Simulation Validation Using Direct Execution of Wireless Ad-Hoc Routing Protocols *

Jason Liu, Yougu Yuan, David M. Nicol,
Robert S. Gray[†], Calvin C. Newport[†], David Kotz[†],
and Luiz Felipe Perrone[‡]

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, IL 61801
{*jasonliu,yuanyg,nicol*}*@crhc.uiuc.edu*

[†] Department of Computer Science, Dartmouth College
6211 Sudikoff Laboratory, Hanover, NH 03755
{*robert.s.gray, calvin.c.newport*}*@dartmouth.edu, dfk@cs.dartmouth.edu*

[‡] Department of Computer Science, Bucknell University, Lewisburg, PA 17837
perrone@bucknell.edu

Abstract

Computer simulation is the most common approach to studying wireless ad-hoc routing algorithms. The results, however, are only as good as the models the simulation uses. One should not underestimate the importance of validation*, as inaccurate models can lead to wrong conclusions. In this paper, we use direct-execution simulation to validate radio models used by ad-hoc routing protocols, against real-world experiments. This paper documents a common testbed that supports direct execution of a set of ad-hoc routing protocol implementations in a wireless network simulator. The testbed reads traces generated from real experiments, and uses them to drive direct-execution implementations of the routing protocols. Doing so we reproduce the same network conditions as in real experiments. By comparing routing behavior* measured *in real experiments with behavior* computed *by the simulation, we are able to validate the models of radio behavior upon which protocol behavior depends. We conclude that it is* possible *to have fairly accurate results using a simple radio model, but the routing behavior is quite sensitive to one of this model's parameters. The implication is that one should i) use a more complex radio model that explicitly models point-to-point path loss, or ii) use measurements from an environment typical of the one of interest, or iii) study behavior over a range of environments to identify sensitivities.*

1. Introduction

Using simulation one must take the precaution that the model may not reflect the reality. Validation of a wireless network simulation is particularly difficult because not only must the implementation of the simulated protocol be validated against its design specifications, but also the model must be able to capture lower-level characteristics of the wireless environment with a proper level of abstraction [3]. The validation problem is amplified when the routing protocol deployed in a real system is implemented and maintained as a separate code base from the one used in simulation.

Direct-execution simulation alleviates the problem of maintaining separate code bases for the same routing protocol by executing the same code designed for real systems directly inside a wireless network simulator. We compile the routing protocol's source code with the simulator's source code with only moderate changes as necessary. The protocol's logic is executed inside the simulator and is driven by

*This work was supported in part by Dartmouth Center for Mobile Computing, DARPA (contract N66001-96-C-8530), the Department of Justice (contract 2000-CX-K001) and the Department of Defense (MURI AFOSR contract F49620-97-1-03821). Points of view in this document are those of the authors and do not necessarily represent the official position of DARPA or the United States government. The U. S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

the simulator's time advancing mechanism. In a discrete-event simulation, the routing protocol code is invoked as a result of the simulator processing events stored in the event queue. Since each protocol instance communicates with other simulated mobile stations by sending and receiving packets through well-defined system calls, we substitute these system calls with calls to the simulator. The packets are redirected to go through the simulated wireless network—all transparent to the protocol implementation. Using direct-execution simulation is desirable for prototyping a protocol implementation, which, after initial simulation evaluation, can be deployed directly in a real network.

Our interest in direct-execution simulation is that it can help us validate the wireless network simulator using the results from real experiments. We run the same routing protocol and application traffic generator code both in simulation and in the real experiment. The difference is that in the real experiment packets are transmitted via the wireless channel and are subject to delays and losses due to signal fading and collisions during the transmission. In simulation, these packets are translated into simulation events scheduled with delays calculated by the radio channel model. Depending on the modeling details, the simulation result may or may not reflect what would happen in reality. Direct-execution simulation provides us a valuable opportunity to investigate the effect of details of a wireless network model on the fidelity of the simulation study.

This paper documents our effort in supporting direct execution of a set of wireless ad-hoc routing protocol implementations and using direct-execution simulation to validate the underlying wireless network models by comparing the results from real-world experiments. We ported five routing protocol implementations for direct execution: APRL, AODV, GPSR, ODMRP, and STARA. Versions of all five protocols were implemented as part of the ActComm project, whose goal is to provide information access through a wireless network to soldiers in the field.[1] One contribution of our research is to provide a common testbed for direct execution of these protocols in simulation. More importantly, we instrumented the testbed to enable validation of various wireless network models. We embedded the routing protocol code with various logging functions. Each laptop computer running the routing protocols in the real experiment has a Global Positioning System (GPS) device and periodically records its location information and average receiving signal quality from other laptops. We later transformed these logs into traces of node mobility and radio connectivity. We adapted the simulator to read the traces and combined them with different stochastic radio propagation models to reproduce the test scenario inside simulation. We compared the results from running these routing protocols in simulation with those collected from the real experiment. Such comparison helps us understand the effect of different wireless network models on the behavior of ad-hoc routing algorithms.

We present two set of experiments in this paper. The first one compares two implementations of the AODV routing protocol, both running in simulation. One AODV implementation was specifically designed for the simulator—the protocol sends and receives packets, and schedules timeouts using the special functions provided by the simulator. The other AODV implementation was developed by the ActComm project, designed for the real network, and was directly executed in the simulator. The goal of this experiment is to validate both protocol implementations and to identify the overhead introduced by direct-execution simulation in terms of memory usage and execution time. The second set of experiments compares the results from a real field test with those from simulation. In the real experiment, we ran the routing protocols on 40 laptop computers, each equipped with a wireless device and a GPS unit. The laptops were carried by people walking randomly in a large field. In simulation, we applied different radio propagation models together with the traces derived from the real experiment. We directly executed the routing protocol implementations in simulation and compared the behavior of these protocols in both environments. The goal of this study is to highlight the importance of modeling decisions on the validity of a wireless simulation study.

The paper is organized as follows. Section 2 provides an overview of the implementations of the ActComm routing protocols and outlines the architecture of our wireless network simulator on which we directly execute these protocol implementations. In Section 3 we briefly describe issues related to direct-execution simulation. Section 4 presents the augmented simulation testbed designed for validation purposes. We focus on the experiments and results in Section 5. Section 6 concludes the paper.

2. Background

2.1. The Routing Protocols

We ported five protocols for direct execution. Any-Path Routing without Loops (APRL) is a proactive distance-vector routing protocol [5]. Rather than using sequence numbers, APRL uses ping messages before establishing new routes to guarantee loop-free operation. Ad-hoc On-Demand Vector (AODV) is an on-demand routing algorithm—routes are created as needed at connection establishment and maintained thereafter to deal with link breakage [10]. Greedy Perimeter Stateless Routing (GPSR) uses GPS positions of the mobile stations to forward packets greedily along a path toward the target's physical location [6]. GPSR uses a perimeter-following algorithm to for-

[1] http://actcomm.thayer.dartmouth.edu/.

ward packets around the boundaries of empty regions that contain no laptops (and hence cause greedy forwarding to fail). On-Demand Multicast Routing Protocol (ODMRP) maintains a mesh, instead of a tree, for alternate and redundant routes for each multicast group [7]. It does not depend on another unicast routing protocol and, in fact, can be used for unicast routing. System and Traffic Dependent Adaptive Routing Algorithm (STARA) uses shortest-path routing [2]. The distance measure is calculated by the mean transmission delay instead of the hop count.

We implemented these protocols for the ActComm project in C++ on Linux. All five implementations perform their routing in user space using IP tunneling and UDP sockets. An IP tunnel is a virtual network device with two endpoints: one as a regular network interface, and the other as a Unix file. Packets sent to the network interface, via a standard UDP socket for example, can be read from the file by any (authorized) user process, while packets written to the file are delivered by the kernel as if they had arrived over the network interface. Each mobile station has a virtual IP address (e.g., 11.0.0.1) associated with the network interface of the tunnel, and a physical IP address (e.g., 10.0.0.1) associated with the network interface of the physical wireless device. The application communicates using virtual IP addresses. The kernel IP routing table in each mobile station is configured to forward packets with virtual destination addresses to the IP tunnel device. At the source of a transmission, the packet sent from the application is forwarded through the IP tunnel to the routing protocol reading the device file. The routing protocol then converts the virtual addresses to physical addresses and selects the next hop to forward the packet to according to its routing table. Packets are forwarded to their neighbors using UDP sockets through the physical (wireless) network device. Once the packet reaches its destination, the physical addresses are translated back into virtual addresses and the routing protocol writes the packets to the device file that represents the IP tunnel, which then deliver the packet to the application via the virtual network interface.

2.2. The Wireless Network Simulator

We developed a high-performance simulator called SWAN as an integrated, flexible, and configurable environment for evaluating different wireless ad-hoc routing protocols, especially in large network scenarios. SWAN is built based on a parallel discrete-event simulator called DaSSF, which has proved successful in simulating large-scale wired networks.[2] We ported and implemented several protocol models that are used frequently in a wireless ad-hoc network. The protocol models can be readily assembled into a protocol stack within each simulated mobile station. Using SWAN, one can dynamically configure each protocol and the underlying wireless network using a specially designed configuration language.

In this paper, we study the effect of various radio signal propagation models on the behavior of the routing algorithms in simulation. In particular, we examine three simple but frequently used stochastic radio propagation models: a Friis free-space model, a two-ray ground reflection model, and a generic propagation model. The Friis free-space model assumes an ideal radio propagation condition: the signals travel in a vacuum space without obstacles. The power loss is proportional to the square of the distance between the transmitter and the receiver. The two-ray ground reflection model adds a ground reflection path from the transmitter to the receiver. The model is more accurate than the free-space model when the distance is large and when there is no significant difference in elevation between the mobile stations. The generic propagation model describes the radio signal attenuation as a combination of two effects: small-scale fading and large-scale fading. The small-scale fading captures the characteristic of rapid fluctuation in signal power over a short period of time or a small change in the node's position—a result primarily due to the existence of multiple paths that the signals travel. The classic models that predict the small-scale fading effect include Rayleigh and Ricean distributions. Large-scale fading is mostly caused by the environmental scattering of the signals and can be further divided into two components: the distance path loss is the average signal power loss as a function of distance and is proportional to the distance raised to a specified exponent; the shadow fading effect describes the variations in signal receiving power measured in decibels and can be modeled as a log-normal distribution. Readers can refer to a textbook on wireless communications (such as Rappaport's book [11]) for a detailed discussion on the radio propagation models.

3. Direct Execution

In simulation, multiple instances of a routing protocol must run simultaneously, driven by the same event queue. Conceivably, each routing protocol can run as a separate process and interact with the simulation kernel through inter-process communication mechanisms. We only need to substitute the system calls related to either communications (i.e., sending or receiving packets) or time (e.g., querying for the current wall-clock time or potentially blocking the user process causing noticeable delays) with calls to the simulator. The replacement can be done at link time after compilation. The major attraction of this approach is that no source code modification is necessary. The drawback, however, lies in its complexity and the potential overhead introduced by the inter-process communication.

[2] http://www.cs.dartmouth.edu/research/DaSSF.

We chose an easier yet faster approach that allows multiple instances of the same routing protocol to execute in the same address space. The method involved moderate modifications to the source code. Similar approaches can be found in the literature [1, 8, 9]. We ported all five ActComm routing protocols together with related programs, such as the application traffic generator used in the real experiment. The number of lines changed accounts for only 3.8% and most changes were related to creating and configuring the routing protocols individually in each simulated mobile station, separated from the protocol's control flow.

3.1. Encapsulations

We modified the protocol code slightly to allow multiple instances of a routing protocol to run simultaneously inside the simulator. Since all these instances are executed in the same address space, we need to provide wrappers so that these instances can be identified and separated in the same execution environment.

We created a protocol session object to represent each routing protocol instance in the simulator. The protocol's interaction with the operating system, such as system calls for sending and receiving packets, was replaced by method invocations of the protocol session. These methods redirect the calls to simulator. We also replaced global variables, which are data objects specific to a routing protocol instance, with member data of the protocol session. We replaced the original `main` function in the routing protocol implementations with a method of the protocol session that configures and initializes the instance.

3.2. Communications

The routing protocol implementations use system calls for communications, such as `sendto` for sending messages through UDP sockets. As mentioned earlier, we replaced these system routines with those supplied by the simulator. Rather than replacing them manually at all places of the source code, we provided a base class that contains methods with the same names as the system routines and with the same parameters. In this way, all classes in the protocol implementations default to call the methods in the base class. The base class contains a reference to the protocol session that represents the routing protocol instance. The methods in the base class forward control through the reference to the protocol session, which passes on the messages through the simulated protocol stack.

We added support in the simulator for UDP sockets. A UDP protocol session manages the UDP sockets on top of the IP layer and its primary function is to multiplex and demultiplex UDP datagrams. We replaced system calls related to UDP sockets, such as `socket`, `bind`, `sendto`, `recvfrom`, and `setsockopt`, with methods that interact with the UDP protocol session. We also implemented the IP tunnel device in the simulator. The device is treated as a network interface below the IP layer in the protocol stack. Packets sent by the application with virtual destination addresses (via UDP sockets) are diverted to the tunnel device by the IP layer. The routing algorithm accesses the IP tunnel through a regular file descriptor. We replaced the file access functions, specifically `open`, `read`, `write`, and `close`, to distinguish the file descriptor for the tunnel device from other regular files. We did not replace operations to regular files since they are used by the directly executed code for logging purposes.

3.3. Timings

The routing protocols executed inside the simulator must be driven by simulation time rather than real time, which means that we must deal with all time-sensitive system calls carefully. We replaced `gettimeofday`, which returns the wall-clock time of the mobile station, with a call to the simulator querying for the current simulation time. We also replaced `select`, which causes the running process to be blocked until any one of the specified set of file descriptors is ready for reading or writing, or the given timeout interval has been elapsed. The ActComm protocol implementations all center on an event loop that contains one call to the `select` function. When the control returns from this function—upon timeouts or incoming messages—the algorithm invokes the corresponding event handlers to process the event. We bypassed the the event loop and directly invoked the event handlers whenever a timeout occurred or a message arrived at the protocol session.

One also has to be aware of the ramifications from the lack of a CPU work model in the wireless simulator. The simulator uses function invocations for packets to travel up and down the protocol stack, without advancing the simulation time. This bears no side-effect for a carefully designed protocol model, where the packet processing time is simulated with proper random delays, but may create problems for a directly executed protocol implementation that pays no special attention to the packet processing time. If in simulation we assume zero packet processing time, the behavior of all instances of a routing protocol could be synchronized in simulation time. This time synchrony could then lead to an unnaturally high probability of packet loss due to collisions at the radio channel. To deal with this problem, we introduced packet jitters at the interface between the simulator and the directly executed code. Each time a message goes through a UDP socket, we added a random delay to model the time needed by the operating system for processing the packet.

4. Support for Simulation Validation

In this section we discuss our support to validate a wireless simulation by comparing results from the real experiment and the direct-execution simulation. Validation of simulations in general and wireless ad-hoc network simulations in particular has been a focal point surrounding the applicability of simulation studies. Johnson first suggested using the logging information from running ad-hoc routing algorithms during the real experiment to simulate identical node movement and communication scenarios [4]. Takai et al. studied the effect of wireless physical layer and radio channel modeling on the performance evaluation of ad-hoc routing algorithms [12, 13].

Our research uses real experiments as the base for comparison. In particular, we ran the routing protocol implementations together with other applications, such as the traffic generator, directly in the simulator. We derived both mobility and radio connectivity traces from the real experiment and combined them with a stochastic propagation model in an attempt to recreate the real network conditions in simulation. We compared the results against those from the real experiment to assess the validity of the radio propagation model.

In all five ActComm routing protocol implementations we embedded a sophisticated logging mechanism, as shown in Figure 1. When the routing protocol runs, it generates an event log that includes all types of events related to the routing algorithm, such as sending or receiving a control message. We used the event log both for analyzing the performance of the routing algorithm and for debugging. We also instrumented the traffic generator with logging functions to record each packet sent and received. We later used this application log to calculate application-level statistics, such as packet delivery rate and end-to-end delay. The traffic generator executed directly in the simulator also read this log to recreate the exact traffic behavior.

In the real experiment, we ran a third program called the *service module* together with the routing protocol and the application traffic generator. The program periodically queried the attached GPS device at the mobile station to log its current position. The program also used `iwspy` to periodically record link quality information. `iwspy` allows the user to set a list of network addresses. The wireless device driver gathers the link quality information, in signal-to-noise ratio (SNR), whenever a packet is received from one of those addresses, that is, from any other laptop. The service module collected the link quality information and averaged it over the last sampling interval. Also, it periodically broadcasted beacon messages that contain position information of all known mobile stations. The original ActComm applications used them to keep every soldier in the field updated with the positions of other soldiers. We recorded the beacon messages and used them to refresh the link quality information.

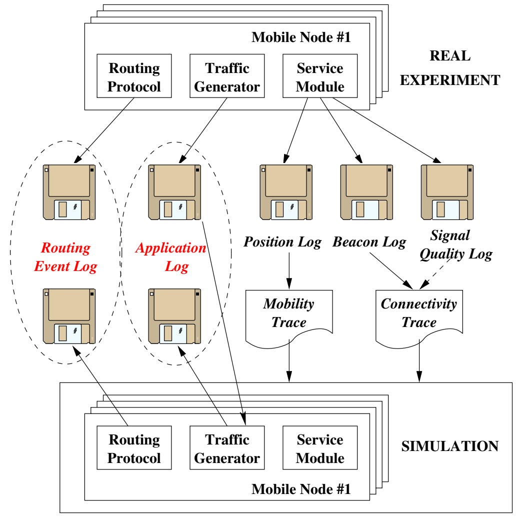

Figure 1. Logs are generated and compared for validating simulation results.

In simulation, the routing protocols are running directly inside the simulator together with the application traffic generator and the service module. We chose to directly execute the service model since we need to reproduce the beacon messages and their effect on the MAC/PHY states of the wireless network.

We further processed the position log from the real experiment to produce a mobility trace, which shows how each mobile station moved during the experiment. In addition, we generated a node connectivity trace from the beacon logs recorded by the mobile stations during the real experiment. The mobility trace states whether a mobile stations can receive a packet from another mobile station over the wireless channel at a given time. The beacon log contains the times at which the beacon messages from other mobile stations were received. Receiving a beacon successfully indicates a link from the sender to the receiver, while missing several consecutive beacons indicates that the receiver may be beyond the transmission range of the sender.

The signal quality log recorded a series of averaged signal-to-noise ratios for packets received at each mobile station from other stations in the network. We did not include the signal quality log in this study. We are currently investigating the use this log to reconstruct the connectivity of the network, as it may provide a better alternative to the beacon log.

We used the radio connectivity trace as a baseline to determine whether two mobile stations could directly communicate with each other. The connectivity information, however, does not capture the state of interference—collisions could happen due to the presence of "hidden terminals." For example, if node B can hear both node A and node C situated on either side, but node A cannot talk to C and vice versa because of the distance, it is possible that node B cannot faithfully receive a packet from A if node C is transmitting another packet to node B. Although the 802.11 MAC layer protocol, which arbitrates packet transmissions over the radio medium, allocates the radio channel before each transmission, it cannot totally prevent collisions. In this case, the simulator must use an interference model to simulate what would happen when two packets arrive at the receiver—one of the packets can be accepted if its receiving power is significantly higher than the other, or both packets can be lost due to interference. Since the interference model relies on the receiving signal power to determine packet receptions, we still need a radio propagation model to simulate the signal power attenuation.

5. Performance and Validation Studies

We conducted two experiments for validation: one comparing the direct-execution simulation of the ActComm AODV protocol implementation with an AODV protocol model implemented natively in the simulator, and the other comparing a real experiment with the simulated wireless network.

5.1. AODV vs. AODV

Our first experiment compared the direct execution of the ActComm AODV protocol implementation with an AODV protocol model implemented natively in SWAN. We ran both protocol implementations in simulation under the same simulated network conditions, with the same application traffic pattern, and the same radio propagation model. Our goal is to validate both protocol implementations against each other and determine how much overhead direct-execution simulation requires.

In the simulation experiment, we tested a network of 50, 100, and 200 mobile stations, out of which we chose 20 mobile stations as traffic sources. We deployed these mobile stations in a square area, sized so that each mobile station had seven neighbors on average (796, 1126, and 1592 meters for each dimension, respectively). We used the random way-point node mobility model: each node moves to a randomly selected point in the area with a speed chosen uniformly between 1 and 10 meters/s; when reaching the point, it pauses for 60 seconds before selecting another point to move to. We chose the IEEE 802.11 protocol for the MAC and PHY layer with standard parameters according to the IEEE specification (with 11 MB/s bandwidth), and we used the generic radio propagation model (with an exponent of 2.5 and shadow fading log-normal standard deviation of 6 dB) to compute radio signal power attenuation. We used a simple application traffic generator: each source periodically sends one packet (of 1 KB in size) to a randomly selected peer with an exponentially distributed inter-arrival time.

The behaviors of the two implementations differed slightly owing to variations in treatment of the AODV specifications. In addition, the ActComm AODV protocol ran in user space using IP tunneling and UDP sockets, while SWAN AODV ran directly on top of IP. The messages from the application traffic generator, when delivered to the ActComm AODV protocol through the IP tunnel, were wrapped with UDP and IP headers. Both the data and control messages used by ActComm AODV were also augmented with UDP headers through UDP sockets. Nonetheless, we found that, with varying traffic load, the overall packet ratio—which is the total number of packets received by the application layer divided by the total number of packets sent—differed only slightly between these two implementation (less than 3%). The similarity in the behavior of the two implementations ensures that using the two implementations to assess the cost of direct execution is meaningful.

Figures 2 and 3 show the difference in total execution time and peak memory usage between the two implementations of the AODV protocol. Clearly, the ActComm AODV (direct-execution) implementation requires more computational resources, but marginally so. The greatest increase in the execution time (about 18%) is at larger network size and heavier traffic load. The increased execution time is mostly caused by the overhead of copying and serialization of real packets. The memory overhead of ActComm AODV (over 100%) is more significant. We attribute it to the additional data structures used by the direct-execution protocol session, the IP tunnel device, and the UDP socket layer, which are proportional to the number of simulated mobile stations. Moreover, in simulation, the directly executed routing protocol and the application send and receive real packets with real message headers and real payloads. The overhead grows with increasing traffic intensity as packets stay longer in the wireless network due to more contentions.

In conclusion, direct-execution simulation requires more computational resources, especially in memory usage. The benefit of directly executing a routing protocol implementation in simulation is the assurance that the protocol implementation exhibits the same behavior as in a real network. A routing protocol model implemented natively in the simulator, however, may benefit from computational optimizations such as eschewing actual message headers and payloads. Thus, a protocol model, once validated, can be used in situations where the resource requirement is critical, such as in a simulation of a large-scale wireless network. On the other hand, the extra costs of direct-execution are not so onerous that it disqualifies the technique as a means of experimen-

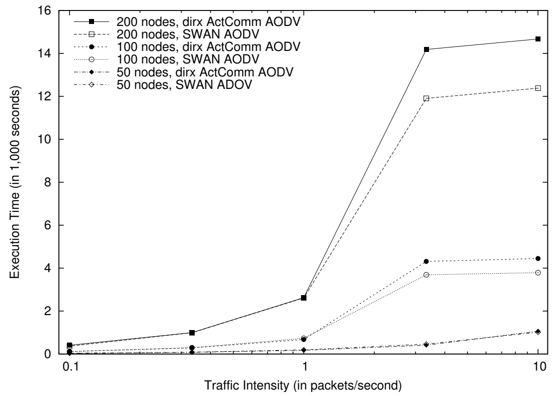

Figure 2. The simulation time for the two AODV implementations with varying traffic load (in log scale).

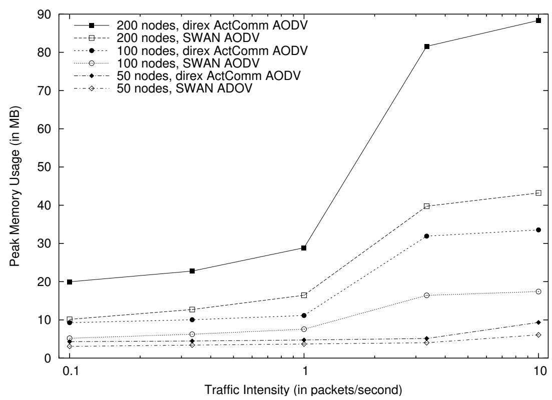

Figure 3. Peak memory usage by the two AODV implementations with varying traffic load (in log scale).

tation. There are obvious advantages to maintaining a common code base between a protocol's actual implementation, and that used to study its behavior in a simulator.

5.2. Simulation vs. Reality

As the second step in our validation, we compared the results from an outdoor routing experiment with our simulation results. In particular, we compared the results from the real experiment with the simulation results using different radio propagation models. The purpose of this study is to reveal the sensitivity of the performance of the routing protocols to the underlying wireless models.

5.2.1 The Real Experiment

The outdoor routing experiment took place on a rectangular athletic field measuring approximately 729 by 1408 feet (or 222 by 429 meters). Each of the 40 laptop computers used in this experiment had a Lucent (Orinoco) 802.11B wireless card operating in peer-to-peer mode at 2 MB/s. 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 10 meters during the experiment.

For this particular outdoor experiment, we included APRL, AODV, ODMRP and STARA (GPSR was still under development). The laptops, whose clocks were set to the time reported by the GPS unit, automatically ran each routing algorithm for 15 minutes, with two minutes of network quiescence between each algorithm to handle cleanup and setup chores. After each routing algorithm had been running for one minute, providing time to reach an initial stable routing configuration, the laptops automatically started a traffic generator that generated "streams" of UDP packets. The number of packets in each stream was Gaussian distributed with mean 5 and standard deviation $\sqrt{2}$; the time between streams was exponentially distributed with mean 15 seconds; the time between packets inside a stream was exponentially distributed with mean 3 seconds; every packet contained approximately 1200 data bytes; and the target laptop for each stream was uniformly randomly selected from among the other laptops. We chose these numerical parameters to approximate the (moderate) traffic volume observed during an earlier demonstration of a military application. The routing algorithm parameters, such as the beacon interval for APRL and the forwarding group lifetime for ODMRP, were set to "standard" values taken from the literature and our own experience.

During the course of the experiment, the laptops were continuously moving. The athletic field was divided into four equal-sized quadrants, one of which was approximately eight feet lower in elevation than the rest of the field. The hills from the higher to lower elevation were steep and short, and thus did obstruct the wireless signal, increasing the frequency with which the routing algorithms needed to find a multi-hop route. At the start of the experiment, the 40 participants were divided into equal-sized groups of 10 each, each of which was instructed to randomly disburse in one of the four quadrants. The participants then walked continuously, always picking a quadrant different than the one in which they were currently located, picking a random position within that quadrant, walking to that position in a straight line, and then repeating. This approach was chosen since it was simple, but still provide continuous movement to which the routing algorithms could react, as well as similar laptop distributions across each of the four routing algorithms.

Each laptop recorded extensive logs as described in Section 4. At the end of experiment, we discovered that seven laptops failed to generate any data or routing traffic due to misconfiguration or hardware problems. Thus, the experiment, in practice, reduced to a 33-laptop experiment and the logs from these 33 laptops were used as the starting point for comparing the real-world and simulated results.

5.2.2 The Simulation

We processed the logs from the real experiments to derive the mobility and radio connectivity traces for each laptop for the duration of running each routing algorithm. We ran the simulation for each algorithm for the designated period. We directly ran the routing protocol and the service module in each simulated mobile station. We modified the application traffic generator to read the application log and generate the same packets as in the real experiment. We focused only on the 33 laptops that actually transmitted, received, and forwarded packets in the real experiments. To reproduce the traffic pattern in simulation, the application traffic generator on each of the 33 nodes still included the 7 crashed nodes as their potential packet destinations.[3]

The mobile stations in simulation followed the mobility trace generated from the real experiment. 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. We used the propagation models to determine the power loss for each packet transmission and calculate the signal-to-noise ratio to quantify the state of interference at the receiver—whether a packet that arrived at a mobile station could be received successfully, or dropped due to significant power loss or collisions. We combined the three models with the connectivity trace derived from the beacon logs, leading to six different radio propagation models in simulation: three using the connectivity traces and the other three not. In the first three cases, we used the connectivity trace to determine whether a packet from a mobile station could reach another mobile station, and then we used the radio propagation models to determine the receiving power for the interference calculation. Comparison of models with measured connectivity with those without give us a means of determining whether the model contains accurate predictive power for connectivity.

5.2.3 The Results

We first examine the packet delivery ratio. Figure 4 shows the packet delivery ratio from the real experiment and the simulation runs with six radio propagation models (three of which used the connectivity trace derived from the real experiment to determine the reachability of the signals). Each simulation result is an average of five runs; the variance is insignificant and therefore not shown. The generic propagation model in the experiment used typical parameters to describe the outdoor environment of the real experiment: we used 2.8 as the path-loss exponent and 6 dB as the standard deviation for shadow fading.

[3] Therefore, the packet-delivery ratios, both from the real experiment and the simulation, should be lower than expected, since those packets with unknown destinations could not be delivered.

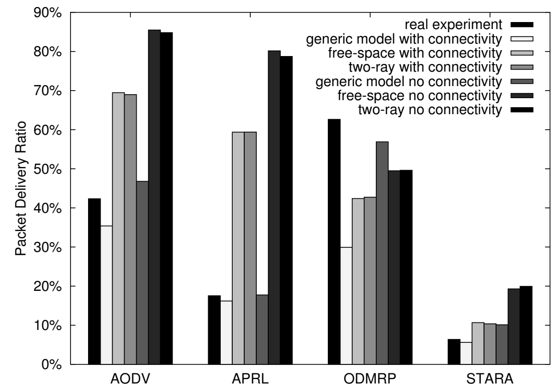

Figure 4. Comparing the data delivery ratio from the real experiment with various radio propagation models. "With connectivity" means the connectivity trace was used.

We found that the simple generic propagation model offered an acceptable prediction of the performance of the routing algorithms, although different propagation models predicted vastly different protocol behaviors. The difference is significant in some cases that could result in misleading conclusions, for example, when comparing the performance of AODV and ODMRP. The inaccuracy in the model prediction introduced by the propagation model is non-uniform and can undermine a performance comparison study of different protocols.

For AODV, APRL, and STARA, the figure shows a large exaggeration of the packet delivery ratio using the free-space model and the two-ray ground reflection model. Both models overestimated the transmission range of radio signals causing shorter routes and therefore better packet delivery ratio. Even with the connectivity trace, the models overestimated the signal quality, failing to capture the lossy characteristic of the radio propagation environment. The performance of ODMRP was underestimated in simulation. ODMRP is a multicast routing algorithm that delivers packets using multiple paths to their destinations. It has a higher demand on the network bandwidth. The overestimated transmission range and signal quality in the free-space and two-ray models caused more contentions and created a negative effect on the simulated throughput.

The packet delivery ratio does not reflect the entire ex-

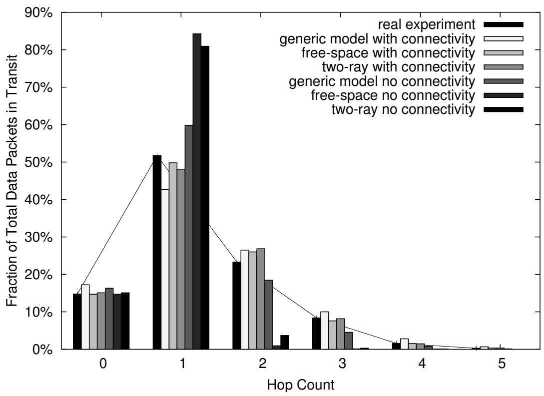

Figure 5. The hop-count histogram of AODV in real experiment and in simulation.

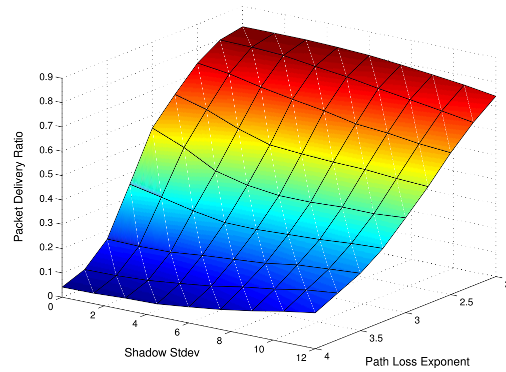

Figure 6. Sensitivity of AODV performance to parameters of large-scale fading model.

ecution environment of the routing algorithm. Figure 5 shows a histogram of the number of hops that a data packet traversed in AODV, before it either reached its destination or dropped along the path. For example, a hop count of zero means that the packet was dropped at the source node; a hop count of one means the packet went one hop: either the destination was the source's neighbor or the packet failed to reach the next hop. The figure shows the fraction of the data packets that traveled in the given number of hops. We see clearly the free-space and two-ray models resulted fewer hops by exaggerating the transmission range. 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.

The generic propagation model with typical parameters to represent the outdoor test environment offered a relatively good prediction of the performance of the routing algorithms. However, one must carefully choose the correct parameters to reflect the wireless environment. The exponent for the distance path loss and the standard deviation in log-normal distribution for the shadow fading are heavily dependent on the environment under investigation. In the next experiment, we ran a simulation with the same number of mobile stations and with the same traffic load as in the real experiment. Figure 6 shows AODV performance in packet delivery ratio with the same network setting but varying the path-loss exponent and the shadow log-normal standard deviation.

The AODV behavior was more sensitive to the path-loss exponent than to the shadow standard deviation. That is, the signal propagation distance had a stronger effect on the algorithm's performance. A shorter transmission range means packets must travel through more hops (via longer routes) before reaching its destination, and therefore has a higher probability to be dropped. A larger shadow standard deviation caused the links to be more unstable, but the effect varied. On the one hand, when the path-loss exponent was small—the signals had a long transmission range, the small variation in the receiving signal strength did not have a significant effect on routing, causing only infrequent link breakage. On the other hand, when the exponent was large, most nodes were disconnected. A variation in the receiving signal power helped establish some routes which were impossible if not for the signal power fluctuation. Between the extremes, a larger variation in the link quality generally caused more transmission failures, and therefore resulted slightly lower packet delivery ratio.

The critical implication of this sensitivity study is that we cannot just grab a set of large-scale fading parameters, use them, and expect meaningful results for any specific environment of interest. On the one hand, pre-simulation empirical work to estimate path-loss characteristics might be called for, if the point of the experiment is to quantify behavior in a given environment. Alternatively, one may require more complex radio models (such as ray-tracing) that include complex explicit representations of the domain of interest. 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.

To summarize, we used simple stochastic radio propagation models and the traces generated from a carefully designed real experiment. Direct-execution simulation provided a common baseline for comparing the behavior of routing protocols both in the real experiment and in simulation. We found that it is critical to choose a proper wireless model that reflects a real-world scenario for studying the performance of ad-hoc routing algorithms. In contrast to earlier studies [12], we found that using a simple stochastic radio propagation model with parameters typical to the outdoor environment can produce acceptable results. We must recognize, however, the results are sensitive to these parameters. It is for this reason we caution that the conclusions drawn from simulation studies using simple propagation models should apply only to the environment they represent. The free-space model and the two-ray model, which exaggerate the radio transmission range and ignore the variations in the receiving signal power, can largely misrepresent the network conditions.

6. Conclusions

This paper reports our effort to support direct-execution simulation of a set of wireless ad-hoc routing protocols to facilitate validation of wireless network models.

In an experiment, we compared two implementations of the AODV protocol: one with direct execution and the other implemented natively in the simulator. We found that direct-execution simulation requires more computational resources, especially in memory usage, thus making the modeled protocol more attractive in a resource-constrained situation, such as studying protocol behaviors in a large network environment. The CPU overhead of direct-execution, however, is moderate and in most case cannot keep direct-execution simulation from being a valuable means of experimentation with the obvious advantage of maintaining consistency between a protocol's actual implementation and that used in simulation.

We conducted a real experiment running the protocols on 40 laptop computers in an outdoor environment. We embedded a sophisticated logging mechanism in the protocol implementations. All activities related to the routing algorithms and the applications were recorded in files. Post-processing these files results in traces that we used in simulation to reproduce the same network condition. We found that one can use a simple stochastic radio propagation model to predict the behavior of the routing protocols with fairly good accuracy, but the results are quite sensitive to the model's parameters. We argue that choosing a proper wireless model that represents the wireless environment of interest is critical in performance evaluation of the routing algorithms.

Our future work includes further analysis to validate different wireless models under different real experimental conditions. We are currently investigating using the link quality information collected by the wireless device driver to improve the accuracy of the connectivity trace. Also, we want to translate the terrain information of the real experiment into a radio propagation gain matrix for a more realistic representation of the wireless environment, and study the effect of such modeling details on the performance evaluation of wireless ad-hoc routing protocols.

7. Acknowledgments

We thank Nikita Dubrovsky, Aaron Fiske, Chris Masone, and Michael DeRosa for implementing the routing algorithms and most of the outdoor experiment infrastructure. Piyush Gupta and Brad Karp helped with the source code of STARA and APRL. Dennis McGrath helped with the initial class structure for the routing algorithms. We thank Chip Elliott at BBN for his valuable insights and suggestions. We also thank Lisa Shay, Susan McGrath, and Eileen Entin for designing the application scenario, and the sixty Dartmouth students and staff members who participated the outdoor experiments.

References

[1] X. A. Dimitropoulos and G. F. Riley. Creating realistic BGP models. *MASCOTS'03*, October 2003.
[2] P. Gupta and P. R. Kumar. A system and traffic dependent adaptive routing algorithm for ad hoc networks. *36th IEEE Conference on Decision and Control*, pages 2375–2380, December 1997.
[3] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of details in wireless network simulation. *SCS Multiconference on Distributed Simulation*, pages 3–11, January 2001.
[4] D. B. Johnson. Validation of wireless and mobile network models and simulation. *DARPA/NIST Network Simulation Validation Workshop*, May 1999.
[5] B. Karp and H. T. Kung. Dynamic neighbor discovery and loopfree, multi-hop routing for wireless, mobile networks. Hardvard University, May 1998.
[6] B. Karp and H. T. Kung. Greedy perimeter stateless routing for wireless networks. *MobiCom'00*, pages 243–254, August 2000.
[7] S. J. Lee, M. Gerla, and C. C. Chiang. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications*, 7(6):441–453, December 2002.
[8] J. Liu, L. F. Perrone, D. M. Nicol, M. Liljenstam, C. Elliott, and D. Pearson. Simulation modeling of large-scale adhoc sensor networks. *European Simulation Interoperability Workshop (Euro-SIW'01)*, June 2001.
[9] M. Neufeld, A. Jain, and D. Grunwald. Nsclick: bridging network simulation and deployment. *MSWiM'02*, pages 74–81, September 2002.
[10] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.
[11] T. S. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall, 1996.
[12] M. Takai, R. Bagrodia, K. Tang, and M. Gerla. Efficient wireless network simulations with detailed propagation models. *Wireless Networks*, 7(3):297–305, May 2001.
[13] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. *MobiHoc'01*, pages 87–94, October 2001.