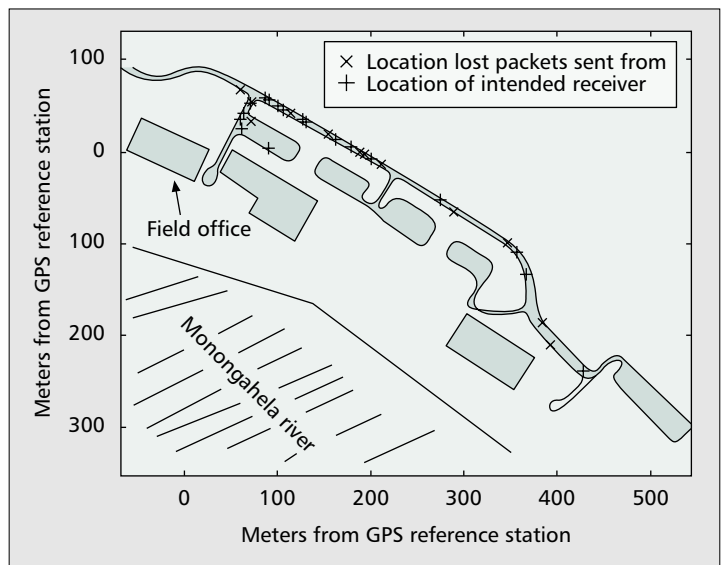
## Wireless Propagation

When our testbed network operated without the layer 3 acknowledgment and retransmission scheme, the average packet loss rate over a single hop was measured as 11 percent. With the ARQ scheme described earlier, the average loss rate over a single hop dropped to 5 percent. The losses are highly correlated with position, but also demonstrate the highly variable nature of wireless propagation due to scattering, multipath, and shadowing effects in the real world.

### Small- and Large-Scale Fading

As an example, Fig. 5 shows the signal level at which packets were received when sent by node T4 directly over one hop to T3. This data was measured during a full run of the testbed, so all the nodes (T1–T5) were in motion and exchanging data when it was recorded. Dropped packets are marked with a + and shown at  $-90$  dBm; the actual value of the received signal strength is unknown because the packets were not received. From 1390 to 1404 s the figure shows most packets being successfully delivered, but with significant numbers of multipath fades of 20 dB or greater. Scattered among these are packet losses, most of which the link-layer retransmission algorithm is able to correct. However, given the impracticality of predicting such losses, link-layer retransmission is the only defense against unrecoverable packet loss. The variability in propagation creates a significant level of inherent packet loss with which higher layers must be prepared to cope.

During the period from 1405 to 1410 s, however, Fig. 5 shows what can be best described as a routing protocol error. Although other paths were available, the protocol continued to send packets directly from T4 to T3 when nearly half those packets were being lost. As discussed later, today's routing protocols must be extended in at least one of two directions. They



■ **Figure 6.** The location of the sending node and receiving node when more than 1 percent packet loss occurred; nodes maintained 10 s separation (90 m).

must predict the continued decrease in average signal level such as occurred from 1400 to 1406 s and switch to a new route, or maintain state to record that a particular link is becoming lossy and avoid it until such time as it functions well again.

### Correlation of Location with Packet Loss

As part of conducting an initial survey of the site, we found it particularly helpful to obtain a rough characterization of the site's propagation environment. We had two cars drive the course at about 30 km/h (20 mi/h), one following the other at a separation of 90 m, with the trailing car transmitting 1024-byte packets to the lead car 10 times/s. The cars made three laps of the course, a total driving time of about 660 s.

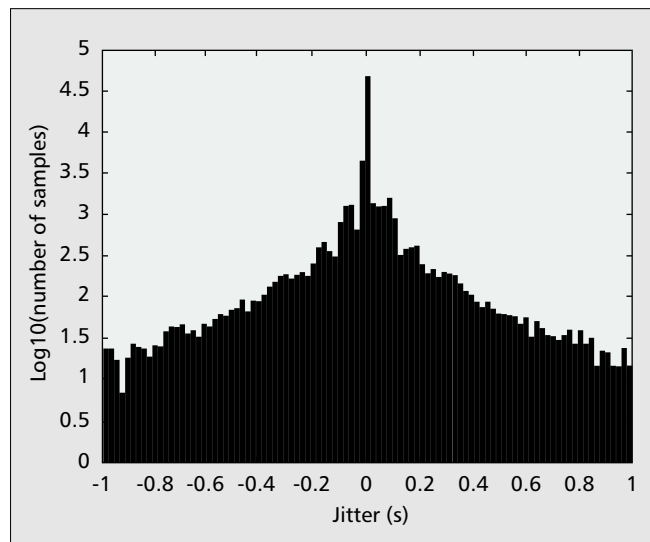
For a separation distance of 90 m, Fig. 6 shows the position of the cars during the intervals when more than 1 percent of the total packet loss occurred. This separation is well less than the nominal WaveLAN-I range of 250 m. Essentially all of the loss bursts occurred while the nodes were on the straightest part of the course with clear line of sight to each other. There is no elevation change along that portion of the course, and the parking lots along the south side of the road were empty during the test. While we have been unable to completely determine the cause of these losses, our current hypothesis is that the radios are suffering from multipath reflection off cars unrelated to the testbed traveling on a road approximately 20 ft north of the course.

### Jitter in Interpacket Spacing

Isochronous communications, such as interactive voice and video, are extremely sensitive to packet jitter. In order to evaluate how well isochronous communication works across our testbed network, we evaluated the jitter experienced by the synthetic audio traffic — part of the composite workload described in Table 1. Each audio connection consisted of alternating simplex packet streams between the communicating parties, and each packet stream consisted of 250-byte UDP packets sent 8 times/s, giving an average bit rate of 16 kb/s. This traffic pattern was chosen to model the push-to-talk mobile radios commonly used on mine and construction sites.

During an average run, a total of 98,000 voice packets are originated, which represents 3.4 h of voice communication. Of these 98,000 packets, 3.8 percent are lost in the network. The testbed does not contain any special handling rules for the voice packets, so the packets are retransmitted according to the same mechanism described earlier. The jitter, defined as the variation in interpacket spacing introduced by the network, ranged from  $-9.4$  to  $6.5$  s. The extreme values of jitter are rare, and typically occur when the network is temporarily partitioned. During a partition, the voice sources continue to send data, but the packets are buffered inside the network. The result is a burst of back-to-back packet arrivals at the destination when the partition heals. As an area for future research, more sophisticated packet handling algorithms might detect these delayed but time-sensitive packets and drop them inside the network to conserve resources [10].

When the most extreme 2 percent of jitter samples are removed as outliers, the range of jitter drops to between  $-1.04$  and  $1.02$  s. The mean jitter is  $0.001$  s, and the standard deviation  $0.143$  s. Figure 7 shows the distribution of jitter samples using a histogram. The y-axis is on a log scale for clarity; each bar is 20 ms wide, and there are 10 times more packets with  $0.0$  s of jitter than packets with either  $\pm 20$  ms of jitter. Ninety percent of the voice packets experience jitter between  $-0.2$  and  $0.2$  s, so a playback buffer of 400 ms should be sufficient for voice communication.



■ Figure 7. Distribution of jitter; Y-axis is on a log scale for clarity.

## The Need for Hysteresis in Route Selection

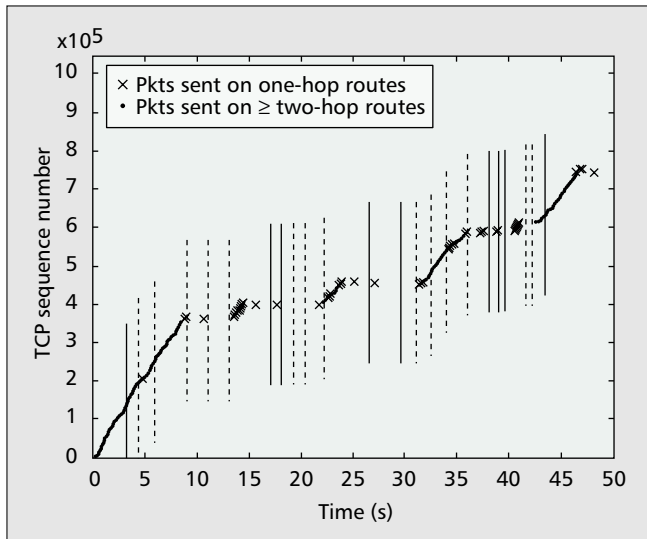
As mentioned above, the packet loss rate seen between any two nodes in the network is highly variable, depending not only on the positions of the nodes involved, but also on the movement of other objects around the nodes. In working with TCP connections carried over the testbed network, we found this variability had a disastrous effect on the bandwidth delivered by these TCP connections. Other researchers have addressed related problems in layer 4 and at the boundary between layer 3 and layer 4 [14, 15], so in this section we will concentrate on a layer 3 issue caused when radio propagation is transiently *better* than expected.

To isolate the layer 3 issue, we conducted an experiment with three nodes arranged in a line. The nodes were positioned by driving two cars in opposite directions and positioning them as far from the middle node as possible, while still allowing both of the end nodes to successfully flood ping the intermediate node with 1024-byte packets. The flood ping tests were carried out serially, meaning that only one node was flood pinging at a time, and a successful flood ping test was defined as sending ping packets as fast as possible with more than 99 percent of the ping ECHO REPLYs being received over a 10 s sample. Once positioned, the nodes remained stationary for the remainder of the test. For the purpose of discussion, let node A be the TCP source, B the intermediate node, and C the TCP sink.

This is a particularly challenging scenario, not only because electromagnetic propagation is highly variable, but because the specific setup of this test deliberately introduces the hidden terminal problem. A number of times during these tests, we saw the DSR retransmission timer expire, creating ROUTE ERRORS and subsequent ROUTE REQUESTS as the nodes attempted to restore connectivity.

As described earlier, nodes using DSR discover routes to other nodes by first sending a non-propagating ROUTE REQUEST that the target node will answer with a ROUTE REPLY if it can directly receive the originator's REQUEST. If the originator does not receive a ROUTE REPLY within 30 ms, it sends a propagating ROUTE REQUEST that floods through the network in a controlled fashion to discover multihop routes to the target.

In order to obtain baseline performance metrics, we used our `macfilter` tool [1], which allows us to create a synthetic



**Figure 8.** A TCP sequence number plot for a 1 Mbyte transfer over a two-hop route. The vertical lines indicate the times at which the TCP source initiated Route Discovery; the dashed lines indicate the times at which only a nonpropagating ROUTE REQUEST was transmitted, and the solid lines indicate both a nonpropagating and a propagating ROUTE REQUEST.

propagation environment among nodes in a laboratory setting. We first made five independent TCP transfers of 1 Mbyte each from node A to node C. Over these five transfers, TCP averaged 0.50 Mbyte/s (61 kbytes/s) with a standard deviation of 0.079 Mb/s. When the same transfers were performed in the field, however, the average data rate was 0.12 Mbyte/s (14.65 kbytes/s) with a standard deviation of 0.025 Mbyte/s — only 25 percent of the throughput measured in the laboratory. In fact, some of the 1 Mbyte data transfers, which were set up to last for a maximum of 50 s, timed out before the entire megabyte could be transferred. In these cases, we report the average data rate for the 50-s duration of the connections.

The time-sequence number plot from a typical two-hop connection in the testbed is depicted in Fig. 8. Sequence numbers marked with a small dot were transmitted using the two-hop route  $A \rightarrow B \rightarrow C$ , whereas sequence numbers marked  $x$  were transmitted using the one-hop route  $A \rightarrow C$ . The dashed vertical lines in the figure indicate when the TCP source A performed a Route Discovery consisting only of a nonpropagating ROUTE REQUEST, and the solid vertical lines indicate when a Route Discovery consisting of both a nonpropagating and a propagating ROUTE REQUEST occurred (by the rules of Route Discovery, if the nonpropagating REQUEST returns a REPLY, the propagating REQUEST is not sent).

The TCP connection in Fig. 8 made very good progress for the first 9 s of the connection, using almost exclusively a two-hop route. However, during the time interval from 9 to 22 s, the connection makes almost no progress, sending about 30 kbytes in this 13 s interval. After processing a ROUTE ERROR at 9 s, the TCP source (node A) initiates a Route Discovery. The nonpropagating ROUTE REQUEST is answered directly by node C, causing A not to send a subsequent propagating ROUTE REQUEST and thus to use a single-hop route to node C. The poor quality of this single-hop link leads to repeated errors and Route Discovery attempts. Finally, at 18 s, node A's nonpropagating ROUTE REQUEST fails to return any REPLYS, so A transmits a propagating REQUEST. This results in the discovery of both the single-hop route and the two-hop route through intermediate node B. By this time, TCP has

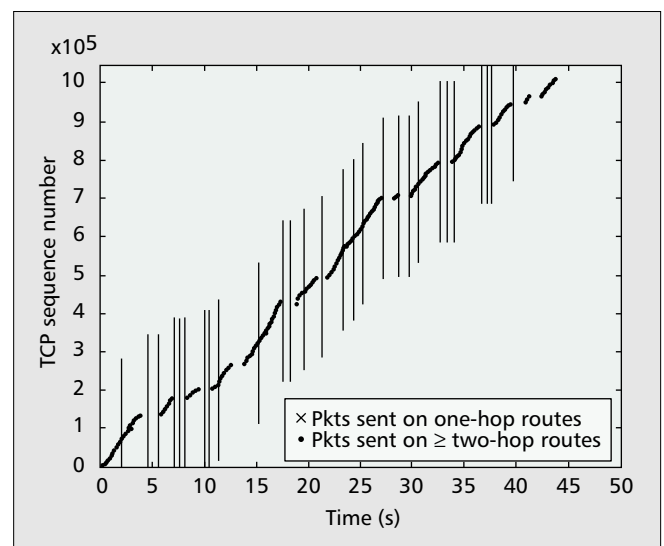
backed off and does not offer the next packet to the network until 22 s. Node A attempts to use the one-hop route it discovered, finds that it does not work well, removes the one-hop route from its Route Cache, and begins using the two-hop route. At this time, the connection again starts making progress. The same scenario of repeated attempts to use a one-hop route occurs again during the intervals 25–32 s and 35–43 s.

This scenario illustrates an important challenge for ad hoc network routing protocols and argues strongly that all routing protocols need some ability to remember which recently used routes have been tried and found not to work. Even traditional distance-vector-style protocols are subject to this problem since they attempt to minimize a single metric, usually hop count.

For example, if A, B, and C in the three-node scenario discussed above were all using a distance vector routing protocol, A would sometimes hear advertisements from C. Since the direct route to C is more optimal in terms of hop count than the route through the intermediate node B, A would attempt to send all of its packets directly to C until that direct route timed out. In other words, without some type of local feedback or other hysteresis, A will often try to send its packets directly to C, effectively black-holing most of these packets since that link is so unreliable. Protocols such as Signal Stability Based Routing (SSA) [16] may behave much better in this scenario.

To evaluate the potential gain of having a mechanism that would prevent the repeated use of the poor direct route from A to C, we emulated perfect routing information by using our `macfilter` to eliminate the discovery of single-hop routes. Figure 9 shows the time-sequence plot for a 1 Mbyte transfer in the field using this “perfect routing.” The flat plateaus are no longer present, and the throughput is 30 percent higher. The remaining Route Discoveries are triggered when packets are repeatedly lost due to variation in wireless propagation, and techniques such as notifications to the TCP module could be used to prevent the TCP stalls that follow these Route Discoveries.

We are presently considering three ways to implement



**Figure 9.** A TCP sequence number plot for a 1 Mbyte transfer over a two-hop route when the `macfilter` utility was used on both the source and destination nodes to prevent the use of single-hop routes. The vertical lines indicate the times at which the TCP source initiated Route Discovery.

such a mechanism in DSR [4]. One solution would be for DSR to cache information about each link for which it receives a ROUTE ERROR. This negative information could be timed out after a period based on the estimated rate of link fluctuation, but would prevent DSR from repeatedly attempting to use a poor-quality link. The drawback of this solution is the difficulty of designing a strategy to pick a reasonable timeout value.

While not a generic technique, in networks where each node knows its position (e.g., due to the use of GPS as in our network), communicating nodes could use the location information propagated by the other nodes to model the position of their correspondents. If the link A to C is found to be bad, DSR could retain that negative information in its cache until it finds that either node A or C has changed position in some reasonably significant way.

The third and more sophisticated approach would combine the signal strength at which the node received ROUTE REPLYs, the position of the nodes, and the mobility pattern of the nodes to estimate the probability of successful communication over a particular route [17].

## Conclusions

We have created a testbed for ad hoc network research, featuring two stationary nodes, five car-mounted nodes that drive around the testbed site, and one car-mounted roving node that enters and leaves the site. Packets are routed between the nodes using the DSR protocol, which also seamlessly integrates the ad hoc network into the Internet via a gateway. The use of Mobile IP permits nodes to roam transparently between the ad hoc network and normal IP subnets.

In preliminary analysis of data from runs of the testbed, we have measured the jitter introduced by the network and found that even under a full load and without any special quality of service handling, the network can support a push-to-talk voice system. We have also explained how our novel heuristic for the DSR-layer packet acknowledgment and retransmission scheme allows nodes to rapidly adapt to changing levels of network congestion. Finally, we have demonstrated the need for hysteresis in the selection of routes that are used in ad hoc networks to prevent the use of routes that exist only transiently.

## Acknowledgments

The Monarch Project ad hoc network testbed was the product of the work of many people, but special recognition is due to Jorjeta Jetcheva, Qifa Ke, and Ben Bennington. We are also grateful for the efforts of the other members of the research team, including Ratish Punnoose, Pavel Nikitin, Dan Stancil, Satish Shetty, Michael Lohmiller, Yih-Chun Hu, Sam Weiler, and Jon Schlegel.

## References

- [1] D. A. Maltz, J. Broch, and D. B. Johnson, "Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed," Tech. rep. CMU CS 99-116, School of Comp. Sci., Carnegie Mellon Univ., Mar. 1999; <http://www.monarch.cs.cmu.edu/papers.html>
- [2] D. B. Johnson *et al.*, "Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet draft, draft-ietf-manet-dsr-04.txt, Nov. 2000, work in progress.
- [3] D. B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proc. IEEE Wksp. Mobile Comp. Sys. and Apps.*, Dec. 1994, pp. 158-63.
- [4] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, T. Imielinski and H. Korth, Eds., Ch. 5, Kluwer, 1996, pp. 153-81.
- [5] D. A. Maltz *et al.*, "The Effects of On-Demand Behavior in Routing Protocols for Ad Hoc Networks," *IEEE JSAC*, vol. 17, no. 8, Aug. 1999, pp. 1439-53.

- [6] D. A. Beyer, "Accomplishments of the DARPA Survivable Adaptive Networks SURAN Program," *Proc. MILCOM '90*, 1990.
- [7] R. E. Kahn *et al.*, "Advances in Packet Radio Technology," *Proc. IEEE*, vol. 66, no. 11, Nov. 1978, pp. 1468-96.
- [8] IEEE Comp. Soc. LAN MAN Std.s Comm., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-1999, 1999.
- [9] C. Perkins, Ed., "IP Mobility Support," RFC 2002, Oct. 1996.
- [10] D. A. Maltz, "Resource Management in Multi-Hop Ad Hoc Networks," Technical Report CMU CS TR00-150, School of Comp. Sci., Carnegie Mellon Univ., Nov. 1999; <http://www.monarch.cs.cmu.edu/papers.html>
- [11] J. Broch, D. A. Maltz, and D. B. Johnson, "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks," *Proc. Int'l. Symp. Parallel Architecture, Algorithms, and Networks*, Perth, Australia, June 1999, pp. 370-75.
- [12] J. Jubin and J. D. Tornow, "The DARPA Packet Radio Network Protocols," *Proc. IEEE*, vol. 75, no. 1, Jan. 1987, pp. 21-32.
- [13] J. Postel, "Internet Protocol," RFC 791, Sept. 1981.
- [14] B. S. Bakshi *et al.*, "Improving the Performance of TCP over Wireless Networks," *Proc. 17th Int'l. Conf. Dist. Comp. Sys.*, May 1997, pp. 365-73.
- [15] G. Holland and N. Vaidya, Analysis of TCP Performance over Mobile Ad Hoc Networks, *MobiCom '99*, 1999, pp. 219-30.
- [16] R. Dube *et al.*, "Signal Stability Based Adaptive Routing for Ad Hoc Mobile Networks," *IEEE Pers. Commun.*, Feb. 1997, pp. 36-45.
- [17] R. J. Punnoose *et al.*, "Optimizing Wireless Network Protocols Using Real-Time Predictive Propagation Modeling," *RAWCON*, Denver, CO, Aug. 1999.

## Biographies

DAVID A. MALTZ [M] ([dave@aonnetworks.com](mailto:dave@aonnetworks.com)) is currently on the founding team of AON Networks. Previously, he was a network architect at FreeSpace Communications and a senior graduate student in the Mobile Networking Architectures (Monarch) Project at Carnegie Mellon University. His recent research work includes the design, implementation, and evaluation of the DSR protocol and TCP Splice, a technique that improves the performance of network proxies by an order of magnitude. He has been a team leader and key implementer in building deployed ad hoc networks, developing systems for network emulation, and researching Mobile IPv4. He is expecting to receive his Ph.D. in computer science from Carnegie Mellon University in May 2001. He earned S.M. and S.B. degrees in electrical engineering and computer science from MIT in 1994.

JOSH BROCH [M] ([josh@aonnetworks.com](mailto:josh@aonnetworks.com)) is presently a founding member of AON Networks, a Palo Alto based startup company. During his four years at Carnegie Mellon University (1995-1999), he authored the first implementation of Mobile IP for IPv6, and, working in concert with David A. Maltz, authored a leading IETF proposal for multihop routing in ad hoc wireless networks, designed one of the first realistic simulation systems for wireless ad hoc networks, and led a group in the design, implementation, and deployment of a full-scale ad hoc network system. Prior to CMU, he spent five years doing network design and implementation work at Connections for Business, Hollywood, Florida. He is presently a Ph.D. candidate in the electrical and computer engineering department at Carnegie Mellon University. He obtained an M.S. in information networking from CMU in 1996 and a B.S. in mathematics and computer science from Barry University in 1995.

DAVID B. JOHNSON [M] ([dbj@cs.rice.edu](mailto:dbj@cs.rice.edu)) is an associate professor of computer science and electrical and computer engineering at Rice University, and is a member of Rice's Computer Systems Laboratory and Center for Multimedia Communication. His research interests include network protocols, distributed systems, and operating systems. Prior to joining the faculty at Rice in 2000, he was an associate professor of computer science and electrical and computer engineering at Carnegie Mellon University, where he was on the faculty for eight years. He received his B. A. degree in computer science and mathematical sciences in 1982, his M. S. degree in computer science in 1985, and a Ph.D. degree in computer science in 1990, all from Rice University. He is leading the Monarch Project at Rice University and Carnegie Mellon University, developing adaptive networking protocols and architectures to allow truly seamless wireless and mobile networking. Related to this research, he has also been active in the Internet Engineering Task Force (IETF), the principal protocol standards development body for the Internet, where he was one of the main designers of the IETF Mobile IP protocol for IPv4 and is the primary designer of Mobile IP for IPv6. He is an editor for *ACM/Baltzer Mobile Networks and Applications* (MONET), *ACM/Baltzer/URSI Wireless Networks* (WINET), *IEEE/ACM Transactions on Networking*, and *ACM SIGMOBILE Mobile Computing and Communications Review*; and has served as a member of the technical program committee or executive committee for over 25 international conferences and workshops. He is an Executive Committee member and Treasurer for ACM SIGMOBILE, and is a member of the ACM, USENIX, Sigma Xi, and the Internet Society.