Empirical Validation of Wireless Models in Simulations of Ad Hoc Routing Protocols

Jason Liu
Department of Mathematical and Computer Sciences
Colorado School of Mines
Golden, CO 80401
xliu@mines.edu

Yougu Yuan
David M. Nicol
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, IL 61801

Robert S. Gray
Calvin C. Newport
David Kotz
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory, Hanover, NH 03755

Luiz Felipe Perrone
Department of Computer Science
Bucknell University
Lewisburg, PA 17837

Computer simulation has been used extensively as an effective tool in the design and evaluation of systems. One should not, however, underestimate the importance of validation—the process of ensuring whether a simulation model is an appropriate representation of the real-world system. Validation of wireless network simulations is difficult due to strong interdependencies among protocols at different layers and uncertainty in the wireless environment. The authors present an approach of coupling direct-execution simulation and traces from real outdoor experiments to validating simple wireless models that are used commonly in simulations of wireless ad hoc networks. This article documents a common testbed that supports direct execution of a set of ad hoc routing protocol implementations in a wireless network simulator. By comparing routing behavior *measured* in the real experiment with behavior *computed* by the simulation, the authors validate the models of radio behavior upon which protocol behavior depends.

Keywords: Wireless network simulation, direct-execution simulation, trace-driven simulation, simulation verification and validation

1. Introduction

Computer simulation has become the primary tool for evaluating the performance of routing protocols in mobile ad hoc networks (MANET), as manifested by the increasing number of research papers published in conferences (such as MobiCom and MobiHoc) that use simulation to conduct performance evaluations. In contrast to real field experiments, which generate unrepeatable results and are difficult to realize, simulation provides a controlled environment for testing the design of routing protocols in a network that can be easily scaled up to thousands or even millions of mobile stations. Simulation enables fast exploration of the design space of the routing protocols under different network conditions—the geographic environment, the user mobility pattern, the application traffic load, and so on.

Using simulation, one must remember that the model may not reflect reality. Validation is a process that determines whether a simulation accurately represents the target system (see Law and Kelton [1] for a detailed discussion on this subject). Validation of MANET

SIMULATION, Vol. 81, Issue 4, April 2005 307-323

DOI: 10.1177/0037549705055017

simulations is particularly difficult because not only must the implementation of a simulated protocol be *verified* against its design specifications, but the model must also be able to capture lower-level characteristics of the wireless environment with a proper level of abstraction [2]. It is known that the performance of the ad hoc network protocols heavily depends on the performance of lower layers of the protocol stack (i.e., the MAC and PHY layers) as well as the wireless channel [3, 4]. Complex models that capture high-level details of the lower protocol layers and the RF propagation and interference can certainly provide more confidence in the validity of the simulation results. However, these models incur expensive computation, which is unlikely to be accommodated by using parallel simulations alone. As a consequence, we find the MANET community diverges in the use of wireless models in the simulation [5]. Questions remain on what would be an appropriate level of abstraction (for a particular objective of a simulation study) and whether one can use simple models as a viable approach to obtain fairly accurate results, particularly in simulations of large-scale mobile ad hoc networks.

We set out to address these problems, using direct-execution simulation and real field measurements to validate wireless models, particularly simple stochastic RF models. Direct-execution simulation alleviates the need to maintain separate code bases for the 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. The protocol's logic is executed inside the simulator and is driven by the simulator's time-advancing mechanism. Particularly in an event-driven simulation paradigm, 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 also desirable for prototyping a protocol implementation, which, after initial simulation evaluation, can be deployed directly in a real network.

In this article, we are interested in the ability of direct-execution simulation to help us bypass the *verification* stage of the routing protocols in simulation—the process determining whether the computer model accurately represents the conceptual description and the actual implementation of the protocols. This paves the way for us to validate the wireless models in the network simulator using measurements from real field experiments. We run the same routing protocol and application traffic generator code both in simulation and in the real experiment. We also include in the simulation a detailed model of the IEEE 802.11 MAC layer protocol—the same protocol is used in the real experiments. The model was originally ported from GloMoSim [6], which has been used widely in the research community and we assume to be accurate. In a real experiment, packets are transmitted via the wireless channel and are subject to delays and potential losses due to signal fading and collision. 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. Such comparison provides us a valuable opportunity to investigate the effect of details of wireless models on the fidelity of a simulation study.

More specifically, this article documents our effort in supporting direct execution of a set of wireless ad hoc routing protocol implementations and using the direct-execution simulation to validate the underlying wireless models by comparing the results from the simulations and the real-world experiments. We ported five routing protocol implementations for direct execution: APRL [7], AODV [8], GPSR [9], ODMRP [10], and STARA [11]. Versions of all five protocols were implemented as part of the ActComm project, the goal of which is to provide information access through a wireless network to soldiers in the battlefield (http://actcomm.thayer.dartmouth.edu/). We created a common testbed for direct execution of these protocols in simulation, and we instrumented the testbed to include various logging functions in the routing protocol code. We did two large outdoor experiments over the course of 2 years. In the first experiment, we ran four routing protocols on 40 laptop computers. In the second experiment, we had 22 laptop computers. The laptops were carried by people walking randomly in an outdoor athletic field. Each laptop computer had a Global Positioning System (GPS) device and periodically recorded 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 mimic the test scenario inside the simulator. We compared the results from running these routing protocols in simulation with the measurements collected from the real experiments to reveal the effect of different wireless models on the behavior of ad hoc routing algorithms.

The contributions of this article are threefold. The first is the development of the testbed that facilitates the validation of wireless models in mobile ad hoc network simulations. The testbed supports direct execution of a set of ad hoc routing protocol implementations in a wireless network simulator. In particular, the testbed can read traces generated from real experiments and use them to drive direct-execution implementations of the routing protocols. Doing so, we reproduce the same network conditions as in real experiments for our validation study. Our second contribution is the use of extensive measurements from two carefully designed outdoor experiments to validate the simple wireless models popular in the MANET community. By comparing routing behavior *measured* in real experiments

with behavior *computed* by the simulation, we are able to isolate and therefore validate the models of radio behavior upon which protocol behavior depends. We are the first in doing so. Our third contribution is the recommendations made from the validation study for the use of simple wireless models. We conclude that it is possible to have fairly accurate results using a simple stochastic RF model, but the routing behavior is quite sensitive to one of this model's parameters. The implication is that one should (1) use a more complex (and more computationally expensive) radio model that explicitly models point-to-point path loss, (2) carefully parameterize the model using measurements from an environment typical of the one of interest, or (3) study behavior over a range of environments to identify sensitivities.

The article is organized as follows. Section 2 provides an overview of the implementations of the routing protocols and outlines the architecture of our wireless network simulator on which we directly executed 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 offers a discussion on the benefits and, more important, limitations of our approach. We provide a brief summary of related work in section 7 before we conclude the article in section 8.

2. Background

In this section, we provide an overview of the routing protocol implementations and the wireless network simulator that we adapted to directly execute protocol implementations.

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 [7]. 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 [8]. 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 [9]. GPSR uses a perimeter-following algorithm to forward packets around the boundaries of empty regions that contain no mobile stations (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 [10]. It does not depend on another unicast routing protocol and, in fact, can be used for unicast routing. The System and Traffic Dependent Adaptive Routing Algorithm (STARA) uses shortest-path routing [11]. The distance measure is calculated by the mean transmission delay instead of the hop count. An empirical comparison of four of these protocols can be found in Gray et al. [12].

Although APRL and STARA are not typical choices of ad hoc routing protocols, we believe our selection provides a representative subsection of the algorithmic design space. Both APRL and STARA are proactive routing protocols. Despite their differences in complexity, both protocols actively maintain routes to all other mobile stations. In contrast, both AODV and ODMRP are reactive routing protocols: routes are established on demand, subject to the traffic requirement. AODV and ODMRP differ primarily in ODMRP's ability to support multicast. GPSR, different from the approaches above, makes use of geographical information. Because of the diversity in the protocol selection, it is important that we build the direct-execution simulation testbed able to accommodate routing algorithms belonging to different algorithmic classes.

We (and others) implemented these protocols for the ActComm project in C++ on Linux. All five implementations performed their routing as user-level applications using IP tunneling and UDP sockets, as shown in Figure 1. An IP tunnel is a virtual network device with two endpoints: one as a regular network interface and the other as a Unix device file (in the `/dev` directory). Packets sent to the network interface, via a standard UDP socket, 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 had a virtual IP address associated with the network interface of the tunnel, as well as a physical IP address associated with the network interface of the physical wireless device. The application communicated using virtual IP addresses. We configured the standard kernel routing tables so that all packets destined to virtual IP addresses were forwarded to the IP tunnel device. At the source, a packet sent from the application was forwarded first through the IP tunnel to the routing algorithm reading the device file (`tun0` in Fig. 1). The routing algorithm then converted the virtual addresses to physical addresses and then selected the next hop to which to forward the packet in accordance with its current routing table. All packets were forwarded to their neighbors using UDP sockets through the physical (wireless) network device (`eth0` in Fig. 1). At an intermediate node, the UDP packet was received from the physical network device (`eth0`) and given to the routing algorithm through the UDP socket interface. The routing algorithm again selected the next hop (using the packet's physical IP address) and forwarded the packet to it using the UDP socket interface, which sent out the packet from the physical network device (`eth0`). Once a packet reached its destination, the physical addresses were translated back into virtual addresses, and the routing algorithm wrote the packets to the device file of the IP tunnel (`tun0`), which then delivered the packet to the application via the virtual network interface.

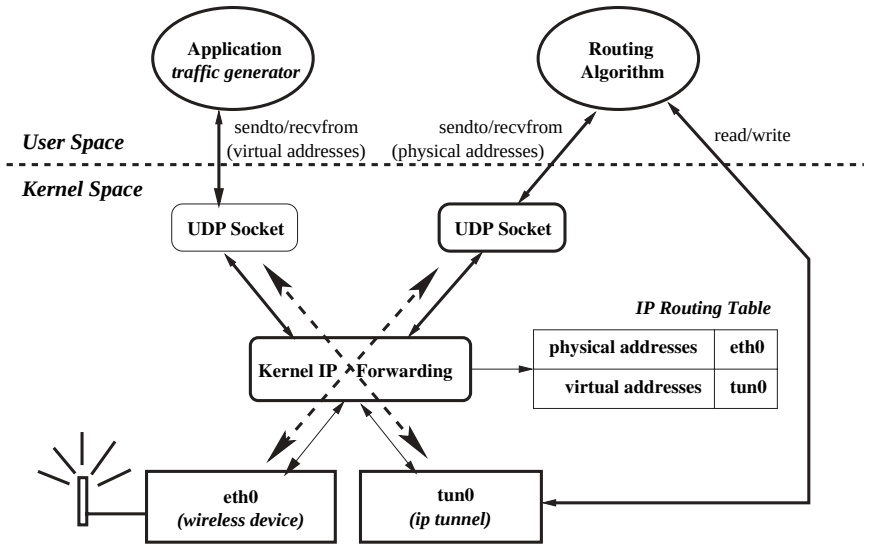

Figure 1. Implementation of routing protocols at user space using IP tunneling

Using IP tunneling and UDP sockets not only simplifies the development and testing of ad hoc routing protocols in *real* networks but also makes a straightforward transition from real-world implementations to direct-execution simulation, as we discuss in section 3. The drawback of this approach is the overhead associated with moving packets between the kernel and the user space. Although this performance penalty has no significant impact on the ActComm applications, designed to run on laptop computers with significant computing resources, it could become unwieldy for performance-critical applications running on less powerful hardware platforms. In this case, we should consider implementing the routing protocols in the kernel space to take advantage of optimizations unavailable at the user level. For the purpose of this validation study, we chose implementation simplicity over efficiency.

Another common feature in the ActComm implementations of the routing protocols is that they are all event driven. At the center of each routing protocol implementation is an event loop that dispatches callback functions in response to timeouts or packet arrivals. As we show later, these implementation features tremendously eased the transition of the routing protocols from real systems to the simulated environment.

2.2 The Wireless Network Simulator

We developed a high-performance simulator, called SWAN, as an integrated, flexible, and configurable simulation environment for evaluating different wireless ad hoc routing protocols, especially in large network scenarios. SWAN is based on DaSSF, a parallel discrete event simulator that has been proven successful in simulating large-scale wired networks (http://alamode.mines.edu/∼xliu/projects/dassf/). SWAN uses novel synchronization algorithms to achieve better performance on parallel platforms for large-scale wireless network simulations [13]. The detail of these synchronization algorithms is beyond the scope of this article.

Conceptually, the architecture of SWAN can be divided into two submodels: the environment model and the node model. The environment model consists of radio channel models, user mobility models, and geographical terrain information. The node model describes the software structure within a mobile station, which consists of a stack of protocol layers interacting through a standard interface. We ported and implemented models of several protocols that are used frequently in wireless ad hoc networks, such as the IEEE 802.11 wireless local-area network (LAN) protocol and AODV. These protocol models can be readily assembled as a protocol stack within each simulated mobile station. One can configure and change the properties of these protocols at runtime using a specially designed configuration language.

In this article, we study the effect of several radio signal propagation models on the behavior of the ad hoc 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: radio 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. This model

is more accurate than the free-space model when the distance is large and 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. 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 on which 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 [14]) for a detailed discussion on the stochastic radio propagation models.

One must understand that these simple models only provide "correct" radio propagation behavior in a statistical sense for particular wireless environments within their design perimeters. Specifically, they do not provide enough modeling details to represent signal propagation in a real environment and therefore cannot offer an exact match to the real experiment results. We elaborate this point in a later section.

3. Direct Execution

In this section, we describe the methods we used to directly execute the routing protocol implementations in the wireless simulation environment. 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 interprocess communication mechanisms. We only need to substitute the system calls related to either communications (such as sending or receiving packets) or time (such as querying for the current wall-clock time or blocking the user process) with calls to the simulator. The replacement can be done either at link time (using linker wrapper functions) or at runtime (by preloading dynamic linking libraries or using the packet-capturing facilities in the kernel). The major attraction of this approach is its generality and that no source code modification is necessary. The drawback, however, lies in its complexity related to and the potential overhead introduced by interprocess communications.

We chose a faster yet slightly more complex approach that allows multiple instances of the same routing protocol to execute in the same address space. The method involved only moderate modifications to the source code. It must be understood that our approach does not intend to be general but rather effective in enabling direct-execution simulation for a range of routing protocols implemented in a common framework for the ActComm project. The goal is to use direct-execution simulation to bypass the verification problem in cases where different programs are developed separately for real systems and for simulation. We ported all five ActComm routing protocols together with related programs, such as the application traffic generator used in the real experiments. The number of lines changed in the source code accounted for only 3.8% of the total. Most changes were repetitive and related to creating and configuring the routing protocols individually in each simulated mobile station and therefore were separate from the protocols' primary control flow.

3.1 Encapsulations

We modified the protocol code only slightly to allow multiple instances of a routing protocol to run simultaneously inside the simulator. Since multiple instances are expected to execute 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 the 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 in the routing protocol implementations with member data of the corresponding 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 a UDP socket. As mentioned earlier, we replaced these system routines with those supplied by the simulator. Rather than replacing them manually at all places in the source code, we provided a base class that contained methods with the same names as the system routines and with the same prototype. 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 then passes on the messages through the simulated protocol stack. This method is guaranteed to work as long as we make sure that all system routines we intend to replace are redefined properly in the base class and that they are called within the methods of the classes deriving from the base class in the protocol implementations.

We added support in the simulator for UDP sockets. A UDP protocol session master manages the UDP sockets on

top of the IP layer, whose primary function is to multiplex and demultiplex UDP datagrams. Using the class inheritance technique, 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`, whose function is to block the running process until any one of the specified set of file descriptors is ready for reading or writing or a given timeout interval has been elapsed. The ActComm protocol implementations all center on an event loop that contains only 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. To provide the same function in an event-oriented simulation worldview, we bypassed 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 delays but may create problems for a directly executed protocol implementation that pays no special attention to the packet processing. The ActComm implementations do not explicitly specify delays for packets that pass through system facilities (such as the IP tunnel). 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 synchrony could then lead to an unnaturally high probability of packet loss caused by collisions at the wireless channel. To deal with this problem, we introduced random 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. We also encountered a case in the STARA implementation where the lack of a work model caused an underflow in a floating-point calculation and threw the simulation into an infinite loop. At each iteration, the algorithm consistently chose to schedule an event with a zero delay. The problem would be corrected automatically in a real network since the wall-clock time advances independently. To solve the problem in simulation, we added a small jitter delay whenever the directly executed code scheduled a zero-delay event.

Note that using jitter delays does not provide an accurate representation of the use of computational resources needed by the real system to process the packets traversing through the protocol stack. These delays are especially important when the system is operating in a resource-limited environment, where an accurate CPU model is needed to simulate the packet-forwarding delays, which become significant in determining the performance of the entire system. In our real experiments, the ad hoc routing protocols are running on laptops with ample processing power. Therefore, an accurate modeling of the CPU consumption (which is costly to simulate) was never called for; the jitter delays were used in this case simply to model the randomness and asynchrony in the packet processing in the system.

4. Support for Simulation Validation

In this section, we discuss our support for validating a wireless simulation by comparing results from the real experiments and the direct-execution simulation. We developed the testbed to facilitate the validation of the stochastic radio propagation and interference models that are widely used in simulation studies in the MANET research community. It must be understood that there is no definitive approach to validating the models because (1) the RF environment in a real experiment cannot be reproduced exactly, and (2) the wireless models are stochastic and, if valid, can only produce statistically "correct" results that match the real experiment. Here, by validation we mean to find out how these wireless models affect the results used in performance evaluations of ad hoc routing algorithms.

Our approach used real experiments as the base for comparison. In particular, we ran the routing protocol implementations together with other applications directly in the simulator. We derived both node mobility and radio connectivity traces from the real experiment and combined them with a stochastic RF model in an attempt to re-create the real network conditions in simulation. We compared the results against those from the real experiment to assess the validity of the underlying wireless models. Since the application layer and the ad hoc routing protocols were the same in the real experiments and in simulation, and we assumed that the detailed model of the IEEE 802.11 at the

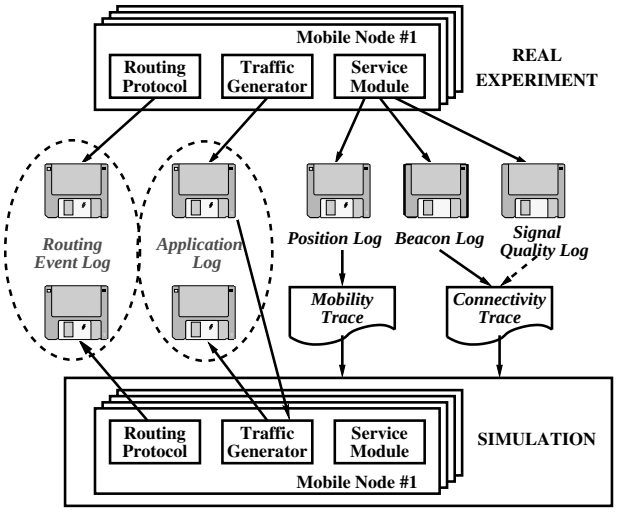

Figure 2. Logs are generated and compared for validating simulation results

MAC layer was accurate, our focus was on the fidelity of the radio propagation and interference models.

In all five ActComm routing protocol implementations, we embedded a sophisticated logging mechanism, as shown in Figure 2. When we ran the routing protocol, it generated an event log including all types of events related to the routing algorithm, such as sending or receiving a packet. We used the event log both for analyzing the performance of the routing algorithm and for debugging. The application traffic generator is a simple program running simultaneously with the routing protocols. The traffic generator models an on-off process: it waits for an exponentially distributed *off* period and, during the *on* period, randomly selects a target mobile station, to which it sends a number of data packets separated by exponentially distributed random intervals. We instrumented the traffic generator with logging functions to record every packet sent and received. We later used this log to calculate application-level statistics, such as packet delivery rate and end-to-end delay. Furthermore, when we directly executed the traffic generator in the simulation experiment, the traffic generator used this log to re-create the same traffic behavior as in the real experiment.

We also ran a third program, called the *service module*, in the real experiment 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 geographical location. The program also used `iwspy` to periodically record link quality information. `iwspy` allows the user to specify a list of network addresses.[1] The wireless device driver updated link quality information (i.e., the signal strength) whenever a packet was received from one of the listed addresses. The service module periodically collected the most recent values in the last sampling interval and recorded them in the signal quality log. Moreover, the service module periodically broadcasted beacon messages that contained 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. Since the beacon messages were sent periodically at low frequency, we expected that the perturbation was insignificant.

In the simulation experiment, the routing protocols were running directly as part of the simulator together with the application traffic generator and the service module. One might think it unnecessary to run the service module in simulation—the location of any mobile station is, after all, always available from the simulator's mobility model. We chose to directly execute the service module since we needed to reproduce the beacon messages and their effect on the state of the wireless network (particularly at the MAC and PHY layers). In this way, the simulation produced the same set of logs as in a real experiment.

Before simulation, we processed the position log from the real experiment to produce a mobility trace, which showed how each mobile station moved over time during the experiment. In addition, we generated a radio connectivity trace from the beacon logs recorded by the mobile stations during the real experiment. The connectivity trace states whether a mobile station can receive a packet from another mobile station over the wireless channel at any given time. We derived radio connectivity using the following method. 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. If node A could hear a beacon message from node B, we assume there was a direct link from node B to node A during the next sampling interval. After that, however, if node A did not receive the next beacon message, it only means that either node A moved out of the transmission range of node B or simply the beacon from B was dropped as a result of corruption or collision with another transmission. To deal with the latter case, we did not immediately remove the link from B to A but instead only did so when three beacon messages from B were missing in succession.

The signal quality log recorded a series of averaged signal-to-noise ratios for packets received at each mobile station. The signal quality log is not included in this study. However, as an alternative to the beacon log, this information could be used to reconstruct radio connectivity of the wireless network. The recorded link quality information is presumably better at capturing the signal propagation and interference scenario than the beacon logs since the

1. We made a minor modification to the standard Linux Card Manager services to increase the maximum number of tracked sources to accommodate the 40 laptops we used in the real experiment.

link quality information is collected at the wireless device driver (presumably at the signal level) regardless of whether an entire packet is successfully received.

Note that, even with the radio connectivity traces, reproducing the state of the wireless environment is difficult. An inappropriate radio channel model can produce misleading simulation results. To have a detailed trace that records every bit of change in the state of the wireless network would require an extremely large amount of storage space even for a short duration. And even if we successfully created a detailed trace, we would still face the problem if we wanted to introduce a slight modification to the application traffic behavior. We used the radio connectivity trace as a baseline to determine whether two mobile stations could directly communicate with each other. It should be noted that the connectivity information 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 they are out of each other's radio propagation range, it is possible that node B cannot faithfully receive a packet from A if node C is transmitting another packet to node B simultaneously. Although the 802.11 MAC layer protocol, which arbitrates packet transmissions over the wireless 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. It is possible that one of the packets can be accepted if its receiving power is significantly higher than the other, or both packets are lost due to the presence of interference.

Since the interference model relies on the receiving signal power to determine packet receptions, we need a radio propagation model to simulate the signal power attenuation. In the next section, we provide some preliminary results on the effect of three simple stochastic radio propagation models, with and without the connectivity trace, and study their effect on the behavior of the routing protocols.

5. Performance and Validation Studies

We conducted two sets of experiments. The first experiment compared the direct-execution simulation of the ActComm AODV protocol implementation with an AODV protocol model implemented natively in the SWAN simulator. This experiment was used to verify two independent protocol implementations against each other and to assess the cost of using direct-execution simulation in support of future studies using this method. The second experiment compared the results from two outdoor experiments and the simulation of a mobile network running multiple ad hoc routing algorithms. The goal of this experiment is to validate the wireless models and, more important, to reveal the sensitivity of the performance of the routing protocols to the wireless models used in simulations.

5.1 AODV vs. AODV

In this experiment, we 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 using the same radio propagation model. Our goal is to verify both protocol implementations against each other and determine how much overhead direct-execution simulation requires.

We tested a network of 50, 100, and 200 mobile stations, out of which we chose 20 mobile stations as traffic sources. Each traffic source randomly selected a target among other mobile stations and sent to it a packet of 1 KB in size before switching to another randomly selected target after an exponentially distributed random interval. 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 m/sec; when reaching the point, it pauses for 60 seconds before selecting another point to which to move. We chose the IEEE 802.11 protocol for the MAC and PHY layer with standard parameters according to the IEEE specification (with an 11-Mb/sec bandwidth), and we used the generic radio propagation model (with an exponent of 2.5 and shadow-fading lognormal standard deviation of 6 dB) to compute the signal path loss.

The behaviors of the two implementations differed slightly owing to variations in treatment of the AODV specifications. In addition, the ActComm AODV ran in user space using IP tunneling and UDP sockets, while the 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 the ActComm AODV were also augmented with UDP headers by UDP sockets. Nonetheless, we found that, with varying traffic load (by changing the mean packet interarrival time), the overall packet delivery ratio—the total number of packets received by the application layer divided by the total number of packets sent—differed only slightly between these two implementations, as shown in Figure 3. Both implementations achieved similar output (less than 3% difference). The similarity in the behavior of the two implementations ensures that using the two implementations to assess the cost of direct execution is meaningful.

Figure 4 shows 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 required more computational resources, but marginally so. The greatest increase in the execution time (about 18%) was at larger network sizes

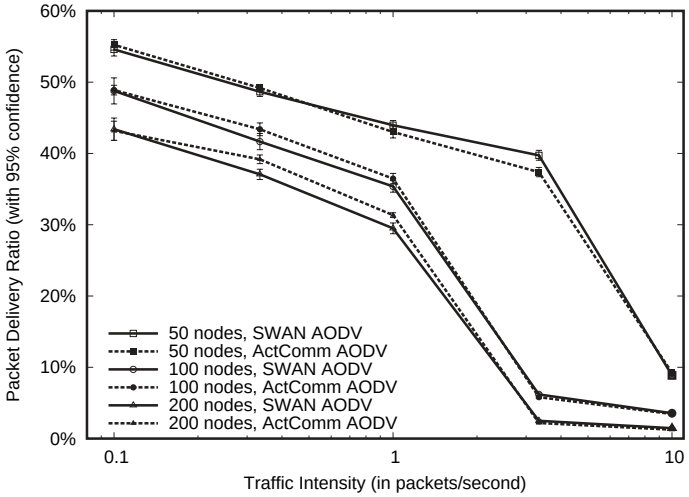

Figure 3. Packet delivery ratio with varying traffic load (in log scale)

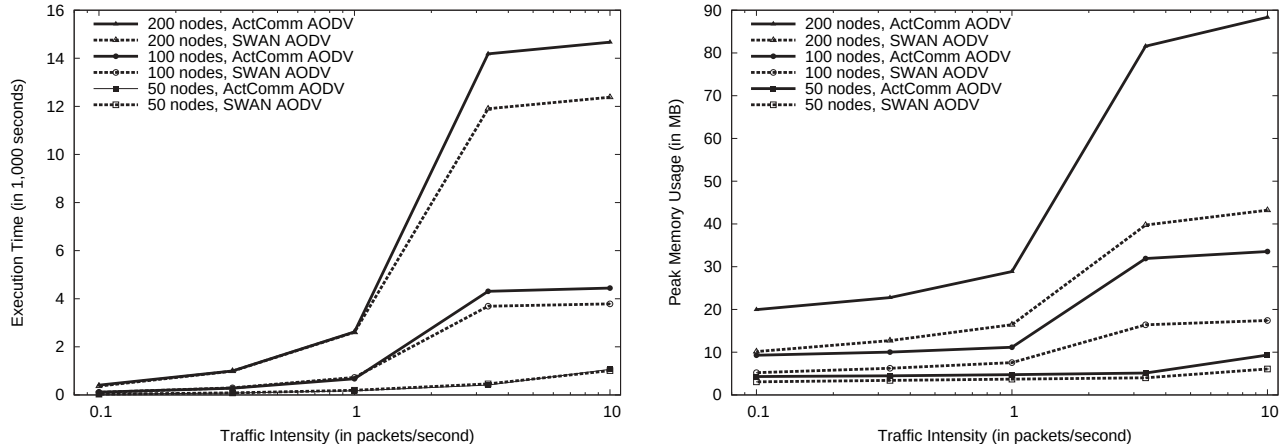

Figure 4. The execution time and peak memory usage for the two AODV implementations with varying traffic load (in log scale)

and heavier traffic load. The increased execution time was mostly caused by the overhead of copying and serializing real packets. The memory overhead of the ActComm AODV (over 100%) was 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 implementation and the application sent and received real packets with real message headers and real payloads. The overhead grew with increasing traffic intensity as packets stayed 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 is more suitable to 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 to disqualify the technique as a means of experimentation. 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 study, we compared the results from two outdoor routing experiments with our simulation results. In particular, we compared the results from the real experiments with the simulation results using different RF models. As mentioned earlier, the simulation results using the simple stochastic wireless models cannot match exactly with the results from the real experiments. By validation, we mean we want to find out whether the simulation can provide us enough confidence in the claims we make to the performance of the routing protocols. In other words, the purpose of this study is to reveal the sensitivity of the performance of the routing protocols to the underlying wireless models. It is not our focus here to compare these protocols. We analyzed the behavior and compared the performance of the protocols in another paper [12].

5.2.1 The Real Experiments

We conducted two outdoor routing experiments in 2 years on the same rectangular athletic field measuring approximately 225 by 365 meters. The field was divided into four equal-sized quadrants, one of which was approximately 2 meters 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 multihop route. We chose this field to conduct the experiments because it was at a distance away from the college campus and its wireless network. We used a different wireless channel in the experiments to further reduce the potential interference.

The first experiment was conducted in October 2003 with 40 laptop computers. The second experiment was conducted in October 2004 with 22 laptop computers. Each laptop was a Gateway Solo 9300 with a 700-MHz Pentium III CPU, 256 KB of CPU cache, 256 MB of main memory, and a 20-GB hard drive and ran Linux kernel version 2.2.19 with PCMCIA Card Manager version 3.2.4. Each laptop had a Lucent (Orinoco) 802.11 wireless card operating in peer-to-peer mode fixed at 2 Mb/sec, and the Card Manager was configured to use the `wvlan_cs`, rather than `orinoco_cs`, driver so that we could collect signal-strength statistics for each received packet. 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 a few meters during the experiment.

The first experiment included four routing protocols: APRL, AODV, ODMRP, and STARA. The second experiment included only APRL, AODV, and ODMRP. GPSR was still under development at the time of the outdoor experiments and therefore was not included in this study. The laptops, whose clocks were set to the time reported by the GPS unit, automatically ran each routing algorithm for 15 minutes (in the first experiment) or 20 minutes (in the second experiment), with 2 minutes of network quiescence between each algorithm to handle cleanup and setup chores. After each routing algorithm had been running for 1 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 5.5 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 traffic volume observed during an earlier demonstration of a military application [15]—approximately 423 outgoing data bytes (including UDP, IP, and wireless Ethernet headers) per laptop per second, a relatively modest traffic volume. The uniform random target selection simply ensured that traffic flowed to all parts of the network. 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 first experiment, the laptops were continuously moving. At the start, the 40 participants were divided into equal-sized groups of 10, each of which was instructed to randomly disburse in one of the four quadrants of the field. The participants then walked continuously, always picking a quadrant different from 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 provided continuous movement to which the routing algorithms could react, as well as similar laptop distributions across each of the four routing algorithms. In the second experiment, we used the same strategy in node movement, however, using only three quadrants of the field. Construction prevented us from using the lower quadrant (the one that was 2 meters lower in elevation than the others). The laptops moved within the remaining three L-shaped quadrants at approximately the same elevation. The difference in the number of laptops and the movement patterns between the two experiments caused variations in the performance of the protocols, as expected.

Each laptop recorded extensive logs as described in section 4. At the end of the first experiment, we discovered that 7 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 simulation results. In the second experiment, the traffic generators in two nodes were found misconfigured when we ran AODV.

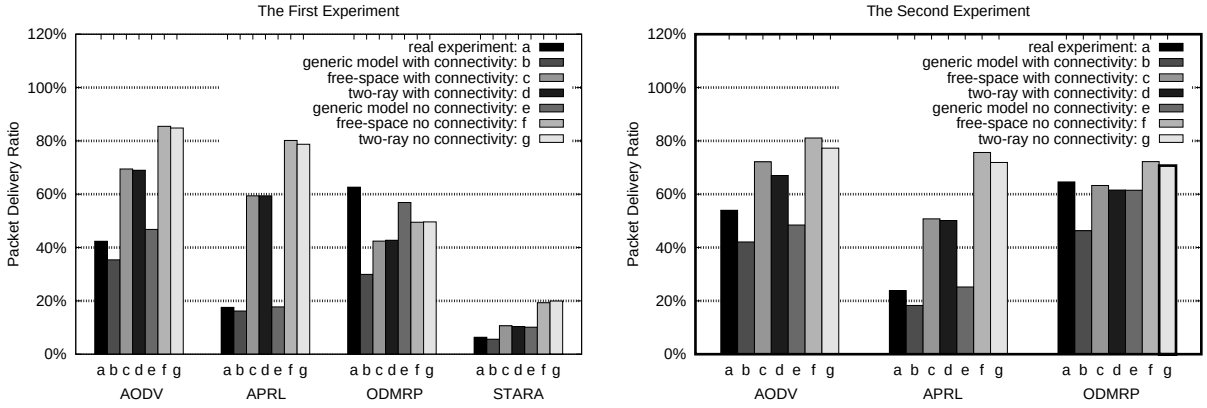

Figure 5. Comparing the data delivery ratio from the real experiments with various radio propagation models. "With connectivity" means the connectivity trace was used.

We made adjustments accordingly in the simulation to reflect the real situation.

5.2.2 The Simulation

We processed the logs from the real experiments to derive the node mobility and radio connectivity traces for each laptop for the duration of running each routing algorithm. We ran the simulation of 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. To compare with results from the first real experiment, we focused only on the 33 laptops that actually transmitted, received, and forwarded packets in the real experiment. To reproduce the traffic pattern in simulation, the application traffic generator on each of the 33 nodes still included the 7 crashed nodes as potential packet destinations.[2] For the second experiment, we included all 22 laptops in the simulation of the routing protocols but discarded the traffic generation of the 2 laptops that were misconfigured in the AODV run.

The mobile stations in simulation followed the mobility trace generated from the real 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 receiving signal power above a minimum threshold. We used the propagation models to determine the power loss for each packet transmission and thus the signal-to-noise ratio to quantify the state of interference at the receiver—to determine whether a packet that arrived at a mobile station could be received or should be dropped. We combined the three models with the connectivity trace derived from the beacon logs, leading to six different radio propagation models: three using the connectivity traces and 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 provides us a means of refining a model's power—if a model is seen to require connectivity information to work well, it is not a robust model because its power of prediction comes from measurements. On the other hand, if a model without measured connectivity information works about as well as does the version with it, then the model itself contains accurate predictive power for connectivity.

5.2.3 The Results

We first examine the packet delivery ratio. Figure 5 shows the packet delivery ratio from the real experiments 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 for the sake of clarity. The generic propagation model 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. (We later show the sensitivity of the results to these parameters.) We have several observations:

- Different wireless models predicted vastly different protocol behaviors. The difference is significant enough in

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

some cases to result in misleading conclusions, for example, when comparing the performance of AODV and ODMRP using the free-space model. The inaccuracy in the model prediction is nonuniform and can undermine a performance comparison study of different protocols.

- The simple generic propagation model with typical parameters for the outdoor test environment of the real experiment offered an acceptable prediction of the performance of the routing algorithms.
- 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 a better packet delivery ratio under the test traffic intensity. Even with the connectivity trace, the free-space model and the two-ray model overestimated the performance of the AODV, APRL, and STARA protocols. None of the two models captured the lossy characteristic of the radio propagation environment—no packets were dropped due to variations in the receiving signal strength.
- STARA's low packet delivery ratio is attributed to the high volume of control packets (measured over 150 per application packet in the experiment), which simply overwhelmed the wireless network. STARA periodically sent dummy data packets to update delay estimates for high latency routes. Optimizations in control packet handling can significantly improve STARA's performance [16]. Our implementation did not take advantage of these optimizations, and therefore the result should not be used to represent the overall performance of the algorithm. The implementation, however, can still serve as a good test case in our simulation validation study. Note that, because of the congestion at the wireless channel, the connectivity trace derived from the beacon messages does not provide an accurate estimate of the network condition.
- The performance of ODMRP was underestimated in the first experiment. ODMRP, which is a multicast routing algorithm that delivers packets using multiple paths to their destinations, has a higher demand on the network bandwidth. The overestimated transmission range in the free-space and two-ray models and the assumptions of the omni-directional radio coverage in simulation caused more contentions and created a negative effect on the throughput. The situation was improved in the second experiment with fewer laptops moving at the same elevation.
- The propagation models that used the connectivity trace generally lower the packet 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, can represent the peculiar radio propagation scenario of the test environment. Without connectivity traces, the propagation models assumed an omni-directional path loss dependent only on the distance, which resulted in a more densely connected network (with fewer hops for packet transmissions) and therefore a better delivery ratio under the given traffic intensity. The difference was more pronounced in the first experiment because the experiment included the lower quadrant—significant elevation changes in the test field led to possible obstruction of radio signals between laptops.

The packet delivery ratio does not reflect the entire execution scenario of the routing algorithm. From the routing event logs, we collected statistics related to each particular routing strategy. Figure 6 shows a histogram of the number of hops that a data packet traversed in AODV before it either reached its destination or was 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 the given number of hops. We see clearly that the free-space and two-ray models without the connectivity trace resulted in fewer hops because of the exaggerated transmission range. The problem was again more pronounced in the first experiment due to the possible obstruction of the signal propagation, resulting in longer routing paths. From the figure, we also see that the connectivity trace was helpful in predicting the route lengths, more likely in the first experiment for the same reason.

A bigger problem with the free-space and two-ray models was that they did not consider packet losses caused by variations in the receiving signal power. We illustrate this point in Figure 7, which plots the beacon reception ratio—the fraction of beacon messages received over the total number of beacon messages sent—at different distances between the transmitter and the receiver during the AODV run in the first experiment. We divided the distance range into buckets of 25 meters each and calculated the fraction of successful beacon receptions at each bucket. For better exposition, the figure shows the reception ratio at each distance bucket as a point (in the middle of the bucket). The beacon reception ratio was lower for models with the connectivity trace because of the additional precondition for the signals' reachability. The ratio generally decreased over longer distances because weaker signals were dropped when collisions happened. The generic propagation model provided the best fit for the real experiment results. In contrast, the two-ray model exhibited a sharp cliff in the curve—without variations, the quality of the modeled wireless channel changed abruptly from relatively good to none as soon as the distance threshold was crossed. Since we assumed 15 dBm as the radio transmission power and –81 dBm as the receiving threshold, the two-ray model had a maximum transmission range of 251 meters. For the free-space model, the range was 604 meters, longer than the maximum separation of laptops in the real experiment. The generic model with a path-loss exponent of 2.8 had a transmission range of only 97 meters. Because of the variations (modeled as a lognormal distribution with correlations over time), the reception ratio decreased gradually as in the real case.

The generic propagation model with typical parameters to represent the outdoor test environment offered a relatively good prediction of the performance of the rout-

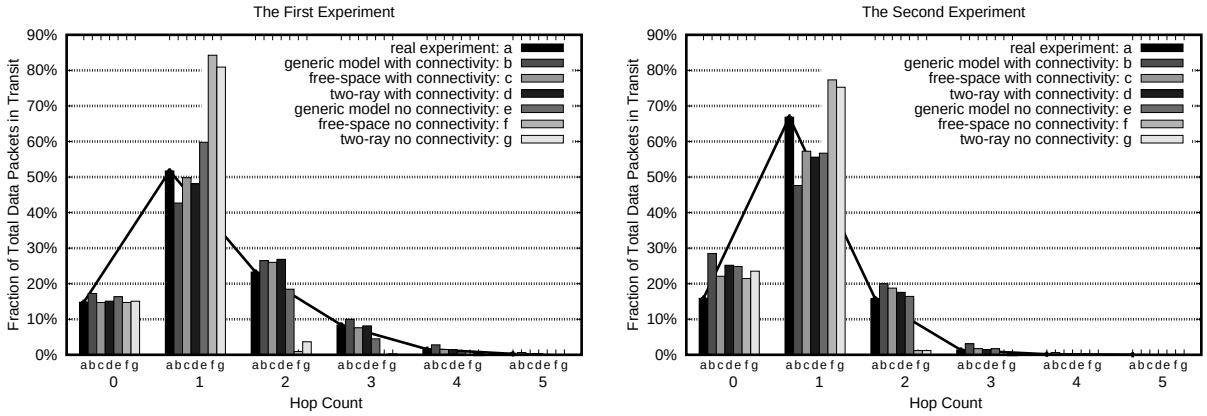

Figure 6. The hop-count histogram of AODV in real experiments and in simulation

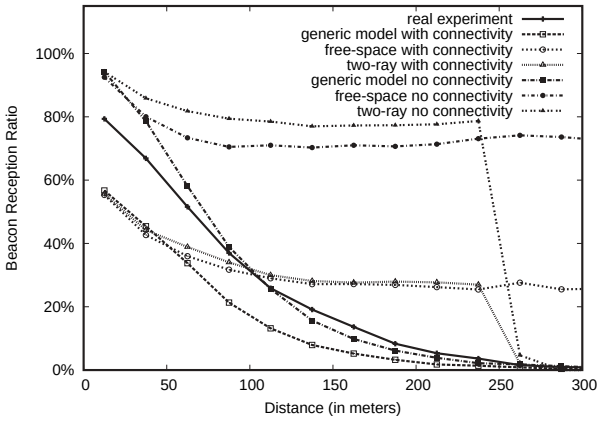

Figure 7. The beacon reception ratio at different distances between the sender and the receiver

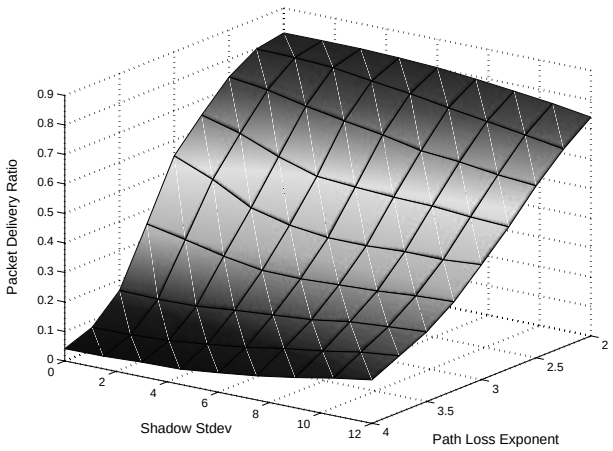

Figure 8. Sensitivity of AODV performance to parameters of large-scale fading model

ing algorithms. One must, however, choose the correct parameters carefully to reflect the wireless environment. The exponent for the distance path loss and the standard deviation in the lognormal 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 8 shows AODV performance in the packet delivery ratio with the same network settings but varying the path-loss exponent from 2 to 4 and the shadow lognormal standard deviation from 0 to 12 dB—the ranges suggested for modeling outdoor environments [14].

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 their destinations and therefore have a higher probability to be dropped. A larger shadow standard deviation caused the links to be more unstable, but the effect varied. When the path-loss exponent was small and 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. When the exponent was large, most nodes were disconnected. A variation in the receiving signal power helped establish some routes that 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 in a 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 one hand, presimulation 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. We emphasize the necessity of experimenting with a range of parameters, as different protocols may perform better under different conditions.

To summarize, we used simple stochastic radio propagation models and the traces generated from carefully designed real experiments. 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 [3], we found that using a simple stochastic RF model with parameters typical to the outdoor environment can produce acceptable results. We must recognize, however, that 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. Discussion

It is difficult to design real experiments to offer a good coverage of diverse experimental settings, such as different geographical terrains and different network conditions (traffic, mobility, etc.). Furthermore, a large-scale outdoor experiment that involves many people and a large amount of hardware equipment can easily become difficult to manage and maintain, not to mention that it is also quite costly to organize. Because of these barriers, using large-scale real experiments for validation cannot be adopted as a general approach. This is why simulation remains important for performance evaluation studies in the MANET community. One would argue, and we agree, that one should validate individual components of the simulation models—one at a time, whenever possible. Actually, the simple stochastic RF models were all tried and true within their design perimeters where the models are applicable. The question is how sensitive the simulation-based performance evaluation is to the wireless models applied without us knowing the exact design perimeters. Here, using data from the carefully designed real experiments can help answer this question. Our approach in coupling direct execution and the traces from the real experiments helps isolate and reveal the effect of the interdependent models in ad hoc network simulations.

The aforementioned drawbacks of conducting large real experiments lead to the limitations of our approach. The two outdoor experiments, described in section 5, only provide us with a few reference points in studying the effect of underlying wireless models on the performance evaluations of ad hoc routing protocols in simulation. Other experiments are needed to explore sensitivities under different settings. For example, we have yet to compare the results from our indoor tabletop experiments with the simulation results [12], in which case, the simple stochastic RF models are highly questionable for representing the indoor environments. Furthermore, it remains a challenge to generalize the results from our validation studies to large network scenarios (with hundreds or more mobile stations).

We applied statistical goodness-of-fit tests to compare the results from the real experiments with the simulation results. We choose not to show the test results simply because the null hypothesis that the results from the real and simulation experiments are from the same probability distribution is obviously incorrect. For example, the packet delivery ratio is an aggregate statistic over tens of thousands of packets transmitted during each experiment and therefore allows only small Monte Carlo errors. A confidence interval validation test would simply reject the hypothesis in this case, unless the packet delivery ratios are very close to each other. Nonetheless, a statistical test can provide a common base for comparing different models and their effect on the performance of the routing protocols. For instance, a chi-square test comparing the hop-count distributions for AODV runs in the first outdoor experiment and simulation clearly shows that both free-space and two-ray ground reflection models without the connectivity trace were way off, and the generic model provided much better match. We can simply draw the same conclusions by inspecting the graphs.

For future work, 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, as well as study the effect of such modeling details on the performance evaluation of wireless ad hoc routing protocols. Furthermore, our comparative studies assumed that the model for the 802.11 MAC layer protocol is valid. We would like to validate the model using simple real test scenarios and the validation testbed described in this article.

7. Related Work

There are varying degrees of complexity involved in the direct execution of routing protocols inside a simulator.

Previously, we ported WiroKit, a portable router for wireless ad hoc networks, developed by BBN Technologies, to execute directly in SWAN [17]. To maintain platform independence, WiroKit has only a few well-defined interaction points with the supporting operating system, which include memory allocations, communications, and accessing the real-time clock. We applied only small changes at the interaction points to allow direct-execution simulation.

Nsclick directly executes the Click Modular Router inside the *ns-2* network simulator [18]. The Click Modular Router provides a layer of abstraction and a flexible and configurable environment for developing ad hoc routing protocols. Similar to our approach, the integration of Click with *ns-2* requires visible changes to the original source code. Recently, Dimitropoulos and Riley [19] incorporated a public-domain implementation of the Border Gateway Protocol (BGP), from the routing software called Zebra, into simulation. The transition involves sophisticated changes to the original source code, including the use of event-driven scheduling to replace the original process-oriented design. Our work differs from theirs in that our approach is more specific to the ActComm framework where source code transformation is straightforward and can be done manually.

Direct-execution simulation has been used extensively in areas such as parallel architectures (e.g., [20, 21]) and distributed algorithms (e.g., [22]). They aim at obtaining an execution profile of the directly executed code. The former is concerned with measuring the execution time of running applications on the simulated computer platforms, whereas the latter is about assessing the efficiency of a parallel algorithm in a distributed environment. These are different from our case of direct-executing routing protocols in an ad hoc network simulation—measuring the execution time of the routing algorithms is of less importance than simulating packet routing across the network and therefore is often ignored in such models unless one wants to study a network in a resource-constrained situation. In this respect, Liljenstam et al. [23] recently proposed a way to efficiently model CPU and memory resources in large network simulations.

Direct-execution simulation is closely related to emulation. Emulation requires that the simulation clock be synchronized with the wall-clock time. Typically, emulation involves running unmodified software prototypes to interact with the emulated entities. For example, a network emulator provides a controlled network environment to facilitate network protocol development and application performance evaluation. Distributed applications, such as Web services, may run unmodified to supply traffic to go through the emulated network. A number of techniques can be used for executing unmodified software, including kernel virtualization [24], network packet capturing [25], dynamic linking library [26], and executable modification [27]. We are currently investigating these approaches.

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 [28] first suggested using the log information from running ad hoc routing algorithms during a real experiment to simulate identical node movement and communication scenarios. We have not seen such validation efforts in realization. There has been research in wireless network emulation based on traces [29]. In their approach, the traces are *modulated* and reduced to a simple wireless network model that preserves the end-to-end characteristics of a real wireless network. Our approach collects traces from the application layer down to the signal reception, which are used selectively to exercise and validate different components of a detailed wireless model.

Finally, Takai and others [3, 4] provide a study of the effect of a wireless physical layer and radio channel modeling on the performance evaluation of ad hoc routing algorithms. Their study considers the effect of several factors—including signal preambles, radio propagation models, and interference and noise calculations—on routing protocols in simulation. Our research is inspired by their studies but differs in our focus on using real experiments to support our simulation studies.

8. Conclusions

This article reports on our efforts to support direct-execution simulation of a set of wireless ad hoc routing protocols to facilitate validation of wireless network models. We conducted two real experiments running multiple protocols on 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. In particular, we constantly recorded the GPS location of the mobile stations, which were later translated into a mobility trace. Each mobile station also recorded the beacon messages from its neighbors, which were used to reconstruct the radio connectivity of the mobile stations. Postprocessing these files resulted in traces that we used in simulation to reproduce the same network condition. Using direct-execution simulation together with the traces from the real experiments, we are able to isolate and validate the effect of the underlying wireless models on the performance evaluation of the ad hoc routing protocols.

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. Because of the sensitivity of the behavior to the underlying radio model, one should either choose a more complex radio model for a more accurate representation of the wireless environment or use a simple model with caution. In the latter case, we suggest that one should either use models that incorporate measurements from an environment typical of the one of interest or study

the protocol behavior over a range of environments for a more complete representation.

9. Acknowledgments

This work was supported in part by Dartmouth's Center for Mobile Computing, DARPA (contract N66001-96-C-8530), the Department of Homeland Security (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 the sponsors. The U.S. government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this work, or allow others to do so, for U.S. government purposes.

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 Technologies and Bill Navidi at the Colorado School of Mines for their valuable insights and suggestions. We thank Lisa Shay, Susan McGrath, and Eileen Entin for designing the application scenario and the 60 Dartmouth students and staff members who participated in the outdoor experiments. We also thank the anonymous reviewers for their helpful comments and suggestions.

A preliminary version of this article appeared in the *Proceedings of the 18th Workshop on Parallel and Distributed Simulation* (PADS 2004) [30].

10. References

[1] Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. 3rd ed. New York: McGraw-Hill.

[2] Heidemann, J., N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. 2001. Effects of details in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, January, pp. 3-11.

[3] Takai, M., R. Bagrodia, K. Tang, and M. Gerla. 2001. Efficient wireless network simulations with detailed propagation models. *Wireless Networks* 7 (3): 297-305.

[4] Takai, M., J. Martin, and R. Bagrodia. 2001. Effects of wireless physical layer modeling in mobile ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01)*, October, pp. 87-94.

[5] Kotz, D., C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott. 2004. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, October, pp. 78-82.

[6] Bajaj, L., M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. 1999. GloMoSim: A scalable network simulation environment. Tech. Rep. 990027, Department of Computer Science, University of California at Los Angeles.

[7] Karp, B., and H. T. Kung. 1998. Dynamic neighbor discovery and loopfree, multi-hop routing for wireless, mobile networks. Unpublished manuscript, Harvard University, Cambridge, MA.

[8] Perkins, C. E., and E. M. Royer. 1999. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February, pp. 90-100.

[9] Karp, B., and H. T. Kung. 2000. Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobi-Com'00)*, August, pp. 243-54.

[10] Lee, S. J., M. Gerla, and C. C. Chiang. 2002. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications* 7 (6): 441-53.

[11] Gupta, P., and P. R. Kumar. 1997. A system and traffic dependent adaptive routing algorithm for ad hoc networks. In *Proceedings of the 36th IEEE Conference on Decision and Control*, December, pp. 2375-80.

[12] Gray, R. S., D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan. 2004. Outdoor experimental comparison of four ad hoc routing algorithms. In *Proceedings of the 7th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, October, pp. 220-9.

[13] Liu, J., and D. M. Nicol. 2002. Lookahead revisited in wireless network simulations. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation (PADS'02)*, May, pp. 79-88.

[14] Rappaport, T. S. 1996. *Wireless communications: Principles and practice*. Englewood Cliffs, NJ: Prentice Hall.

[15] Gray, R. S. 2000. Soldiers, agents and wireless networks: A report on a military application. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, April.

[16] Gupta, P. 2000. Design and performance analysis of wireless networks. Ph.D. diss., University of Illinois at Urbana-Champaign.

[17] Liu, J., L. F. Perrone, D. M. Nicol, M. Liljenstam, C. Elliott, and D. Pearson. 2001. Simulation modeling of large-scale ad-hoc sensor networks. In *Proceedings of the European Simulation Interoperability Workshop (Euro-SIW'01)*, June.

[18] Neufeld, M., A. Jain, and D. Grunwald. 2002. Nsclick: Bridging network simulation and deployment. In *Proceedings of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'02)*, September, pp. 74-81.

[19] Dimitropoulos, X. A., and G. F. Riley. 2003. Creating realistic BGP models. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS'03)*, October, pp. 64-70.

[20] Dwarkadas, S., J. R. Jump, and J. B. Sinclair. 1994. Execution-driven simulation of multiprocessors. *ACM Transactions on Modeling and Computer Simulation* 4 (4): 314-38.

[21] Reinhardt, S. K., M. D. Hill, J. R. Larus, A. R. Lebeck, J. C. Lewis, and D. A. Wood. 1993. The Wisconsin Wind Tunnel: Virtual prototyping of parallel computers. In *Proceedings of the 1993 ACM SIGMETRICS Conference*, May, pp. 48-60.

[22] Dickens, P. M., P. Heidelberger, and D. M. Nicol. 1996. Parallelized direct execution simulation of message passing parallel programs. *IEEE Transactions on Parallel and Distributed Systems* 7 (10): 1090-1105.

[23] Liljenstam, M., J. Liu, D. M. Nicol, Y. Yuan, G. Yan, and C. Grier. 2005. RINSE: The real-time interactive network simulation environment for network security exercises. *Proceedings of the 19th Workshop on Parallel and Distributed Simulation (PADS '05)*, Monterey, CA, pp. 119-28.

[24] Huang, X. W., R. Sharma, and S. Keshav. 1999. The ENTRAPID protocol development environment. In *Proceedings of the 1999 IEEE INFOCOMM Conference*, March, pp. 1107-15.

[25] Simmonds, R., and B. Unger. 2003. Towards scalable network emulation. *Computer Communications* 26 (3): 264-77.

[26] Song, H. J., X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. 2000. The MicroGrid: A scientific tool for modeling computational grids. *Scientific Programming* 8 (3): 127-41.

[27] Mukherjee, J. 2002. A compiler directed framework for parallel compositional systems. Master's thesis, Department of Computer Science, Virginia Polytechnic Institute and State University.

[28] Johnson, D. B. 1999. Validation of wireless and mobile network models and simulation. In *DARPA/NIST Network Simulation Validation Workshop*, May.

[29] Noble, B. D., M. Satyanarayanan, G. T. Nguyen, and R. H. Katz. 1997. Trace-based mobile network emulation. In *Proceedings of the 1997 ACM SIGCOMM Conference*, September, pp. 51-61.

[30] Liu, J., Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. 2004. Simulation validation using direct execution of wireless ad-hoc routing protocols. In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS'04)*, May, pp. 7-16.

Jason Liu *is an assistant professor of Computer Science at the Colorado School of Mines, Golden, Colorado.*

Yougu Yuan *is a graduate student at Dartmouth College, Hanover, New Hampshire.*

David M. Nicol *is a professor at the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois.*

Robert S. Gray *is currently working for BAE Systems.*

Calvin C. Newport *is a graduate student in the Computer Science and Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, Cambridge, Massachusetts.*

David Kotz *is a professor in the Department of Computer Science at Dartmouth College, Hanover, New Hampshire.*

Luiz Felipe Perrone *is an assistant professor in the Department of Computer Science at Bucknell University, Lewisburg, Pennsylvania.*