

Measuring and Explaining Differences in Wireless Simulation Models *

*Dheeraj Reddy George F. Riley
Bryan Larish Yang Chen*

Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250
{riley, dheeraj, blarish, ychen}@ece.gatech.edu

November 17, 2008

Abstract

Network Simulation tools have played very significant role in wireless network research in the past decade. Compared to wired networks, there are a lot more parameters that effect the behavior of wireless networks. For reasons of simplicity and computational complexity, some details are abstracted out when implementing protocol models in network simulators. IEEE 802.11 has been the dominant protocol for the MAC and PHY layers of most wireless networks in practice. Subtle implementation variations in the 802.11 protocol result in largely divergent results when higher level protocols are simulated on top of the 802.11 MAC. In this paper we concentrate on these implementation details and show how these affect the behavior of the 802.11 MAC model in various simulators. We also identify the abstractions that result in the diverging behavior of the 802.11 MAC protocol model and try to compare those with reference to the IEEE specification.

1 Introduction

Wireless network simulations are widely used and are an accepted method for studying the behavior of various classes of wireless networks, including Ad-Hoc and Infrastructure based networks. The IEEE

802.11 specification defines a Medium Access Control (MAC) layer that is one of the most popular one in theory as well as in practice. It has become the de-facto standard for most wireless implementations. Dependable wireless network simulations require that accurate models of these wireless PHY and medium-access-control protocol models be developed. On the other hand, comparing and contrasting the behavior of various higher layer protocols is an essential part of wireless networking research. Since, medium-access-control is the lowest layer of the protocol stack, it is essential to understand how various implementations behave during ideal conditions as well as under heavy traffic. The common reference for all implementations is the IEEE 802.11 specification [6]. Identifying the reasons for differing behaviors also gives us an insight into the behavior of the actual implementations of the protocols.

There are several wireless network simulators used in the research community [8, 10, 12, 1, 7]. However, not all higher layer protocols (routing, transport and application etc.) are implemented on all simulators. Thus, comparison is often made between higher layer protocols implemented on various network simulators. Despite the fact that most network simulators implement the same 802.11 MAC specification [6], unless all the implementations behave in a fairly consistent manner, these comparisons will be inconclusive at best. For our study we have restricted our research space to three simulators, namely, ns-2, GTNetS, and GloMoSim. We find that there are substantial differences in the baseline 802.11 implementation. These differences are due

*This work is supported in part by NSF under contract numbers ANI-9977544, ANI-0136969, ANI-0240477, ECS-0225417, and DARPA under contract number N66002-00-1-8934.

to various factors ranging from differing interpretations of specification to subtle details in the implementation that have been ignored for reasons of simplicity and computational complexity. Over the past few decades wired network models have evolved to an extent that we understand what parameters at the physical and data link layer affect the simulation results. As a result suitable abstractions have been developed. In comparison, wireless models are newer and provide less guidance about what details can be ignored without affecting the accuracy of the simulation.

While ns-2 [8] and GloMoSim [12] are fairly well known, GTNetS [10] is a relatively new network simulation environment. GTNetS is a scalable network simulation environment designed to support large to very-large scale simulations. The design of the simulator closely matches the real network protocol stacks and hardware. This enables the users and developers to clearly identify issues that may cause the simulator to behave differently than a typical real network. The simulator is completely implemented in object-oriented C++, leading to easy extensions for new models or modified behavior of existing models. A detailed description of the distinguishing features of GTNetS can be found in [9].

This paper quantifies the differences in the modeling of 802.11 MAC protocol in the tested simulation scenarios and presents the reasons for diverging behavior. Our studies are based specifically on two sets of experiments designed to understand the behavior of the MAC protocol. We begin by looking at the baseline implementation of the 802.11 MAC without considering the affect of contention resolution. We then consider the contention resolution mechanisms in the 802.11 MAC protocol. We also present the reasons why different 802.11 contention resolution mechanism behave differently under competing medium contentions.

The remainder of the paper is organized as follows. Section 2 focuses on the relevant related work. Section 3 describes our experimental methodology and the simple forwarding methodology that we use. Section 4 presents the experimental results. Finally, section 5 discusses the conclusions from this work.

2 Related Work

Considerable research has been done discussing the accuracy of wireless MAC algorithms and their simulation models. The relative performances of the various algorithms has been compared and contrasted using wireless network simulations. There has been comparatively very little work that focuses on how a given algorithm or a protocol (or both) perform on two or more different simulators. It is generally assumed that since IEEE 802.11 is a specification, all wireless MAC implementations based on the specification [6] will behave similarly, ignoring the physical layer differences.

Heideman et al. [5] looked to answer the validity of their simulation models in ns-2. They strived to answer the question as to what level of simulation validation is required. S.R. Das et al. [3] tried to compare routing protocols and tried to identify the various MAC protocol characteristics that might affect them.

In the context of wireless network simulations, Heideman et al. [4] were the first to look at the effects of detail in simulation. They discuss the various trade-offs in more detailed or abstract simulation models. They evaluate the effect of detail using four case studies of wireless simulations for protocol design. They suggest largely two approaches to cope with the varying levels of detail. First, by using robust networking algorithms that are stressed in similar ways by random error as by detailed models. Second, they suggest visualization techniques that can help pinpoint incorrect details. However they fail to quantify the effects of these missing details in the network simulations.

Takai et al. [11] present a set of factors at the physical layer that are relevant to the performance evaluation of higher layer protocols. These factors include received signal strength, path loss, interference, noise computation and preamble length. They concentrate on ns-2 [8] and GloMoSim [12]. They conclude that the factors at the physical layer not only affect the absolute performance of a protocol, but because their impact on various protocols is non-uniform, it can even change the relative ranking among protocols for the same simulation scenario.

Cavin et al. [2] present the simulation results of a simple straightforward algorithm using ns-2, GloMoSim and OPNET Modeler [1]. Although they

show that significant differences exist between the simulators, they do not attempt to explain the reasons for these differences. They speculate that this may be due to the mismatch of the modeling in the physical layer or due to the different levels of detail provided to implement the wireless models.

3 Experimental Setup

We constructed two wireless network simulation experiments to study how close various 802.11 implementations were to the published specification, and to compare the model implementations in the three simulation tools studied. These experiments were designed to verify the ideal behavior of the 802.11 MAC under no media contention as well as the contention resolution behavior. For both sets of experiments, the topology is as shown in Figure 1. A total of 100 nodes are placed on a 1000 by 1000 meter grid as shown, with 100 meter spacing between nodes. The nodes are assigned IP addresses as shown with node 0 in the lower left corner, node 50 in the upper right, and node 99 adjacent to node 0. Only the low-order 8 bits of the IP address are shown, with the upper 24 bits being assigned an arbitrary, but common value for all nodes. There is no node mobility, and no wireless routing protocol at all in any of our experiments. The transmission range (or transmission power) value was set such that our maximum radio range was 250 meters.

The first experiment, “experiment one”, was designed to be as simple as possible, showing only the proper operation of the RTS-CTS-DATA-ACK exchange described by the 802.11 specification. The experiment works as follows. At simulation time 0, node 0 creates a data message of 512 bytes in length, and sends it to node 1. All other nodes do nothing until a packet is received. When node n receives the data packet addressed to it, it immediately forwards the packet to node $n + 1$. As can be seen in our topology, the distance from node n to node $n + 1$ is usually only 100 meters, with 200 meter distances occasionally. In all cases, the neighbor is only one hop away, and thus no multi-hop routing protocol is needed. When node 0 receives the packet from node 99, that denotes the end of a *round*, and the beginning of the next round. The starting time of each round is noted, and the experiment continues for 100 rounds.

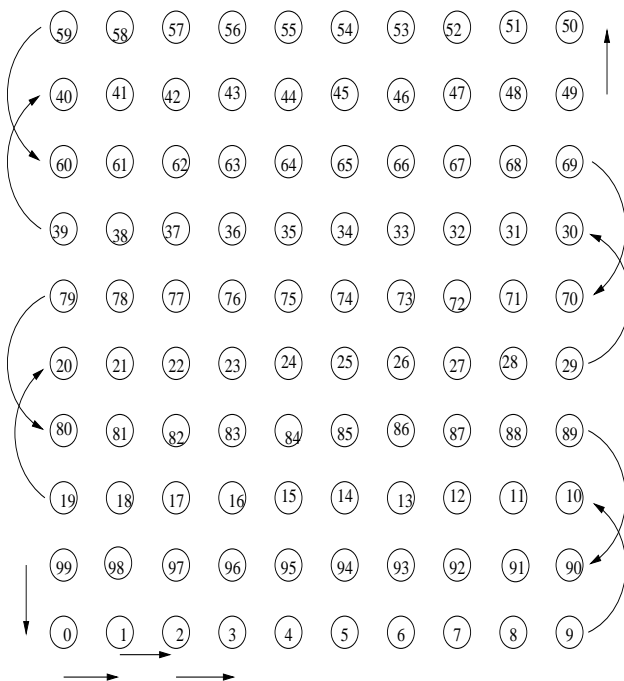


Figure 1: Simulation Network Topology

When this experiment was designed, we anticipated no randomness at all and completely reproducible and deterministic results, since there should never be any channel contention and thus no sampling of contention window random variable. In fact, this turned out to not be the case. Upon receipt of the DATA frame, all three simulators forwarded the received data packet up the protocol stack to the application layer, while simultaneously starting the transmission of the ACK frame. When the application received the packet, it immediately forwards it to the next hop and sends the packet down the protocol stack to layer two. Of course, the medium is busy at that time sending the ACK frame, resulting in a sampling of the contention window backoff variable, and some randomness in the simulation. To circumvent this randomness, we implemented a variation of experiment one that delays the forwarding of the data packet by a fixed amount of time arbitrarily chosen to be 500 microseconds. Since the 500 microsecond delay is longer than the amount of time needed to send an ACK frame, this variation did indeed result in deterministic results, as shown later.

The second experiment, “experiment two”, was designed to exercise more of the features of the 802.11 protocol specification, specifically the proper management of the *Contention Window*, *backoff*

timers, and the *Network Allocation Vector Timer (NAV Timer)*. The topology and protocol parameters are identical to those in experiment one. Again, at time 0 node 0 creates a data message and forwards it to node 1, as in experiment one. Each node n forwards the packet to node $n + 1$ as before, and rounds are measured and noted by node 0 as before. However, in experiment two, all nodes will spontaneously create a packet at a time randomly chosen in the interval [0 .. 10ms) and forward that packet to their next-hop neighbor. Thus instead of a single packet in the network at any one time, we have 100 packets all contending for channel bandwidth. Clearly there is significant randomness in this experiment, due to random starting times for competing packets and random backoff times while waiting for the channel to be come idle. This experiment was repeated 100 times for each simulation tool, and average results and 90% confidence intervals recorded. Note that in this experiment, there is significant packet loss due to MAC layer retransmission limits, and thus frequently we do not complete 100 rounds.

A summary of the parameters that we used in our experiments is given in Table 1 below.

4 Results and Discussion

In this section we presents our simulation results and discuss the reasons accounting for the differing results where applicable. We also discuss the changes that we had to incorporate in the various simulation tools as result of this analysis.

First, we performed a pencil and paper analysis of the expected results for experiment one. Table 2 presents these calculations showing the time needed for each frame during a unicast handshaking exchange. The analysis assumes a 11Mbps rate for data frames, a 2Mbps rate for control frames, and a 192 bit preamble sent at 1Mbps. As discussed in the previous section, we observed some randomness in the experiment one simulations due to the immediate forwarding of a data packet prior to the transmission of the corresponding ACK frame. We circumvented this randomness by including an artificial *forwarding delay* which allowed time for the ACK frame to be transmitted before forwarding a new data packet. Table 3 shows the analysis without any forwarding delay. Without a forwarding delay, the backoff random

Table 1: IEEE 802.11 parameters used in our simulations

Parameter	Value
basic Rate	2 Mbps (RTS/CTS/ACK rate)
data Rate	11 Mbps (Data rate)
preamble Rate	1 Mbps (Preamble rate)
RTS Size	20 bytes
CTS Size	14 bytes
Ack Size	14 bytes
DIFS	50 microseconds
SIFS	10 microseconds
Slot Time	20 microseconds
UDP Header	8 bytes
IP Header	20 bytes
LLC/SNAP Header	8 bytes
Preamble	24 bytes
Data Header	34 bytes
Payload	512 bytes
Forward Delay	500 microseconds (removes contention)
Initial CW	31 Slot times
Node Spacing	100 meters
Speed of Light	300 meters/microsecond
Hops per round	100
Number of Rounds	10

Table 2: Theoretical duration calculations (without forwarding delay)

	Duration	Cumulative Time
RTS Rx Time	322 us	322 us
CTS Rx Time	258 us	581 us
Data Rx Time	620 us	1200 us
Backoff Delay	360 us	1560 us
Per Round (100 hops)	0.1560 secs	
Time for 100 rounds	15.60 secs	

Table 3: Theoretical duration calculations (with forwarding delay)

	Duration	Cumulative Time
RTS Rx Time	322 us	322 us
CTS Rx Time	258 us	581 us
Data Rx Time	620 us	1201 us
Forward Delay	500 us	1701 us
Per Round(100 hops)	0.1701 secs	
Time for 100 rounds	17.01 secs	

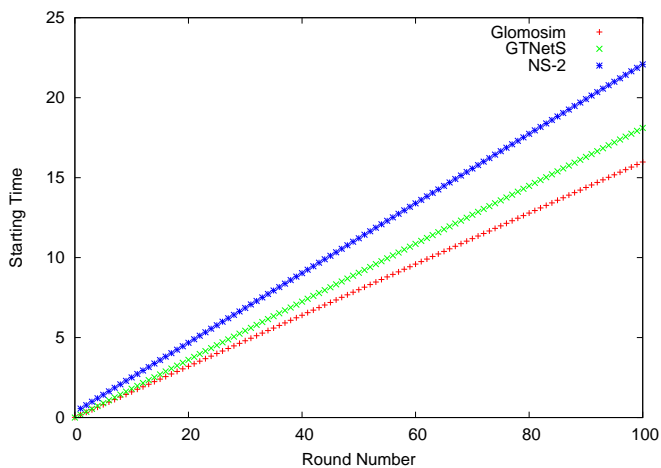


Figure 2: Starting Time vs. Round Count for Experiment 1 using default simulation parameters

variable is sampled in the range [0 .. 31], in units of a 20 microsecond slot time. Thus the expected value of this backoff is 300 microseconds. The calculations show the expected time for one round should be 0.1560 seconds without the forwarding delay, and 0.1701 seconds with the forwarding delay.

For the initial attempt at experiment one, we simply set the payload size to 512 bytes, set the 802.11 data rate to 11Mbps, and set the transmission range to 250 meters for all simulators. We used default values for all other parameters, and ran the experiment on all three simulation tools. The results are shown in Figure 2. The x-axis on the graph is the round number, and the y-axis shows the measured starting time of each round. The results presented are for the variation of experiment one using the forwarding delay of 500 microseconds.

It is easy to see that the results are quite different

for the three simulators, with only *GTNetS* matching the value calculated by our theoretical analysis. Further investigation revealed the following differences in the protocol implementations between the various tools.

1. *GloMoSim* always sends the control frames (RTS, CTS, and ACK) at the same rate as the data frames. Both *GTNetS* and *ns2* allow for differing rates for control frames, and both default to sending control frames at 2Mbps. The protocol specification indicates that control frames should be sent at the slowest rate, to allow for all nodes overhearing these frames and reacting accordingly.
2. *ns2* used the *Address Resolution Protocol (ARP)* during the first round, to discover the *MAC* address of the next hop neighbor. For subsequent rounds, the cached value of the neighbor's *MAC* address is used. Neither *GTNetS* nor *GloMoSim* used *ARP*, but rather determined the neighbor's *MAC* address by using global knowledge. *GTNetS* has the option to use *ARP* for *MAC* address discovery, but this option is not enabled by default.
3. Neither *GloMoSim* nor *ns2* account for the *Logical Link Control – Service Next Access Point (LLC SNAP)* header required for protocol demultiplexing at layer 2, while *GTNetS* includes this header.
4. *ns2* adds an additional random delay after the expiration of the *DIFS* timer and *SIFS* timers, presumably to account for noise in the clocks used in the hardware implementations.

Once these discrepancies were determined, we took corrective action as follows:

1. Set the rate for control frames (the so-called *Basic Rate*) to 11Mbps in both *GTNetS* and *ns2*, to match the basic rate used by *GloMoSim*. Ideally we would have modified *GloMoSim* to use the slower basic rate, but were unable to determine how to do this easily.
2. Ignore the results measured by *ns2* in the first round. For the *ns2* simulations, we ran 101 rounds and removed the first round data points. This removed the effects of the *ARP* packets.

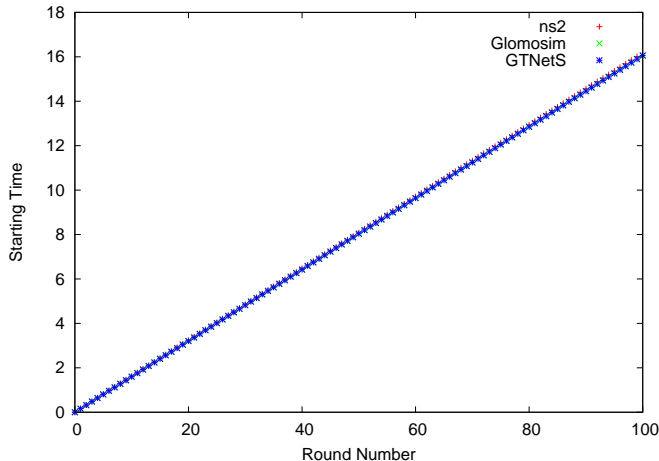


Figure 3: Starting Time vs Round Count for Experiment 1

3. Adjust the payload size for both *GloMoSim* and *ns2* slightly to account for the missing *LLC/SNAP* header.
4. The extra random delays were removed from *ns2*.

Once we made the adjustments listed above, all three simulators produced identical results, as can be seen in Figure 3. Analytical predictions similar to Table 3 with *basicRate* changed to 11 Mbps proves that the results are indeed correct.

The results of experiment 2 are plotted in Figure 4. Since we expected considerable randomness and variation in the results, we executed each simulation 100 times, and show the average result and the 90% confidence intervals. For these experiments, all model parameters were the same as in the second experiment one, using 11Mbps for the basic rate. One point to note is that, as each simulation progresses, some packets are dropped due to limits on the *MAC* layer retransmissions. Our application uses the unreliable *UDP* protocol, and therefore there are no retransmission attempts at the transport layer. Thus, there is less congestion in the network as more and more packet are dropped, and the rounds gradually take less time. Further, there is some chance that the data packet originated by node 0 is lost within a round. When this happens, there are no further rounds started (since the prior round never completes), and no additional data points plotted for that experiment.

It is easy to see that the results vary widely between the simulators. As can be seen from figure 4. *GTNetS* seems to be the most aggressive of the three simulators in that it finishes its 100 rounds the fastest. On the other hand, *ns-2* takes the longest to finish its 100 rounds. *GloMoSim* is a little slower than *GTNetS* but has a significant overlap. We spent considerable effort with testing and code inspection of the various protocol models, and discovered a number of differences discussed below. In some cases, we were able to correct the differences, and in some cases were not. In cases where the difference was corrected, the corrected version of the simulator was used for the experiments shown in figure 4.

1. When initiating a backoff, the *GTNetS* implementation doubled the value of the contention window and then sampled the random variable. The other two simulators sampled the random variable and then doubled the contention window. The specification indicated that both actions should be done, but does not state in which order. We changed *GTNetS* to sample and double as is done by the other tools.
2. All three simulators initiate a backoff after sending an *ACK* frame, if another packet is immediately available to be sent. This indicates that a data request from a higher layer was made while the medium was busy sending some other packet. However, the *GloMoSim* simulator samples the backoff timer without doubling the contention window in this case. We changed *GloMoSim* to double the contention window value in this case.
3. When a backoff timer is active and a new *NAV* timer is scheduled due to an *RTS* or *CTS* frame being overheard, the backoff timer should be suspended while the *NAV* timer is active. However, during channel idle periods the backoff timer should count down even when the *NAV* timer is counting. This will occur during interframe spacing periods, for example between the *CTS* and *DATA* frames. This is clear in IEEE 802.11 specification, section 9.2.5.2, as follows:

“A station performing the backoff procedure shall use the carrier-sense mechanism to determine whether

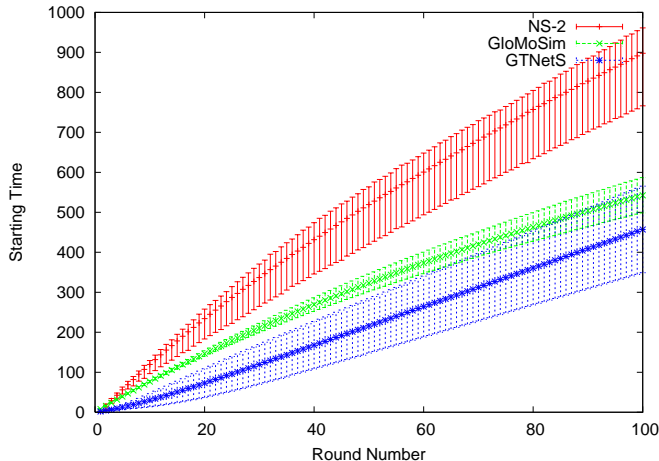


Figure 4: Starting Time vs Round Count for Experiment 2

there is activity during each backoff slot. If no medium activity is indicated for the duration of a particular backoff slot, then the backoff procedure shall decrement its backoff time by a SlotTime.”

Neither *GloMoSim* nor *ns2* allow the backoff timer to advance in this situation, resulting in longer backoff periods than the *GTNetS* implementation. We did not correct this difference, due to the lack of clear physical channel state transition event interfaces in both *GloMoSim* and *ns2*. In general, the notion of the medium being busy is a combination of virtual carrier sense as well as the physical carrier sense. If either of them indicate that the medium is busy, then the backing off mechanism is paused. It is resumed only when both the carrier sense mechanisms indicate idleness. In any scenario, where, the logical expression $(VCSBusy() \parallel PCSBusy())$ is true because of the latter half of the expression the actions taken during busy periods are likely to be incorrect.

To understand the backoff behavior better, we divide the interval between the minimum contention window (CW_{min}) and the maximum contention window (CW_{max}) into 6 bins. We then instrument the simulator to record how many times the backoff mechanism sampled in each bin. This indicates

Table 4: Distribution of backoffs in various bins

Bin	GTNetS	GloMoSim
0-31	10287	21016
31-63	2719	3328
63-127	1840	1023
127-255	1325	510
255-511	867	274
511-1023	746	253

Table 5: Average number of rounds completed per run

Simulator	Average Completed Runs
GTNetS	78.00
GloMoSim	62.18
NS-2	56.97

to us how successful the backoffs were, at resolving the medium contention. The backoff bin values for a typical run of 1000 seconds of experiment 2 are tabulated in Table 4. It can be seen that most of the contentions are resolved in the first backoff period itself. If the backoff mechanism extends to the last bin, then the chances of the contention being resolved is very low. Another interesting observation is that *GloMoSim* backoffs a lot more (almost double) times than *GTNetS*. This is because *GTNetS* samples the contention window only when it is actually backing off whereas, *GloMoSim* unconditionally samples it the first time as it receives a packet from the network queue.

As is expected, not all packets finish the 100 rounds in our experiment. To see how the backoff mechanisms help in this regard, we tabulate the average number of rounds that are completed per simulation run. The results are shown in Table 5. It can be seen that in *GTNetS*, more simulation runs complete their 100 rounds than *NS-2* and *GloMoSim*.

To summarize the experimental results, we were able to get nearly exact agreement between all three simulation tools in the simple experiment with no channel contention and no backoffs. In the more complicated scenario, there is still considerable variation in the measured performance of the network. Further analysis needed to determine if the differ-

ences we uncovered but were unable to easily correct can account for these discrepancies, or if there are other as yet undetermined variations in the implementations.

Differing implementations actually have substantial effect on the performance studies of various protocols above the MAC layer. Let us consider the example of a routing protocol which requires some form of controlled flooding to propagate route requests in a network. This is not an unreasonable assumption since, many popular routing protocols like DSR require this. From the results of experiment 2 it is easy to see that in a heavily congested environment GTNetS and GloMoSim provide a greater performance for the same protocol specification being implemented. While doing the performance study of such a routing protocol in a heavily congested network, it is easy to see that the same protocol will provide with considerably more encouraging results when implemented in GTNetS or GloMoSim than if it were to be implemented in ns-2.

5 Conclusions

We performed a comprehensive set of simulation experiments comparing the implementation of the wireless IEEE 802.11 protocol in three different simulation tools. Our experiments were designed to be as simple as possible, eliminating any variation due to node mobility, multi-hop routing protocols, or physical layer path loss computations. We showed that, after some small changes in the protocol implementations, the three simulation tools report nearly identical results in the simplest case with no channel contention. However, in the more complicated experiment with channel contention, there is still considerable variation in the predicted performance of the network. Since any realistic simulation-based study is likely to have channel contention, we believe that these simulation studies might produce differing results depending on which simulation tool is used. We are not claiming that any one tool is right and the others wrong, but rather that they are different.

We are still actively studying the implementations in the three simulation tools, and will continue to identify and correct where appropriate any differences found. However, we also expect that in a field experiment using wireless devices and 802.11, there

may be considerable variation in measured results depending on the hardware and firmware used for the experiments. Different interpretations of the specification can just as easily be present in hardware implementations as in simulation software. The important issue is not whether different experiments give different results, but in understanding the source and overall effects of the differences.

References

- [1] S. Bertolotti and L. Dunand. Opnet 2.4: an environment for communication network modeling and simulation. In *Proceedings of the European Simulation Symposium*, October 1993.
- [2] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43, New York, NY, USA, 2002. ACM Press.
- [3] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM'00)*, pages 3–12, 2000.
- [4] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. chan Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 3–11, Phoenix, Arizona, USA, January 2001. USC/Information Sciences Institute, Society for Computer Simulation.
- [5] J. Heidemann, K. Mills, and S. Kumar. Expanding confidence in network simulation. *IEEE Network Magazine*, 15(5):58–63, Sept./Oct. 2001.
- [6] IEEE. IEEE Standard 802-11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. *Institute of Electrical and Electronic Engineers*, 1999.

- [7] D. N. Jim Cowie, Andy Ogielski. The scalable simulation framework. Software on-line: <http://www.ssfnet.org/homePage.html>, 1999.
- [8] S. McCanne and S. Floyd. The LBNL network simulator. Software on-line: <http://www.isi.edu/nsnam>, 1997. Lawrence Berkeley Laboratory.
- [9] G. Riley. Large-scale network simulation with GTNetS. In *Proceedings of the 2003 Winter Simulation Conference (WSC'03)*, pages 676–684, Dec 2003.
- [10] G. F. Riley. The Georgia Tech Network Simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 5–12. ACM Press, 2003.
- [11] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 87–94, New York, NY, USA, 2001. ACM Press.
- [12] X. Zeng, R. Bagrodia, and M. Gerla. Glo-MoSim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, May 1998.