Validation of Wireless and Mobile Network Models and Simulation

David B. Johnson
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891 USA

`http://www.monarch.cs.cmu.edu/`
`dbj@cs.cmu.edu`

1 Introduction

Wireless and mobile networks present substantial challenges in the validation of large-scale network models and simulation, even beyond the already difficult problem of validation in more conventional wired and stationary networks. These additional challenges are due to the complications and subtleties of physical movement and wireless propagation, making the system highly variable and substantially increasing the complex interactions between the parts of the system and the surrounding environment. These same factors also make wireless and mobile experiments in the real world not easily or accurately repeatable, reducing the use of such experiments for validation.

In particular, the position and movement of nodes in the network can have a significant effect on the behavior and performance of the system being modeled. The position and possible movement of other objects in the environment around the nodes themselves, such as buildings, hills, and trees, or vehicles, people, and rain, can also significantly effect the system being modeled. Furthermore, to accurately control an entire experiment in the real world, all of these positions and movements would need to be controlled to within a fraction of a wavelength of the radios involved, due to differences in the radio multipath environment even such small position differences can cause. Having complete control over all of these factors is simply not fully achievable in any real system, and so models and real experiments, to some degree, can only be approximations.

2 Simulation Environment and Related Work

In the Monarch Project[1] at Carnegie Mellon University, our work in modeling and simulation of wireless and mobile networks has largely been focused on the area of multi-hop ad hoc networks, but has also expanded to other areas including in-building wireless LAN design issues. In ad hoc networks, we have performed extensive simulation studies, both to analyze the performance and behavior of our own Dynamic Source Routing protocol (DSR) [1, 4, 5] and to compare its performance to other proposed ad hoc network routing protocols [2].

Our simulation environment consists of a set of wireless and mobile extensions that we have created [2], based on the publicly-available *ns-2* simulator from the VINT Project [3]. These extensions provide a detailed model of the physical and link layer behavior of a wireless network and allow arbitrary movement of nodes within the network. At the physical layer, we provide realistic modeling of factors such as free space and ground reflection propagation, transmission power, antenna gain, and receiver sensitivity. At the link layer, we model the complete Distributed Coordination Function (DCF) Media Access Control (MAC) protocol of the IEEE 802.11 Wireless LAN protocol standard, along with the standard Internet Address Resolution Protocol (ARP). These wireless and mobile extensions are available from the Monarch Project Web page, and have now been adopted as a standard part of the VINT *ns-2* release available from UC Berkeley.

The radios that we currently model in our simulation environment are those used in the commercial WaveLAN product, and we have also built an outdoor ad hoc networking testbed in Pittsburgh using these radios, which we are using to evaluate our protocols and as a basis for validating our simulation. We have operated the testbed regularly over four months of testing and experimentation between December 1998 and March 1999 [6].

The testbed consisted of 5 mobile nodes implemented as cars driving at about 25 MPH over a course past two stationary nodes separated by a distance of about 700 m (about 2–3 radio hops). In each car, a laptop computer implemented the DSR ad hoc network routing protocol, served as an endpoint in different higher layer protocol connections and applications, and allowed local logging of network events on the laptop's hard disk. As the cars moved, the route between the two stationary nodes was constantly changing, and the route between any car and any other car also changed frequently as the cars moved relative to one another. The area used for the testbed was open to general vehicle traffic and has several Stop signs, so the speed of each node also varies over time, just as it would in any real, deployed network. All of the routes within the ad hoc network were dynamically found and maintained through our DSR ad hoc network routing protocol [1, 4, 5]. In operating the testbed, we experimented with a wide variety of data traffic types and network loads, including bulk file transfer, telnet, constant bit rate UDP streams simulating voice or video, and realtime position and status reporting packets.

Each car in the ad hoc network also was outfitted with a highly accurate GPS receiver operating in Real Time Kinematic (RTK) mode, providing each node with its own current position to centimeter-level accuracy. During different runs of

[1]The Monarch Project is named in reference to the migratory behavior of the monarch butterfly, and can also be considered as an acronym for "Mobile Networking Architectures."

the testbed, we were thus able to have each mobile node log its own current GPS position, as well as the source, destination, and contents of each packet sent or received. To facilitate additional position logging, the sender's current GPS position was piggybacked on each packet sent, which was logged along with the data of the packet on receipt. In addition, the signal strength and signal quality (reported by the WaveLAN hardware) for each received packet was also logged.

In related work, we have also recently created a system for *emulating* an ad hoc network on a stationary, wired network. Since the validation needs and possible approaches for network emulation are quite similar to those for network simulation, we are attempting to develop techniques that will work for both. Network emulation allows some real network (for example, a stationary, wired network) to be made to behave in the same way as some other network of interest (for example, a wireless network) by altering the one network's behavior, as seen by packets on the network, to perform like the other's. Network emulation for simple wireless networks (essentially, with all user communication through a centralized base station) has been demonstrated in the *trace modulation* work at Carnegie Mellon and UC Berkeley [7].

We have extended the trace modulation concept to create two alternate approaches to the emulation of ad hoc networks. In the "trace emulation" approach, we generate a trace of the desired network's behavior using simulation, and then use this trace to drive the standard trace modulation system in the operating system kernel of the real machines on the real network; each packet is then delayed or dropped under control of trace modulation. In the "direct emulation" approach instead, each packet from a real machine is sent to a centralized machine on which a simulation of the desired network is running, and the packet is delayed or dropped according to the behavior determined wholly inside the simulation, for each individual packet; if the packet is not dropped within the simulation, it is then resent on the real network at the appropriate time to its real destination. In either approach, the behavior of the emulated network is thus correct if the underlying simulation is correct, and if no unwanted artifacts are introduced in the emulation process.

3 Approaches to Validation

Our initial approaches to the validation of our simulation work were to check the operation of the system according to a number of logical consistency checks. For example, we analyzed the power and simulated radio behavior as a function of distance between small groups of nodes in the simulation, to ensure that the propagation, capture effect, and carrier sense models were working as designed. The IEEE 802.11 MAC model was studied in a variety of scenarios, including experimentally testing that when all nodes are within wireless transmission range of each other, no data packets experience collisions (regardless of offered traffic load), and that each node is able to make progress sending packets. This verified that the carrier sense, RTS/CTS, and back-off mechanisms of the 802.11 model appeared to be working correctly.

In addition, the simulator can log traces of various events that occur during the simulation, and these log files were manually verified for each logged event in a large number of short simulation runs or portions of longer runs. Finally, the results of each simulation were internally checked to be sure that no packets were "lost" by the simulator. That is, those packets originated by the "application layer" sources that were not logged as either received or dropped at the end of the simulation run were exactly those packets that were in transit at the end of the simulation.

Although these initial validation checks gave us substantial confidence in our results, they do not actually validate that any run from the simulator is "equivalent" to any run of the same system configuration in the real world. Given the substantial logging information provided by our ad hoc network testbed implementation, however, we believe we have enough information to simulate the identical movement and communication scenarios experienced in these real implementation experiments. We will then be able to assess the degree to which the results from the simulator match those from the real world.

However, we are just beginning to develop methods that will allow us to compare the results between a simulation run and a corresponding real experiment in a meaningful way. We believe that simply comparing the typical type of high level performance metrics that might normally be the output of such runs, such as the average user-level TCP throughput or the percentage data packets successfully delivered by the routing protocol, will not provide enough information and may indeed produce an incorrect conclusion from the comparison. If these metrics match between a simulation run and a real experiment, it does not *necessarily* mean that the models and simulation are working correctly. In addition, in some cases, even small, otherwise insignificant differences between the two systems can result in possibly quite large differences in the two performance metrics. For example, TCP's congestion control and recovery algorithms can significantly change the timing of TCP's packet transmissions (and retransmissions) and thus may have a large effect on the TCP throughput and on the overall load placed on the network under test.

One approach to comparing the results from simulations and real measurements (or emulations) that we are considering is a comparison based on the progression of some performance metric as a function of time. For example, rather than simply comparing TCP throughput at the end of the simulation run and at the end of the measurements from the real world, it is possible to compare the simulated and measured systems using graphs of TCP data bytes transferred over time. Such graphs, which are also known as time-sequence number plots, show not only the total TCP throughput, but also show the change in the "speed" at which the TCP connection operates over time. Similar graphs could be produced for other performance metrics of interest.

For example, Figure 1 shows three time-sequence number plots for the same TCP connection between the same two nodes in a 16-node ad hoc network. The top curve shows the

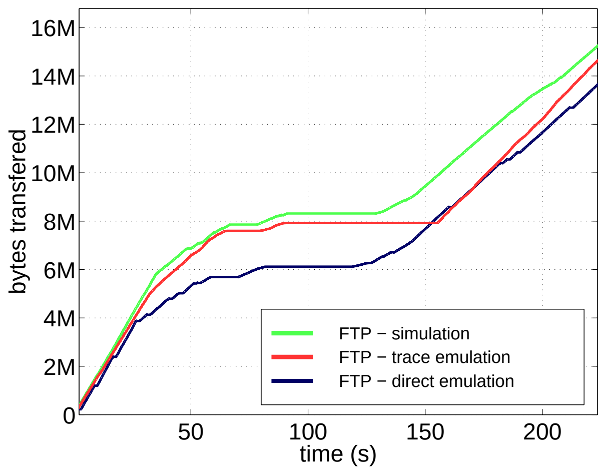

Figure 1 Time-sequence number plots for a TCP connection between two nodes in a 16 node ad-hoc network with TCP and CBR cross-traffic.

connection's behavior in a completely simulated ad hoc network using our simulator, while the other two curves show the same connection's behavior when run on an *emulated* ad hoc network, using the two different techniques for emulation that we have developed.

In order to evaluate how closely the emulations come to reproducing the simulation results, we must be able to quantitatively answer the question: "how similar are these curves?" To a human, the curves all clearly have the same shape, but the differences are interesting. Due to the experimental setup, the direct emulation curve is shifted approximately 10 seconds earlier in time than the other two curves — an effect for which the comparison function will need to correct. The shape of the direct emulation curve tracks the the simulation curve more tightly than the trace emulation curve, although the absolute RMS error is significantly greater.

Such comparisons can be made over different time scales, depending on the application of the network, and the time scale chosen may affect the comparison results. For example, when measured over intervals of 10 seconds, the slope of the simulation and the direct emulation results are substantially more similar than are the simulation and trace emulation results. For some protocols and applications (e.g., a background file transfer), a fairly coarse time scale may be reasonable, but for other protocols and applications (e.g., new wireless MAC protocols), a much finer granularity may be necessary. We imagine that the time scale, and the performance metrics of interest, could thus be parameters chosen by the modeler.

An alternative approach to validation could be to record a trace of all significant events (e.g., all packets sent, received, or forwarded) during the experiment in the real network, and to create a similar trace during a corresponding simulation run. The two trace files could then be compared, somewhat in the spirit of the standard Unix "diff" program. That is, matching sections of the two trace files could be identified and lined up, and the differing sections could then be identified. Each such match or difference could contribute to an overall "score" indicating the degree to which the two traces were the same. However, such a comparison seems to be extremely difficult to perform, since each trace represents the intertwined events caused by many different nodes and traffic streams in the network. Each of these events possibly influences other events in the trace and may also be independent of yet other events. Many events in the trace may even be able to be ignored, but it seems hard to identify which without risking too much abstraction and introducing inaccuracy. We are currently considering possible methods for turning this general trace comparison approach into a more concrete algorithm.

4 Conclusions

The validation approaches suggested in this paper can be used to compare different runs using simulation, emulation, or measurements from the real world. They seem to be appropriate at least for small- or medium-scale networks, and should be able to be applied to large-scale networks given suitable choices by the modeler. Each validation achieved for a given simulation environment or model also adds to the basis for confidence in other results from the same tools.

Acknowledgements

The ideas presented in this paper are a product of the Monarch Project at Carnegie Mellon, whose members currently include Josh Broch, Yih-Chun Hu, Jorjeta Jetcheva, David B. Johnson, Qifa Ke, and David A. Maltz. This work was supported in part by the National Science Foundation (NSF) under CAREER Award NCR-9502725, and by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, AFMC, DARPA, Carnegie Mellon University, or the U.S. Government.

References

[1] Josh Broch, David B. Johnson, and David A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-01.txt, December 1998. Work in progress.

[2] Josh Broch, David A. Maltz, David B. Johnson, Yih-chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, Dallax, TX, October 1998.

[3] Kevin Fall and Kannan Varadhan, editors. *ns* Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 1999. Available from http://www-mash.cs.berkeley.edu/ns/.

[4] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[5] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[6] David A. Maltz, Josh Broch, and David B. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1999.

[7] Brian Noble, M. Satyanarayanan, Giao Nguyen, and Randy Katz. Trace-Based Mobile Network Emulation. In *Proceedings of SIGCOMM '97*, pages 51–61, September 1997.