Outdoor Experimental Comparison of Four Ad Hoc Routing Algorithms*

Robert S. Gray, David Kotz, Calvin Newport, Nikita Dubrovsky,
Aaron Fiske, Jason Liu, Christopher Masone, Susan McGrath, and Yougu Yuan

Dartmouth College, Hanover, NH 03755

Dartmouth College Computer Science Technical Report TR2004-511

June 18, 2004

Abstract

Most comparisons of wireless ad hoc routing algorithms involve simulated or *indoor* trial runs, or outdoor runs with only a small number of nodes, potentially leading to an incorrect picture of algorithm performance. In this paper, we report on the results of an outdoor trial run of four different routing algorithms, APRL, AODV, GPSR, and STARA, running on top of thirty-three 802.11-enabled laptops moving randomly through an athletic field. The laptops generated random traffic according to the traffic patterns observed in a prototype application, and ran each routing algorithm for a fifteen-minute period over the course of the hour-long trial run. The 33-laptop experiment represents one of the largest outdoor tests of wireless routing algorithms, and three of the algorithms each come from a different algorithmic class, providing insight into the behavior of ad hoc routing algorithms at larger real-world scales than have been considered so far. In addition, we compare the outdoor results with both indoor ("tabletop") and simulation results for the same algorithms, examining the differences between the indoor results and the outdoor reality. The paper also describes the software infrastructure that allowed us to implement the ad hoc routing algorithms in a comparable way, and use the *same* codebase for indoor, outdoor, and simulated trial runs.

1 Introduction

Ad hoc wireless networks can provide connectivity in a variety of environments in which traditional communication infrastructures are absent. One notable environment is the modern battlefield [Gra00]. Although some urban locations may have cell phone or other communication networks, these systems may be destroyed, non-operational or inaccessible under battle conditions. Therefore, warfighters must rely on communication mechanisms that they can carry with them, and the military is actively pursuing soldier-mounted ad hoc networks for data communications. In the US Army's vision of the Future Force Warrior, [1] for example, soldiers will have transmission equipment that will make each person a node in an ad hoc network. [2] The application of ad hoc

* Supported under Contract Number F49620-93-10266 from the Department of Defense (DoD) and Air Force Office of Scientific Research (AFOSR), with additional support from Dartmouth College, DARPA under Contract Number F30602-98-2-0107, and the Department of Homeland Security (DHS) under Contract Number Number 2000-DT-CX-K001 (S-2). Points of view in this document are those of the authors and do not necessarily represent the official position of the DoD, AFOSR, DARPA, DHS or Dartmouth College.

[1] http://www.natick.army.mil/soldier/WSIT/

[2] http://www.mit-kmi.com/archive_article.cfm?DocID=14

networks between vehicles is also possible, and, in fact, the United States military currently is using this technology in Iraq. [3] Similarly, ad hoc networks can be used to expedite response in civil emergencies, as in our own work on remote triage for casualties [WMB03].

Many other applications of ad hoc networks exist, such as urban rooftop networks or land- or ocean-based sensor networks, but we focus on the battlefield and emergency-response scenarios in our research work. More specifically, in this paper, we focus on routing algorithms for *outdoor*, *mobile* ad hoc networks. Unfortunately, although there have been a few outdoor tests with a large number of stationary sensors, there have been few, if any, outdoor tests with a large number of mobile laptops or other personal computing devices. Most outdoor tests have involved only a few nodes. Moreover, results obtained from a small number of real nodes or a large number of *simulated* nodes do not necessarily generalize to a large number of real nodes, potentially leading to an incorrect view of routing-algorithm performance.

This paper partially addresses this gap by analyzing the results of an outdoor trial run of four different routing algorithms, APRL, AODV, GPSR, and STARA, running on top of thirty-three 802.11-enabled laptops moving randomly through an athletic field. The laptops generated random traffic according to the traffic patterns observed in a prototype battlefield application, and ran each routing algorithm for a fifteen-minute period over the course of the one-hour trial run. This 33-laptop experiment represents one of the largest outdoor tests of ad hoc routing algorithms, and possibly represents the largest outdoor test for which results are publicly available. In addition, we compare these outdoor results with both indoor ("single shelf") and simulation results for the same algorithms. Finally, the paper also describes the software infrastructure that allowed us to implement the ad hoc routing algorithms in a comparable way, and use the *same* codebase for indoor, outdoor, and simulation trial runs.

With this analysis, the paper accomplishes three goals. First, three of the routing algorithms each come from a different algorithmic class, and the outdoor experiment therefore provides insight into how these classes of algorithms behave at larger real-world scales than have been considered so far. Second, the comparison with indoor and simulated results confirms again that real-world performance is difficult to predict without real-world experiments, but also confirms that it is possible to model radio and other behaviors accurately enough for simulated results to be meaningful. Finally, the implemented software infrastructure provides an effective starting point for further large-scale, outdoor routing experiments, either with one of our four implemented algorithms or with another algorithm.

Section 2 describes the four routing algorithms that we considered, and Section 3 presents our experimental infrastructure. Section 4 analyzes the results of the outdoor, indoor and simulation experiments, considering the suitability of specific algorithms for the outdoor environment, and examining whether indoor experiments or simulation can predict the outdoor results. Finally, Section 5 summarizes related work, and Section 6 concludes with the major lessons learned and important areas of future work.

2 Routing Algorithms

The experiments involved four routing algorithms, Any Path Routing without Loops (APRL) [KK98], Ad hoc On-demand Distance Vector (AODV) [PR99], On-Demand Multicast Routing Protocol (ODMRP) [LSG02], and System- and Traffic-dependent Adaptive Routing Algorithm (STARA) [GK97, Gup00]. APRL and STARA were developed at Harvard and the University of Illinois as part of

[3] http://lists.personaltelco.net/pipermail/general/2003q2/012362.html

a Dartmouth-led Multi-disciplinary University Research Initiative (MURI) on field communications for military personnel, while AODV and ODMRP are algorithms that are extensively used as comparison baselines for other routing algorithms (and that have seen some use in military applications). Although APRL and STARA are not standard choices, we feel that our selection is representative of an important subsection of the ad hoc routing space, simple (APRL) and complex (STARA) pro-active routing protocols, and two related reactive protocols (AODV and ODMRP). In this section, we provide a general overview of how the four algorithms work. We save a discussion of specific parameters, such as time out values, for the experiment section.

2.1 APRL and STARA

APRL and STARA are both pro-active routing algorithms. APRL is quite simple in that it only tries to discover a fixed number of routes, not necessarily shortest, from one node to another, while STARA is quite complex in that it uses dynamic latency measurements to select from multiple available routes.

With APRL [KK98], each node broadcasts routing beacons at a fixed frequency, and each such beacon contains a complete copy of the node's current routing table. Neighboring nodes use the embedded routing information to update their own routing table, and then propagate the updated information further as part of their own beacons. Initially, the routing tables are empty, and a beacon serves only to tell one node that it has another node as a neighbor. The receiving node, however, then will be able to include the existence of the sending node in its own beacons. Eventually every node in the network will learn about a route to every other node, the exact amount of time depending on the diameter of the network and the beaconing interval. Nodes use whatever route they learn about first, without regard to the route's length or quality, but routes will time out if they are not refreshed by a beacon within a specific time interval. In addition, APRL will record a configurable number of alternate routes to each destination, and will switch to an alternate route as soon as a primary route times out.

The beaconing approach described so far can lead to loops in the routing topology since routes on different nodes will not time out at the same time. Suppose, for example, that the current routes to A are linear and go from node D to node C to node B to node A. In the worst case, node A moves out of range of the other nodes, node B's route to A times out, and node B immediately hears a beacon from D containing D's route to node A. Node B now will believe that it can send its node-A traffic through node D, resulting in a three-node loop for all traffic destined for node A. To address this problem, APRL does not use a route as soon as it learns about it from a beacon, but instead sends a probe packet along the route. The destination node, if the packet makes it that far, sends the probe packet back to the sender along the reverse route. The sender will start using the route only if the probe packet successfully reaches the destination and returns. If a node has an alternate route, the node will switch to the alternate route when the primary fails, but will send out a probe packet to confirm that the alternate route still is valid.

STARA takes a different approach than APRL, and tries to accurately estimate each route's latency so that it can choose the best route among many [GK97, Gup00]. STARA uses latency, rather than hop count, as a metric, since interference from traffic or environmental features can cause a short-hop route to have a significantly longer end-to-end delay than a long-hop route.

STARA maintains a list of neighbors that are in immediate transmission range of a particular node, and updates that list through the periodic broadcast of Neighborhood Probe (NP) packets and Neighborhood Probe Acknowledgment Packets (NP_ACK). When sending a packet to a non-neighbor, STARA probabilistically chooses a neighbor through which to route the packet. Initially, the probability distribution is uniform, but STARA will adjust the probabilities according to its

latency observations. Specifically, just before STARA sends a Data Packet (DP), it attaches a current timestamp to the packet. When the data packet reaches its destination, the destination node sends a DP_ACK back to the original machine; this DP_ACK contains the original timestamp along with a new timestamp indicating the time that the packet was received. Once the DP_ACK has returned to the original node, the route delay is calculated simply as the difference between the two timestamps. STARA does not require the clocks on the nodes to be synchronized, since the *difference* between delays to the same destination is independent of clock skew. As STARA receives delay information, it adjusts the probability distribution to be proportional to the observed delays, and distributes traffic along available routes that have approximately the same minimal delay (to reduce packet re-sequencing requirements).

One problem with STARA is that routes with large delays are seldom (if ever) used, which means that their delay estimates are updated infrequently or not at all. Thus, an improvement in the delay of a formerly slow path may go unnoticed. To avoid this, STARA periodically sends a Dummy Data Packet (DDP) along those routes for which a large time has elapsed since the last update of their delay estimates. The developer of STARA presents a simple version of DDPs in [GK97], which sends every DDP as a separate network packet, and a more complex version in [Gup00], which condenses multiple dummy data packets into a single network packet. We implemented the simpler version, and as we will see, the simpler version overloads our real network. The more complex version could be expected to have significantly better performance since it reduces control traffic by an order of magnitude or more. Thus, our STARA results should *not* be taken as an overall view of STARA's performance, but instead should serve as another demonstration of how a single detail in control-packet handling can undo the performance gains expected from a more complex algorithm. To emphasize this point, we will refer to our STARA implementation as STARA-SIMPLE or STARA-S in the rest of this paper.

2.2 AODV and ODMRP

AODV and ODMRP are closely related re-active routing algorithms, differing mainly in ODMRP's support for multicast traffic and its inclusion of data packets inside route-discovery packets.

Like any reactive protocol, AODV searches for a route only when a node has data traffic that it needs to send [PR99]. Specifically, when a node needs a route to a destination for which it does not have a route (or for which the route has expired), AODV broadcasts a Route Request (RREQ) message. [4] All nodes that receive the RREQ message, but do not have a route to the requested destination, rebroadcast the RREQ, while at the same time establishing a reverse route to the source node. When the RREQ reaches an intermediate node that has a sufficiently recent route to the destination (as determined by sequence numbers associated with routing entries) or reaches the destination node itself, the intermediate or destination node sends a Route Response (RREP) message back to the source node along the reverse route. [5] Once the original host receives the RREP message, it records and begins using the new route.

All nodes keep track of their neighbors through regularly broadcast HELLO messages. When a host detects a missing neighbor through a lack of HELLO messages, it invalidates the routes that used that neighbor as the gateway and broadcasts a Route Error (RERR) message that includes a list of all destinations that are now unreachable due to the failed link. Nodes that receive an RERR message will invalidate their own routes, and broadcast their own RERR messages. Any node that

[4] As an optimization, AODV can use an expanding ring search method for broadcasting RREQs, but we do not implement this capability in our version.

[5] An intermediate host also will send an RREP packet to the destination of the RREQ to make sure that the destination still has a route to the source.

still needs to communicate with one of the unreachable destinations will search for an alternative route through the RREQ/RREP process.

ODMRP is a multicast ad hoc routing protocol that, unlike many other multicast protocols, does not require a separate underlying unicast protocol. ODMRP establishes and maintains routes in much the same manner as AODV. When a node has a data packet that it needs to send, but has no route to the destination, ODMRP broadcasts a JoinQuery packet. In contrast to AODV, however, ODMRP embeds the data packet inside the JoinQuery, so that the data packet reaches the destination as soon as the JoinQuery does. A node that receives a new JoinQuery rebroadcasts it, and if the node wants to receive packets on the multicast IP address of the JoinQuery, it sends a JoinReply packet back to the source using the reverse route. Intermediate hosts receiving the JoinReply set a forwarding flag for the corresponding multicast group in a "forwarding group" table, indicating that they are on the path between the source and one or more of the destinations and thus should rebroadcast any future incoming data packets. A forwarding group has a preset expiration timeout, and ODMRP refreshes the group at regular intervals by resending JoinQuery packets. ODMRP only resends JoinQuerys so long as there still are data packets that need to be sent, however.

In our experiments, all data traffic is sent to only a single destination, reducing ODMRP to the unicast case and leaving the inclusion of the data packet in the JoinQuery as the primary difference between ODMRP and AODV.

3 Experimental Setup

We conducted outdoor, indoor, and simulation experiments for the four routing algorithms described above. All three experiments used the *same* hardware and software infrastructure, and the indoor and simulation experiments used the position traces from the outdoor experiment to duplicate the outdoor movement patterns. For the indoor experiment, the movement patterns were duplicated by having a laptop ignore any traffic from a laptop that was too far away at the corresponding time in the outdoor experiment. This approach to simulating movement does not address the fact that the wireless channel behaves differently when all the laptops are on the same tabletop, and we will see below that there are significant performance differences when the four algorithms are run indoors rather than outdoors (and in simulation rather than with real hardware).

3.1 Hardware Platform

The routing experiments ran on top of a set of 41 Gateway Solo 9300 laptops, each with a 10GB disk, 128MB of main memory, and a 500MHz Intel Pentium III CPU with 256KB of cache. We used one laptop to control each experiment, leaving 40 laptops to actually run the ad hoc routing algorithms. Each laptop ran Linux kernel version 2.2.19 with PCMCIA card manager version 3.2.4, and had a Lucent (Orinoco) Wavelan Turbo Gold 802.11b wireless card. Although these cards can transmit at different bit rates and can auto-adjust this bit rate depending on the observed signal-to-noise ratio, we used an ad hoc mode in which the transmission rate was fixed at 2 Mb/s. Specifically, we used Lucent (Orinoco) firmware version 4.32 and the proprietary ad hoc "demo" mode originally developed by Lucent. Although the demo mode has been deprecated in favor of the IEEE-defined IBSS, we used the demo mode to ensure consistency with a series of ad hoc routing experiments of which this outdoor experiment was a culminating event. [6] The fixed rate did

[6] Our series of experiments began before IBSS was available in standard 802.11b wireless cards, and time and personnel constraints prevented both demo-mode and IBSS-mode experiments.

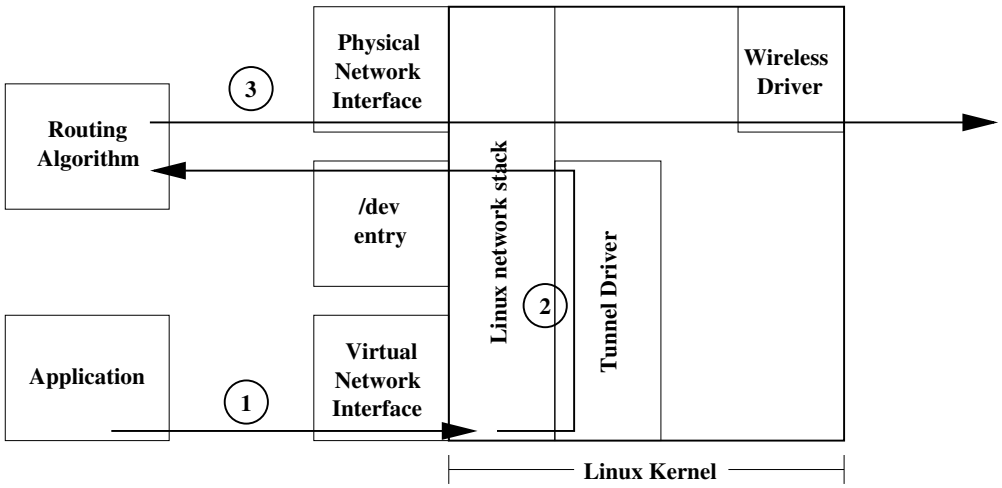

Figure 1: A schematic of the tunnel device and the path of an outgoing data packet.

make it much easier to analyze the routing results, since we did not need to account for automatic changes in each card's transmission rate. On the other hand, we would expect to see variation in the routing results if we had used IBSS instead, both due to its multi-rate capabilities and its general improvements over the demo mode. The routing results remain representative, however, since demo mode provides sufficient functionality to serve as a reasonable data-link layer. Finally, each laptop had a Garmin eTrex GPS unit attached via the serial port. These GPS units did not have differential GPS capabilities, but were accurate to within thirty feet during the experiment.

3.2 Software Infrastructure

To allow as accurate a comparison as possible between the four routing algorithms, we needed the implementations of those algorithms to be as similar as possible. If one algorithm operated in kernel space and another operated in user space, for example, it would not be possible to attribute any difference in packet latencies solely to the way in which an algorithm finds and uses its routes. For this reason, although we used existing source code as a guide in all four cases, we implemented each algorithm from scratch so that we could minimize any implementation differences. There are four key features that the implementations have in common:

- All four algorithms are implemented as user-level applications through the use of a tunnel device. The tunnel device, which we ported from FreeBSD, provides a network interface on one end and a file interface, specifically a /dev entry, on the other end. All packets sent to the network interface can be read from the file, and all packets written to the file will be sent up Linux's network stack. Each node is assigned two IP addresses, one associated with the physical network device, and one associated with the tunnel or virtual network device. Applications use the virtual IP address, while the routing algorithms use the physical IP address. We configure the standard Linux routing tables so that all virtual IP addresses are directed to the virtual network interface. Any application-level packets, therefore, are directed through the tunnel, and the routing algorithms can read those packets from the file end. Figure 1 shows the tunnel device and the path of an *outgoing* data packet.

  The tunnel allowed us to avoid any implementation work at the kernel or driver level, and also to switch from one routing algorithm to another (during experiments) simply by stopping one

user-level process and starting another. The drawback of our approach is the additional overhead associated with moving packets between kernel and user space. Our laptops, however, had more than enough capacity for the modest amount of network traffic that we generated in most of our experiments,[7] and thus we chose implementation simplicity over maximum performance.

- All four algorithms use UDP for traffic destined for a specific neighbor and multicast IP for traffic that should reach *every* neighbor. Multicast IP, as opposed to broadcast IP, allows us to run multiple routing algorithms at the same time without adding filtering code to every algorithm, a useful feature in some of our earlier experiments. Each algorithm simply subscribes to its own multicast address.

- All four algorithms use an event loop that invokes algorithm-specific handlers in response to (1) incoming UDP or multicast IP network traffic or (2) the expiration of route or other timeouts. As with user-level routing, the event-loop approach leads to additional overhead, but allows more straightforward implementation.

- All four algorithms are implemented in C++ and *share* a core set of classes. These classes include the event loop, as well as unicast and multicast, routing, and logging support.

With these design choices and shared classes, algorithm-specific code is confined to the packet handler classes that process incoming control and data packets, the timer handler classes that process timed actions (such as route expiration), a logging class that logs algorithm events, and utility classes that serialize and unserialize control packets. Minimizing the algorithm-specific code simplified implementation and debugging, and should make the routing results as independent of a particular implementation choice as possible.

The routing algorithms themselves are not enough, of course, since we also need to generate application-level traffic, record a position trace for each laptop, and manage the end-to-end experiment. A *traffic generator* runs on each laptop, and sends a sequence of packet streams to randomly selected laptops. Each stream contains a random number of packets of a random size. Two Gaussian distributions determine the packet numbers and sizes, two exponential distributions determine the delay between streams and packets, and a uniform distribution determines the destination laptops. A *GPS service* also runs on each laptop, reading and recording the current laptop position from the attached GPS unit, and broadcasting beacons that contain the laptop's position (as well as sequence-numbered positions that it has received from other laptops). We thus are flooding the GPS beacons through the network, an appropriate choice in our application domain where soldiers and first responders need to see a continuous view of each other's positions. In addition, broadcasting the beacons allows us to build a connectivity graph, independent of any particular routing algorithm, as to which laptops actually can *hear* which other laptops. Such a connectivity graph serves as input to the indoor experiments, and also will allow future examination of cases where the routing algorithms did not discover a route even though there was at least some minimal connectivity that could have made the route possible. Finally, we use a set of Tcl scripts to setup and run the experiments.

3.3 Outdoor Experiment

The same software infrastructure is used for the outdoor, indoor and simulation experiments, but we will describe most of the specific parameters (timeout values, beacon intervals, etc.) in this

[7] We will see that the STARA routing algorithm overloaded our network and laptops.

section. We use the same parameter values for the indoor and simulation experiments, since we seek to compare all three sets of results.

The outdoor routing experiment took place on a rectangular athletic field measuring approximately 225 (north-south) by 365 (east-west) meters. This field can be roughly divided into four flat, equal-sized sections, three of which are at the same altitude, and one of which (at the south-east corner) is approximately four to six meters lower. There was a short, steep slope between the upper and lower sections. We chose this particular athletic field because it was physically distant from campus and the campus' wireless network, reducing potential interference. In addition, we configured the 802.11 cards to use wireless channel 9 for maximum separation from the standard channels of 1, 6 and 11, further reducing potential interference. We used all 41 laptops, one as the control laptop, and 40 as application laptops.

The GPS service on each laptop recorded the current position (latitude, longitude and altitude) once per second, and synchronized the laptop clock with the GPS clock to provide sub-second, albeit not millisecond, time synchronization. Every three seconds, the GPS service broadcast a beacon containing its own position and any other positions about which it knew. Three seconds is longer than strictly necessary for displaying accurate positions to soldiers or first responders, but was necessary to build a reasonably accurate connectivity trace. Each beacon contains at most 41 position records of 21 bytes each, and has a maximum data payload of 861 bytes.

The traffic generator on each laptop generated packet streams with a mean packet size of 1200 bytes (including UDP, IP and Ethernet headers), a mean of approximately 5.5 packets per stream, a mean delay between streams of 15 seconds, and a mean delay between packets of approximately 3 seconds. These parameters produced approximately 423 bytes of data traffic (including UDP, IP and Ethernet headers) per laptop per second, a relatively modest traffic volume, but corresponding to the traffic volume observed during trial runs of one of our prototype military applications [Gra00].

Each of the routing algorithms, APRL, AODV, ODMRP and STARA, ran for fifteen minutes with a two-minute period between successive routing algorithms to handle setup and cleanup chores. Fifteen minutes per algorithm leads to an overall experiment time of approximately an hour and a half (given the time needed for initial boot and experiment startup), corresponding to the maximum reliable lifetime of our laptop batteries. The traffic generator ran for thirteen minutes of each fifteen-minute period, starting one minute after the routing algorithm to allow the pro-active routing algorithms to reach a stable state.

The best parameters for the four routing algorithms could not be determined precisely beforehand, since there are no experiments of this size for which data is available. Instead, we used values that gave effective results in published simulation studies or that were set as default values in the sample source code obtained from the developers. We did adjust some values to achieve some degree of consistency between the algorithms, however.

Summarizing the major parameters, APRL recorded up to seven routes per destination, one primary and six alternates, broadcast its beacons every 6 seconds, and expired any route that had not been refreshed by a beacon within the last 12 seconds. STARA broadcast a neighborhood probe every 2 seconds, sent a dummy data packet if a path had gone unexplored for 6 seconds, removed a neighbor from a node's neighborhood set if two successive neighborhood probes passed without an acknowledgment, and weighed delay estimates by 0.9 on each update to exponentially forget old delay information as new information became available. AODV broadcast each RREQ twice, expired a route if it had not been used in 12 seconds, sent a HELLO every 6 seconds, and removed a neighbor from a node's neighborhood set if it did not receive two successive HELLOs. ODMRP refreshed an in-use route every 6 seconds, and expired a route (i.e., a forwarding group) if it had not been used for 12 seconds. As can be seen, these parameters reduce to 6 seconds between beacons, HELLO messages or route refreshes for APRL, AODV and ODMRP, and 12 seconds for a

route to time out for APRL, AODV and ODMRP, either by direct timeout or by failure to receive two successive HELLOs. Using equivalent values for STARA, however, led to unacceptably slow convergence of the delay estimates, particularly given the fifteen-minute window available to us for each algorithm. Reducing the parameters improved convergence, but at the expense of even more control overhead, an effect that we will consider below.

The laptops moved continuously during the experiment. At the start of the experiment, the participants were divided into equal-sized groups of ten, each participant given a laptop, and each group instructed to randomly disburse in one of the four sections of the field (three upper and one lower). The participants then walked continuously, always picking a section different than the one in which they were currently located, picking a random position within that section, walking to that position in a straight line, and then repeating. This approach was chosen since it was simple, but still provided continuous movement to which the routing algorithms could react, as well as similar spatial distributions across each algorithm.

Finally, during the experiment, seven laptops generated no network traffic due to hardware and configuration issues, and an eighth laptop generated the position beacons only for the first half of the experiment. The seven complete failures left thirty-three laptops actually participating in the ad hoc routing.

3.4 Indoor and Simulation Experiments

The indoor and simulation experiments used the recorded GPS information from the outdoor experiment to simulate the outdoor network connectivity. Specifically, if a node did *not* receive two successive GPS beacons from another node, the two nodes were assumed to be out of range of each other (disconnected). Otherwise the two nodes were assumed to be in range of each other (connected). In addition, we used the same routing-algorithm parameters as in the outdoor experiment and the same codebase (through the use of a direct-execution technique [LYN+04] in the simulation case), and were careful to take into account the seven failed laptops. For example, the seven failed laptops were left in or configured to the same failed state that had occurred outdoors, and the traffic generators were allowed to generate traffic destined for one of those failed laptops. Finally, we averaged the results from multiple runs, each with a different traffic generation pattern to get the final indoor and simulation results. The overall goal of this approach is to leave the behavior of the wireless channel as the primary difference between the outdoor, indoor, and simulation experiments.

4 Analysis

We evaluate the relative performance of the four algorithms with four metrics: *message delivery ratio, communication efficiency, hop count, and end-to-end latency.* Before examining these metrics, we define three important terms. First, a *message* is a group of dummy bytes produced by a node's traffic generator for intended transportation to a randomly-selected destination. All of our generated messages are small enough to fit into a single data packet. [8] Second, a *data packet* is any transmitted packet containing message data. Therefore, every *message* requires the transmission of at least one *data packet* to reach its destination. A message also can generate more than one data packet, depending on the length of the route or the delivery strategy of the algorithm. It also is possible for a message to generate *no* data packets, since the sending node may fail to identify any active route toward the message's destination. Finally, a *control packet* is any transmitted

[8] The mean message size was 1200 bytes, including all relevant headers.

	Message Delivery Ratio	Data Packets Per Message	Control Packets Per Message	Total Packets Per Message	Average Hop Count (successful messages)	Message Latency (seconds)
AODV	0.50	1.32	6.18	7.50	1.61	0.37
APRL	0.20	0.90	32.40	33.30	2.11	0.49
ODMRP	0.77	22.79	22.80	45.59	2.47	1.62
STARA-S	0.08	0.20	150.47	150.67	1.18	2.98

Table 1: The key outdoor statistics for each algorithm

packet that does not contain message data. Control packets are the means by which our routing algorithms communicate status information with nearby nodes.

4.1 Outdoor Results

Message Delivery Ratio. The message delivery ratio is the total number of messages received at their intended destination divided by the total number of generated messages (over the lifetime of the algorithm's run), and measures each algorithm's overall success in providing end-to-end message delivery. This metric could be referred to as the *packet delivery ratio*, but we substitute the term *message* for *packet* to emphasize that a single application-level message can turn into many network packets.

The first column of Table 1 shows the message delivery ratio for each algorithm. A striking result is the dominance of ODMRP, best explained by ODMRP's aggressive approach to route discovery. Instead of discovering a route and *then* sending the desired message, ODMRP embeds the message inside the route-discovery control packets. This greatly increases the chance that a message will reach its intended destination, since we need only to get one message from source to destination. If route discovery is a separate process, we must get three messages from source to destination (or back), the route-discovery control message, the route-response control message, and the application-level data message. AODV performs better than APRL for a similar reason. APRL pro-actively finds routes before they are needed, but the timeout values allow a relatively large time window during which a route physically might no longer be viable but still appears in most of the routing tables. Messages generated during this time will be lost, and will count against the message delivery ratio. If the nodes in our experiment had been more static, or if the physical environment had otherwise provided less opportunity for route breakage, we would see much less of a gap between the AODV and APRL delivery ratios. Different timeout values or minimal message-buffering capabilities also might help APRL.

STARA-S' message delivery ratio is the worst of the four algorithms, simply because STARA-S overwhelms the wireless network and the individual nodes with the dummy data packets used to re-explore previously high-latency routes. Gupta proposes condensing multiple control packets arriving at a node into a single control packet before rebroadcast [Gup00], but neither he nor we implemented this solution. As is, STARA-S confirms again that inefficient handling of a single control-packet type can undo any potential benefits of a more complex algorithm.

Communication Efficiency. The second through fourth columns of Table 1 show the average number of data packets, control packets, and total packets transmitted for each generated message. The first two are calculated by dividing the total number of data or control packets by the total number of generated messages (over the traffic-generation lifetime of each algorithm), and the

third is simply the sum of the other two. These metrics provide insight into each algorithm's data transportation efficiency.

ODMRP dominates on data-packet overhead since it embeds many data packets inside flooded control packets. If the size of the transmitted messages were large, the effect of this data packet load on available bandwidth would be dramatic. In our experiment, however, the generated traffic was relatively modest, and ODMRP did not create excessive congestion. In addition, if the traffic were concentrated on fewer destinations, or if the network were more static, we would observe fewer data packets since ODMRP would not need to flood the network as often. AODV transmitted 1.32 data packets per message while APRL transmitted only 0.90. This difference, although small in magnitude, is quite significant since APRL used longer routes on average than AODV, suggesting that the data-packet count should be higher. APRL often had no route at all from a source to a destination, however, leading to many messages that produced *no* data packets. Instead, the messages were dropped on the source node. Finally, STARA-S transmitted the fewest data packets, simply because the overload conditions prevented most data packets from making it onto the network (and prevented STARA-S from accurately discovering routes). Note that we count dummy data packets as control packets, rather than data packets.

On the other hand, STARA-S produces the most control traffic, generating, on average, 150 control packets for each message due to its excessive transmission of dummy data packets. APRL generated 32 control packets for each message, while AODV, the best for this metric, generated only 6 control packets for each message. In our scenario of modest traffic and dynamic connectivity, AODV's reactive discovery clearly required less control overhead than APRL's pro-active discovery.

Finally, if we consider the total number of data *and* control packets versus the total number of messages, we see that AODV generates an average of 7.50 packets per message, APRL generates 33.30, ODMRP generates 49.59, and STARA-S generates 150.67. Surprisingly, ODMRP does not fare much worse than APRL. One might assume that ODMRP's aggressive network flooding would lead to a more significant increase in traffic costs as compared to APRL's periodic route advertisements, but ODMRP's 45.59 packets transmitted for each message is not overwhelmingly larger than APRL's 33.30. It should, however, be noted that if the traffic volume increased, APRL could gain a noticeable lead over ODMRP, particularly in terms of transmitted *bytes*. The majority of APRL's traffic is in the form of streamlined control packets [9] sent at fixed intervals, whereas ODMRP includes copies of its data packets with much of its route-discovery traffic. Considering that AODV and ODMRP are both reactive algorithms that flood the network to discover routes, it also was surprising at first glance to note how many fewer packets per message are required by AODV. AODV, however, is a unicast protocol, and can halt the route-discovery process at any intermediate node that contains a valid route to the destination. ODMRP must always fully flood the route-discovery packets so that new nodes have the opportunity to join the multicast group.

Generally, APRL's volume of control traffic depends on its beaconing interval (time-based), AODV's volume depends on the rate at which sources send data traffic to new destinations (data traffic-based), and STARA-S' volume depends on the rate at which it re-explores unused routes (time-based). The relative control overhead of the algorithms can change significantly depending on the traffic pattern and timeout values, therefore, and APRL may gain a significant advantage over reactive algorithms such as AODV if the amount of data traffic increases significantly.

Hop Count. The next column of Table 1 shows the average number of hops that successfully received messages traveled to reach their destination. For ODMRP, we use the hop count for the first copy of each message to reach the intended destination.

STARA-S has the lowest average hop count for successfully received packets, but this result is

[9] APRL includes only a simple binary indication of whether a route exists in its routing beacons.

due to the excessive control traffic that made successful packet transmission difficult. Only data packets whose final destination was an immediate neighbor of the original sender had a good chance of successful receipt. AODV required the next fewest hops for successful packets, since AODV tries to select for the shortest path, whereas APRL and ODMRP do not consider path length. In fact, ODMRP has the largest average hop-count value, since many of its messages follow a semi-random path to the destination during undirected route discovery phase, On the other hand, if traffic was concentrated on a few destinations, ODMRP's hop count would drop significantly, since most data traffic could be sent over an already discovered route (rather than included in a flooded route-discovery packet).

One interesting result, not shown in the table, is that STARA-S dropped or lost nearly 88% of the generated messages before those messages left the source node (zero-hop), and APRL dropped or lost nearly 63%. For STARA-S, which averaged nearly 150 packets per message, interference caused the large number of zero-hop failures, with most such failures involving the transmission of a packet that was never received. For APRL, most zero-hop failures occurred when APRL explicitly dropped the packet due to a lack of a valid route to the destination node, indicating that APRL's periodic route advertising scheme was unable to consistently maintain adequate routing information in our experiment. While there were many cases in our experiment where a route physically did not exist between two nodes, AODV had a zero-hop failure rate of only 25%, indicating a more serious problem with APRL's performance.

End-to-End Latency. The main obstacle to calculating end-to-end message latencies is a lack of synchronization between the individual node clocks, creating a situation in which a comparison of receiver and sender timestamps is not sufficient for generating accurate latency values. Although NTP was a possibility, we made the decision to avoid the extra bandwidth usage during the experiments. Instead, we relied on the GPS units to provide accurate timestamps through our periodic synchronization of the node clock with the GPS clock. Since we required regular GPS queries for other purposes (such as tracking node mobility), this approach did not introduce significant extra overhead, and required no bandwidth usage. We found, however, that our node clocks still drifted from each other on the order of tens to a few hundreds of milliseconds, partially due to delays in reading the time from the GPS unit and invoking the kernel system-time calls.

Though relatively small, this clock drift is still significant, since many of our calculated end-to-end latency values are within the same order of magnitude. To overcome this, we paired the log entries corresponding to transmission and reception of the same beacons, and then used the paired timestamps to calculate the average clock skew between laptop pairs over non-overlapping time windows. The skew value for the appropriate window then was used to "correct" the apparent latency of a transmitted data packet. We recognize that this approach is only approximate for several reasons: (1) it is unrealistic to assume that two nodes A and B will receive and timestamp a broadcast beacon at the same instant, due to physical transmission time and computational overhead that may delay the actual log entry; (2) in a multi-hop routing environment, it is possible that the sender and the receiver of a given packet are too distant to have recently received the same beacon, producing a time window during which the sender and receiver have no available clock-skew value; and (3) using the average observed time skew calculated over a given bucket duration is not necessarily as accurate as always searching out the closest single time synchronization event. Accordingly, we do not present our end-to-end latency values as precise measurements. We do, however, maintain that our corrected values are more accurate indications of transit time than relying on uncorrected timestamps.

The last column of Table 1 shows the average corrected end-to-end latency value value for successful messages. We find the expected relationship between end-to-end latency and hop count. For AODV, APRL, and ODMRP, the average end-to-end latency value increases as the hop count

increases (although not strictly proportionally). For STARA-S, the hop count is low, but the end-to-end latency is high. This abnormality arises due to the large volume of control traffic, which significantly increases the time needed to receive and forward any data packet.

Conclusions. Any conclusions that we draw from this outdoor experiment must be qualified by the conditions of our particular testing environment. A markedly different scenario would produce markedly different results. For example, our nodes were mobile, our terrain was obstruction-free but non-uniform, and our participants carried the laptops tucked under their arms (thus potentially placing their bodies between the 802.11 card and the intended destination laptop). The dynamic environment penalizes an algorithm like APRL that does not seek routes on demand, while the modest traffic load helps an aggressive flooding algorithm like ODMRP. The way in which the laptops were carried changes the connectivity characteristics, and, in particular, may prevent successful transmission along routes that would have been fine with head-mounted antennas. Other important factors include the random pair-wise communication between nodes and the human approximation of the random-waypoint mobility model that we used. With such qualifications in mind, we can present the following conclusions, which track with other results in the literature:

- **AODV is efficient, but far from effective in all scenarios.** Though its message delivery ratio was not as high as ODMRP, it delivered messages significantly better than APRL and STARA-S. More importantly, on all measures of communication efficiency, AODV generated by far the least amount of traffic for each message. In addition, AODV was successful in consistently finding short paths, giving it the additional advantage of having the lowest average end-to-end latency. In an environment with limited bandwidth or energy resources, AODV is a good choice. In an environment with extremely high mobility, however, AODV may not be able to keep up with topology changes.

- **ODMRP is highly effective for some scenarios, but very bad for others.** ODMRP generates significant network overhead. Its flooding is bandwidth-intensive, and if data packets are large, ODMRP could suffer significant performance penalties. At the same time, however, it had the highest message delivery ratio of all four algorithms, and will handle high mobility better than AODV through its packet flooding. If bandwidth and energy resources are plentiful, data packets are small, and communication reliability is crucial, ODMRP is a good choice.

- **Reactive is better than proactive in dynamic environments.** APRL's and STARA-S's poor performance, as compared to AODV and ODMRP's relative success, highlights the general advantage of a reactive routing approach. Our analysis of APRL shows an unnecessarily large number of messages dropped before leaving their source node, and STARA-S crippled itself with excessive proactive discovery. Although STARA-S likely can be fixed through additional engineering, if we had restrained the existing STARA-S to produce a reasonable volume of control traffic, it would have faced the same lack of quality routing information as APRL. Similarly, if we had increased APRL's beaconing interval to increase the timeliness of its routing information, it would have suffered from an excess amount of control traffic like STARA-S. This observation underscores the perhaps unresolvable tension between control traffic and message delivery success for proactive algorithms operating in dynamic environments: if you make your algorithm efficient, its reliability drops; if you make your algorithm reliable, its efficiency drops. Reactive approaches clearly are preferable for many dynamic scenarios.

Overall, this outdoor experiment should serve as a useful data point for future work, demonstrating the relative performance of four common (types of) routing algorithms in a (relatively)

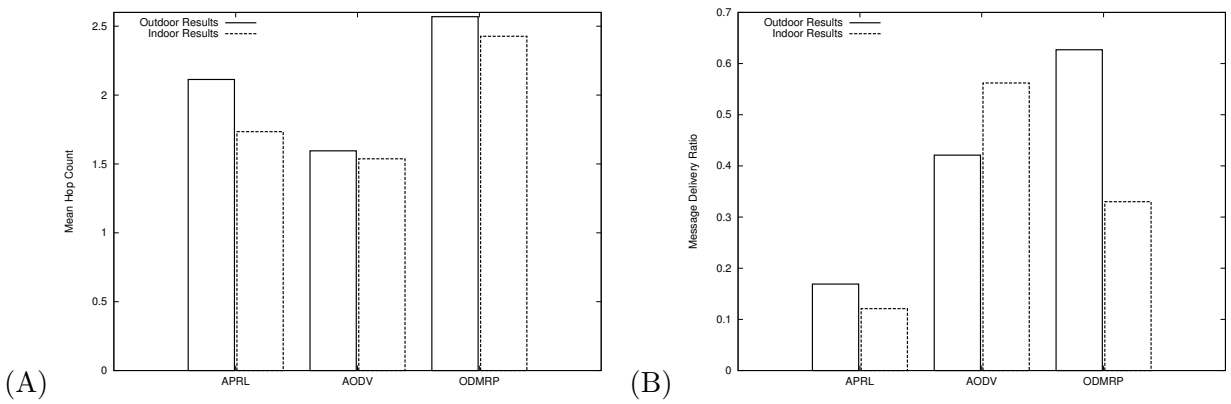

Figure 2: The hop counts **(A)** and message delivery ratios **(B)** for the indoor and indoor experiments. ODMRP delivers far more message outdoors, while AODV delivers far more messages indoors.

large-scale, real-world experiment.

4.2 Comparison with Indoor Results

During the indoor experiment, we placed the laptops on two shelving units, ten feet apart from each other with twenty and twenty-one laptops per shelf respectively. We used the same hardware and codebase except that (1) we configured the GPS service to read the position information from the outdoor mobility trace, rather than from the GPS unit; and (2) we derived a connectivity trace from the outdoor beacon logs, and configured each routing algorithm to ignore traffic from laptops that were actually unreachable during the outdoor experiment. We used three different (randomly generated) traffic patterns, and averaged the indoor routing results for three separate runs to ensure that we were comparing the outdoor results with a meaningful view of indoor results. [10] The primary difference between the outdoor and indoor experiment then is that every indoor laptop can hear every packet due to the physical proximity of the nodes, a situation in which the 802.11b protocol theoretically can significantly reduce collisions through its standard CSMA/CA[11] protocol. We would expect, therefore, that the routing algorithms will behave differently indoors even though we use the outdoor connectivity information.

What was surprising in our case, however, was the magnitude of the change in the message delivery ratios when we moved indoors. Figure 2 shows the message delivery ratio and mean hop count for APRL, AODV and ODMRP; we omit STARA-S since the control-packet overload led to the same minimal performance as in the outdoor experiment (and the outdoor overload made the derived connectivity trace suspect anyway). Although the hop counts are roughly the same in the outdoor and indoor experiments, the message delivery ratios for AODV and ODMRP change dramatically, with AODV suddenly doing much better than ODMRP. Even if the outdoor traffic pattern somehow represents an outlier in the traffic space, the switch in relative performance remains significant. The largest standard deviation for any indoor message-delivery value was 7.732% for ODMRP, and the outdoor results are more than three standard deviations away from the indoor average for all three algorithms.

The reason for this performance switch is clear in hindsight. ODMRP includes the data packet in the flooded route-discovery packets, leading to route discovery packets an order of magnitude

[10] Future software enhancements will allow us to replay the outdoor traffic pattern exactly.
[11] Carrier Sense Multiple Access with Collision Avoidance

larger than those of AODV (for our payload sizes). In addition, every node must rebroadcast the route-discovery packet, since ODMRP is a multicast protocol that allows nodes to dynamically join multicast groups. During the indoor test, every laptop is in range of every other laptop, and the CSMA/CA protocol cannot overcome the higher collision rate caused by ODMRP's flood of large route discovery packets. Outdoors, each packet in the flood can be heard by only a fraction of the laptops, significantly reducing the collision potential. At the same time, AODV generates a much smaller number of much smaller packets, leading to relatively few collisions even indoors. Moreover, these packets will not suffer the environmental effects that can occur outdoors. In other words, in the absence of collisions, each packet has a much greater chance of successful reception indoors than outdoors. These two factors, increased collisions for ODMRP and greater chance of successful reception for AODV, lead to the performance switch.

These indoor results are important for two reasons. First, they confirm again that indoor experiments with real hardware will not necessarily lead to a correct view of algorithm performance, no matter the scale of the ad hoc network. In this case, indoor experiments would have led us to select the wrong algorithm for outdoor use, leading to a 33% drop in outdoor performance. Second, the results also suggest that the benefits of embedding a data packet inside a flooded route-discovery packet depend heavily on the number of nodes that can hear each packet within the flood. Even when used entirely outdoors, it is worth considering routing algorithms that turn on or off embedded data packets depending on the "clustering" of the nodes.

4.3 Comparison with Simulation Results

Given the failure of indoor experiments to accurately represent outdoor results, we need to consider simulation. As described in an earlier paper [LYN+04], we configured a wireless network simulator to mimic our outdoor experimental conditions. We then conducted a simulation experiment with three different commonly used physical-layer models under a variety of traffic loads, considering in each case whether the simulation provided results comparable to those observed outdoors.

Simulator. SWAN is a simulator for wireless ad hoc networks that provides an integrated, configurable, and flexible environment for evaluating ad hoc routing protocols, especially for large-scale network scenarios. SWAN contains a detailed model of the IEEE 802.11 wireless LAN protocol and a stochastic radio channel model, both of which we used in this study. In addition, we used SWAN's direct-execution simulation techniques to execute the *same* routing code that was used in the indoor and outdoor experiments. We modified the real routing code only slightly to allow multiple instances of a routing protocol, the traffic generator, and the GPS service to run simultaneously in the SWAN's single address space. We also extended the simulator to read the position traces generated by the outdoor experiment. We included the seven failed nodes, and ran the GPS service program so that we would generate just as much network traffic as in the outdoor and indoor experiments.

Experiments. We examined three radio propagation models: a free-space model, a two-ray ground reflection model, and a generic propagation model. The simulator delivered each transmitted packet to all neighbor stations that could receive the packet with an average signal power beyond a minimum threshold, and we used the propagation models to calculate the signal power and signal-to-noise ratio at the receiver. We combined the three models with the connectivity trace derived from the beacon logs, leading to six different radio propagation models in simulation: three that used the connectivity traces and three that did not. In the first three cases, the model used the connectivity trace to determine whether a packet from one node could reach another node, and then used the radio propagation models to determine the receiving power for the interference calculation. Using a connectivity trace is not ideal, since we would instead prefer a model that

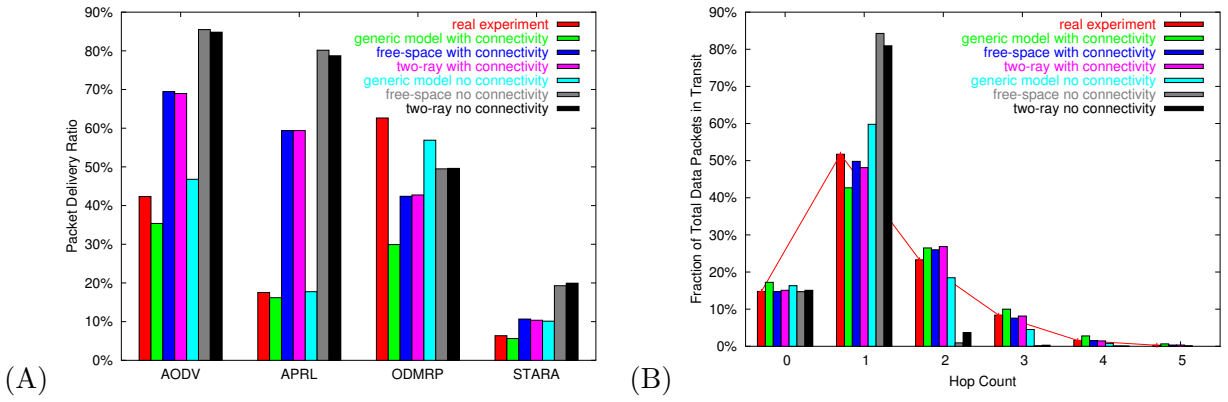

Figure 3: **(A)** Comparison of the message delivery ratio from the outdoor experiment with the message delivery ratio under different simulation strategies ("with connectivity" means we used the connectivity trace); **(B)** comparison of the hop counts for AODV in the real experiment with the hop counts for AODV under different simulation strategies.

predicts connectivity, but here we are interested in comparison between the two options.

Simulation Results. Figure 3 *(A)* shows the message delivery ratio from the outdoor experiment and the simulation runs with the six radio propagation models. Each simulation result is an average of five runs, and the standard deviation is insignificant. We used typical parameters, 2.8 as the path-loss exponent and 6 dB as the standard deviation for shadow fading [Rap96]. We have several general observations, with more details appearing in [LYN+04].

First, the simple generic propagation model (with standard path-loss and fading parameters) offered an acceptable prediction of the outdoor performance of the routing algorithms. Second, different propagation models predicted vastly different protocol behaviors. The difference is significant enough in some cases to lead to incorrect conclusions, such as when comparing the performance of AODV and ODMRP. Moreover, the inaccuracy introduced by the propagation model is non-uniform and can undermine a performance comparison study of different protocols. The free-space and two-ray ground models were particularly susceptible to underestimation or overestimation of the delivery ratio as exaggerated transmission ranges led to shorter paths but more contention than had actually occurred outdoors. The STARA-S results, however, are influenced by potential inaccuracies in the derived connectivity trace, as with the indoor experiment. Finally, the propagation models that used the connectivity trace generally lowered the message delivery ratio, when compared with the propagation models that did not use the connectivity trace. This result is not surprising: the connectivity trace, to some degree, represents the peculiar radio propagation scenario of the outdoor environment. In our experiment, there were significant elevation changes in the test field that led to the obstruction of radio signals between laptops that were close by in distance. Without connectivity traces, the propagation model assumes an omni-directional path loss dependent only on the distance, which resulted in a more connected network (fewer hops) and therefore a better delivery ratio.

Other statistics, such as the average hop count for AODV as shown in Figure 3 *(B)*, suffered similar underestimation or overestimation as the message delivery ratio. We also see that the connectivity trace was helpful in predicting the route lengths, which confirms that the problem with the free-space and two-ray models using the connectivity trace was that they did not consider packet losses due to the variations in receiving signal power.

With these simulation results, it is clear that pre-simulation work to estimate path-loss char-

acteristics or more complex channel models might be needed. On the other hand, if the objective is to compare protocols, knowledge that the generic propagation model is good lets us compare protocols using a range of path-loss values. While this does not *quantify* behavior, it may allow us to make *qualitative* conclusions about the protocols over a range of environments.

5 Related Work

There have been a limited number of evaluations of ad hoc routing algorithms on top of real hardware. For example, De Couto et al evaluate the performance of a modified version of DSDV in a static, 29-node indoor testbed, discovering that a path-throughput metric for route selection can improve total network throughput for many traffic scenarios [CABM03]; He et al use a static, 70-node indoor sensor network to evaluate a distributed spanning-tree approach to route formation [HKK+04]; and Fang et al do a preliminary evaluation of their DAM data-routing and aggregation protocol on top of a static, 25-node indoor testbed [FZG03]. Most relevantly, Lundgren et al used their Ad Hoc Protocol Evaluation testbed (APE) to evaluate the performance of AODV and OLSR with up to 37 nodes moving along indoor hallways [LLN+02]. Other researchers have done real-world tests of other services in ad hoc networks, including navigation, localization and tracking services [LRR03, SHS01, WSBC04]; lightweight security mechanisms [PSW+01], MAC protocols [WC01], Our work complements these and similar efforts in that we compare four ad hoc routing algorithms under outdoor conditions with a significant number of moving nodes. To our knowledge, our work represents the largest *outdoor* and *mobile* routing experiment for which results are publicly available.

Several researchers have developed general software infrastructures that can be used to implement (and evaluate) ad hoc routing or other protocols. For example, DIRAC provides a support framework for services, such as link-layer assisted fast handover, channel-adaptive scheduling, and link-layer enforced policing, in base-station networks, and may be extensible to ad hoc networks [ZZC+03]; the SPINS framework can be used to construct security-aware routing and other protocols in resource-constrained ad hoc networks [PSW+01]; Kawadia et al have developed an Ad hoc Support Library (ASL) that provides the low-level routing services needed in many ad-hoc routing protocols [KZG03]; Zhang and Li have developed an emulation-based testbed for evaluating ad-hoc routing implementations indoors [ZL02]; Lundgren et al have developed the APE testbed mentioned above [LLN+02, TLN03]; Kohler et al have developed the Click modular router for rapidly composing *elements* to create arbitrary routing functionality [KMC+00]; and Neufeld et al have developed the *nsclick* extension that allows the direct execution of Click modules inside ns simulations [NJG02]. DIRAC and SPINS have a different focus than ad hoc routing, but shared routing infrastructure that we used for our experiments can be viewed as a user-space version of the ASL, and has similar goals as the Click modular router and the nsclick extension (although is more specific to ad hoc routing). In addition, our experimental infrastructure operates in much the same way as the Zhang and Li testbed and the APE testbed, including the use of a tunnel device or modified driver to allow user-level routing.

Many researchers are working on improved simulation techniques that have the potential to provide a closer (or simpler) match between outdoor, indoor and simulated performance. For example, there has been work on interference modeling [JPPQ03], mobility modeling [HGPC99, SK99], and channel modeling [ZHKS04, KR03, KZJL01]. In addition, Takai et al explored the effects of different physical-channel models on simulation results, observing similar changes in relative algorithm ranking as when we re-ran our experiment indoors [TMB01]. All of this work will help to make future simulations accurate enough to predict outdoor behavior under much wider ranges

of conditions.

Finally, most researchers use simulation to evaluate new ad hoc routing algorithms or other protocols. We hope that our work can help to provide guidance on whether these simulation results accurately predict outdoor performance, and that our data corpus can be used to fine-tune the simulations for further evaluation. One simulation study of note is Broch's comparison of DSDV, TORA, DSR and AODV [BMJ+98]. Our own study parallels this study in many ways.

6 Conclusion

Generally, reactive algorithms were preferred for our (relatively) large number of nodes and our modest traffic load, with ODMRP outperforming AODV due its inclusion of the original message in the flooded route-discovery packets. Interestingly, however, ODMRP's performance dropped precipitously (and AODV's improved by a similar amount) when the nodes were indoors and could all hear each other, in both cases due to the different levels of contention and packet loss. Indoor experiments on real hardware clearly cannot predict the outdoor performance of common routing algorithms. On the other hand, the indoor performance does suggest that contention may play a larger role outdoors than might be expected, with results changing dramatically depending on the "clustering" of the network. Finally, the simulation results indicate that simulation, with an appropriate choice of models, can accurately predict outdoor performance. A non-experimental methodology for selecting the simulation parameters, however, remains unclear.

The most immediate area of future work must be the development of a community-wide corpus of outdoor ad hoc routing results, conducted under some standardized set of conditions to allow meaningful comparison of results taken at different times and by different organizations. Unfortunately, such an effort is beyond the capabilities of any single organization (with the possible exception of the government). Whether the corpus is small or big, another critical area of future work is to improve and develop procedures by which a simulation can be calibrated to match observed outdoor behavior. A large part of any such effort must be to reach a point where we can argue model completeness, so that we assure ourselves of simulation accuracy without running even more outdoor experiments as confirmation. Finally, and somewhat tangentially, some of the results in our outdoor experiment confirm that hybrid routing algorithms or routing algorithms that dynamically adjust their routing strategy according to the current network environment remain a very promising area of work.

Acknowledgments

Many thanks to Lisa Shay and Eileen Entin for assistance with the military scenarios that led to our routing experiments; to Brad Karp and Piyush Gupta for developing APRL and STARA respectively and generously making their code available to us; to Michael DeRosa for software development; and to the army of Dartmouth faculty, staff members, and students who gave up an afternoon to carry laptops around an athletic field.

References

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the 4th Annual Conference on Mobile Computing and Networking (MOBICOM 1998)*, pages 85–97, Dallas, Texas, October 1998.

[CABM03] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of the 9th Annual Conference on Mobile Computing and Networking (MOBICOM 2003)*, pages 134–146, San Diego, California, September 2003.

[FZG03] Q. Fang, F. Zhao, and L. Guibas. Lightweight sensing and communication protocols for target enumeration and aggregation. In *Proc. of the 4th Int'l Symposium on Ad Hoc Networking and Computing (MOBIHOC 2003)*, pages 165–176, Annapolis, Maryland, June 2003.

[GK97] P. Gupta and P. R. Kumar. A system and traffic dependent adaptive routing algorithm for ad hoc networks. In *Proc. of the 36th IEEE Conference on Decision and Control*, pages 2375–2380, December 1997.

[Gra00] R. S. Gray. Soldiers, agents and wireless networks: a report on a military application. In *Proc. of the 5th Int'l Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, April 2000.

[Gup00] P. Gupta. *Design and Performance Analysis of Wireless Networks.* PhD thesis, University of Illinois at Urbana-Champaign, August 2000.

[HGPC99] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *Proc. of the 2nd Int'l Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 1999)*, pages 53–60, Seattle, Washington, August 1999.

[HKK+04] T. He, B. Krogh, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, and J. Hui. Energy-efficient surveillance system using wireless sensor networks. In *Proc. of the 2nd Int'l Conference on Mobile Systems, Applications, and Services (MOBISYS 2004)*, pages 270–283, Boston, Massachusetts, June 2004.

[JPPQ03] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. of the 9th Annual Conference on Mobile Computing and Networking (MOBICOM 2003)*, pages 66–80, San Diego, California, September 2003.

[KK98] B. Karp and H. T. Kung. Dynamic neighbor discovery and loopfree, multi-hop routing for wireless, mobile networks. Harvard University, May 1998.

[KMC+00] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[KR03] S. A. Khayam and H. Radha. Markov-based modeling of wireless local area networks. In *Proc. of the 6th Int'l Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2003)*, pages 100–107, San Diego, California, September 2003.

[KZG03] V. Kawadia, Y. Zhang, and B. Gupta. System services for ad-hoc routing: Architecture, implementation and experiences. In *Proc. of the First Int'l Conference on Mobile Systems, Applications, and Services (MOBISYS 2003)*, San Francisco, California, May 2003.

[KZJL01] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig. A Markov-based channel model algorithm for wireless networks. In *Proc. of the 4th Int'l Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2001)*, pages 28–36, Rome, Italy, July 2001.

[LLN+02] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluations, March 2002.

[LRR03] Q. Li, M. De Rosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In *Proc. of the 9th Annual Conference on Mobile Computing and Networking (MOBICOM 2003)*, pages 313–325, San Diego, California, September 2003.

[LSG02] S.-J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Kluwer MONET*, 7(6):441–453, December 2002.

[LYN+04] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. Simulation validation using direct execution of wireless ad-hoc routing protocols. In *Proc. of the 18th Workshop on Parallel and Distributed Simulation (PADS 2004)*, Kufstein, Austria, May 2004.

[NJG02] M. Neufeld, A. Jain, and D. Grunwald. Nsclick: Bridging network simulation and deployment. In *Proc. of the 5th Int'l Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2002)*, pages 74–81, Atlanta, Georgia, September 2002.

[PR99] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *WMCSA*, pages 90–100, February 1999.

[PSW+01] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security suite for sensor networks. In *Proc. of the 7th Annual Conference on Mobile Computing and Networking (MOBICOM 2001)*, pages 189–199, Rome, Italy, July 2001.

[Rap96] T. S. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall, 1996.

[SHS01] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of the 7th Annual Conference on Mobile Computing and Networking (MOBICOM 2001)*, pages 166–179, Rome, Italy, July 2001.

[SK99] J. Scourias and T. Kunz. An activity-based mobility model and location management simulation framework. In *Proc. of the 2nd Int'l Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 1999)*, pages 61–68, Seattle, Washington, August 1999.

[TLN03] C. Tschudin, H. Lundgren, and E. Nordström. Embedding MANETs in the real world. In *Proc. of the 8th Int'l Conference on Personal Wireless Communications (PWC 2003)*, Venice, Italy, September 2003.

[TMB01] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *Proc. of the 2nd Int'l Symposium on Mobile Ad Hoc Networking and Computing*, pages 87–94, Long Beach, California, October 2001.

[WC01] A. Woo and D. Culler. A rate-adaptive MAC protocol for multi-hop wireless. In *Proc. of the 7th Annual Conference on Mobile Computing and Networking (MOBICOM 2001)*, pages 221–235, Rome, Italy, July 2001.

[WMB03] S. Wendelken, S. McGrath, and G. Blike. A medical assessment algorithm for automated remote triage. In *Proc. of the 25th Annual Engineering in Medicine and Biology Conference*, Cancun, Mexico, September 2003.

[WSBC04] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. HOOD: A neighborhood abstraction for sensor networks. In *Proc. of the 2nd Int'l Conference on Mobile Systems, Applications, and Services*, pages 99–110, Boston, Massachusetts, June 2004.

[ZHKS04] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proc. of the 2nd Int'l Conference on Mobile Systems, Applications, and Services (MOBISYS 2004)*, pages 125–138, Boston, Massachusetts, June 2004.

[ZL02] Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. In *Proc. of the Third Int'l Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002)*, pages 104–111, Lausanne, Switzerland, June 2002.

[ZZC+03] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, and J. J.-R. Li. DIRAC: A software-based wireless router system. In *Proc. of the 9th Annual Conference on Mobile Computing and Networking (MOBICOM 2003)*, pages 230–244, San Diego, California, September 2003.