Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

USING NETWORK SIMULATION IN CLASSROOM EDUCATION

George F. Riley

Georgia Institute of Technology
Atlanta GA. USA

ABSTRACT

The use of network simulation tools has become ubiquitous in nearly all areas of computer network design and research. However, simulations have been less prevalent in undergraduate and graduate level networking fundamentals classes. We believe that high-quality network simulation tools can be used to enhance the overall learning experience by students in these classes. We discuss the use of network simulations on our graduate-level ECE6110, ”CAD for Computer Networks” class. The class has a two-pronged focus, first in simply learning the capabilities and use of one or more network simulation tools, and secondly to use those tools to demonstrate network behavior of a number of different network topologies and under a variety of conditions, both wired networks and wireless networks. We present a brief overview of the capabilities of the ”ns-3” network simulator, followed by a detailed discussion of the simulation assignments and the learning objectives that are met by students running the simulations and analyzing the results.

1 INTRODUCTION

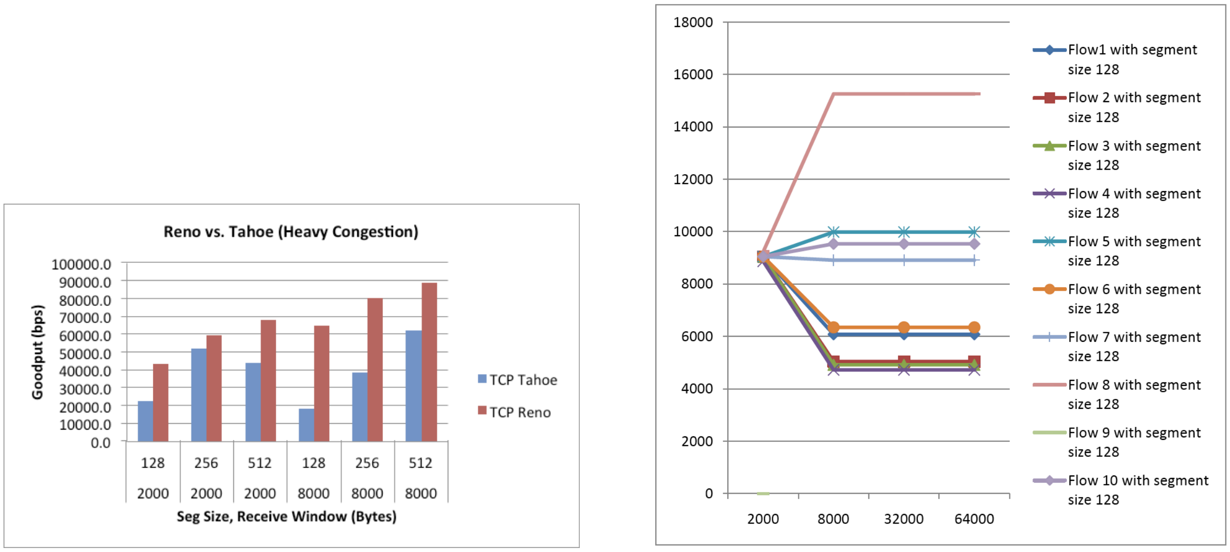
Network simulation tools have been an active research and development area for many years, with a variety of different tools available for researchers and faculty to model and analyze network behavior. We actively use the relatively new *ns-3* for both research and education. For research we use the simulator to model a variety of differing network scenarios both wired and wireless to test and evaluate new protocols or modifications to existing protocols. For education, we have found that a packet–level detail simulator such as *ns-3* and a detailed network animation package such as *NetAnim* are valuable tools to help students understand the complex behavior of packet–switched networks.

Our graduate–level class, ECE6110 (*CAD for Computer Networks*) was originally designed with the intent to teach students how to model and analyze behavior of complex networks using discrete event simulation methods. Originally, the class focused on using *ns-2* and its associated animation tool *NAM*. Later we switched to using the *Georgia Tech Network Simulator* (*GTNetS*) (Riley 2003) which has better scalability and efficiency, and is somewhat easier to use. In 2010 we switched to using the NSF–funded *ns-3* simulator, which is rapidly gaining a world–wide user and contributor base.

As we gained more experience in the class, we discovered that many students were in fact learning much more about the behavior of both wired and wireless networks than they were about modeling and simulation. Given this, we set out to design several simulation experiments to help students gain insight both about simulation methods, but more importantly illustrate the interactions between queuing methods and queue limits, segment sizes, receiver window sizes (for wired networks) and transmission power, routing protocol, and network density (for wireless networks).

We will start with a brief overview of the *ns-3* network simulation tool in Section 2. Then we describe one of the assigned simulation experiments for wired networks and some sample simulation results in Section 3. Next we describe an extensive simulation experiment measuring performance of a large wireless network and some sample results in Section 4. Students were able to choose their own final projects (from a set provided by the instructor) and we present some representative projects in Section 5. Finally, we summarize our thoughts on using network simulation in the classroom in Section 6.

978-1-4673-4782-2/12/$31.00 ©2012 IEEE



(a) Single Flow Experiments Results.

(b) Ten Flow Experiments Results.

Figure 1: Wired Experiment Results.

2 OVERVIEW OF NS-3

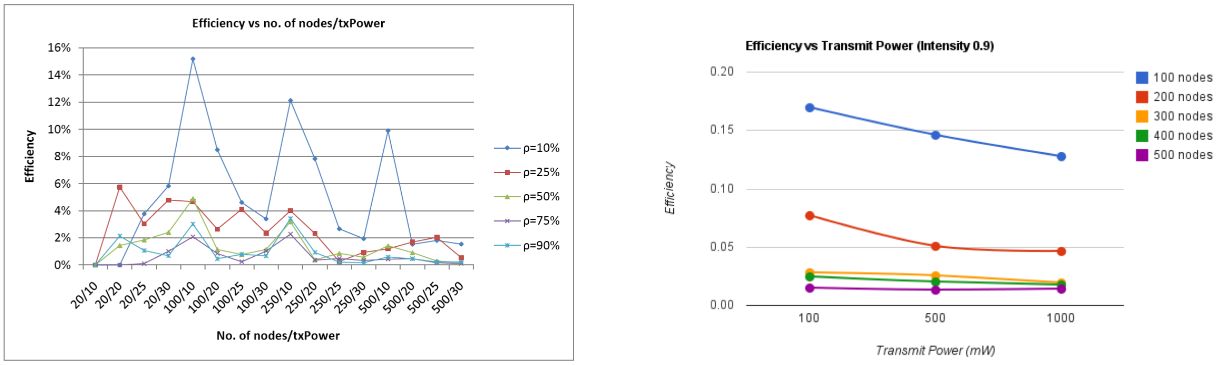
The *ns-3* network simulator is the result of a collaborative effort between researchers at Georgia Tech, University of Washington, INRIA France, and Bucknell University. This work was begun in summer of 2006 as a result of a grant from the U. S. National Science Foundation. The approach proposed and used by the team was a complete redesign of a new simulation tool, rather than any attempt to modify or adapt older simulators such as *ns-2* (McCanne and Floyd 1997). In addition, the design was such that the ability to utilize distributed simulation methods to increase scale and performance was inherent in the design. The *ns-3* simulator has seen continual development and enhancement since 2006, with more than 40 developers worldwide contributing models to the tool. Recent *ns-3* releases have been downloaded more than 5000 times per month.

The design of *ns-3* uses a very novel event scheduling mechanism that allows any member function with arbitrary parameters on any *C++* object to act as an event handler. This approach greatly eases model development, as compared to simple well-known event handler abstract classes. Further, the *ns-3* design makes considerable use of helper objects that ease the burden on a simulator user when creating complex topologies with applications. Network packets are modeled and managed in *ns-3* as simply an array of bytes, exactly as packets are represented in real networks and real routers. The rationale for this design decision was originally to ease the integration of *ns-3* with actual networks using *network emulation*.

Additionally, the *ns-3* tool was designed and implemented to support distributed simulation of larger network topologies. Some detailed discussion of the design and implementation of the *ns-3* distributed simulation can be found in (Pelkey and Riley 2011). The approach used to implement the distributed execution of *ns-3* was to utilize the well-known *MPI* message passing library for both time synchronization and packet migration between logical processes.

3 WIRED NETWORK EXPERIMENTS

The first simulation experiment is designed to illustrate the interactions between queue size, TCP receiver window size, TCP segment size, and TCP implementation variant. The students create a simple dumbbell topology with a single TCP sender and a single TCP receiver. The TCP parameters and queuing parameters are varied over a large number of ranges, and the effective throughput of the network is measured and



(a) Wireless Experiment Results.

(b) More Wireless Experiments Results.

Figure 2: Wireless Experiment Results.

reported. After this set of experiments, the students created 10 simultaneous flows sharing the same bottleneck link and observed fairness (or lack thereof) between the flows.

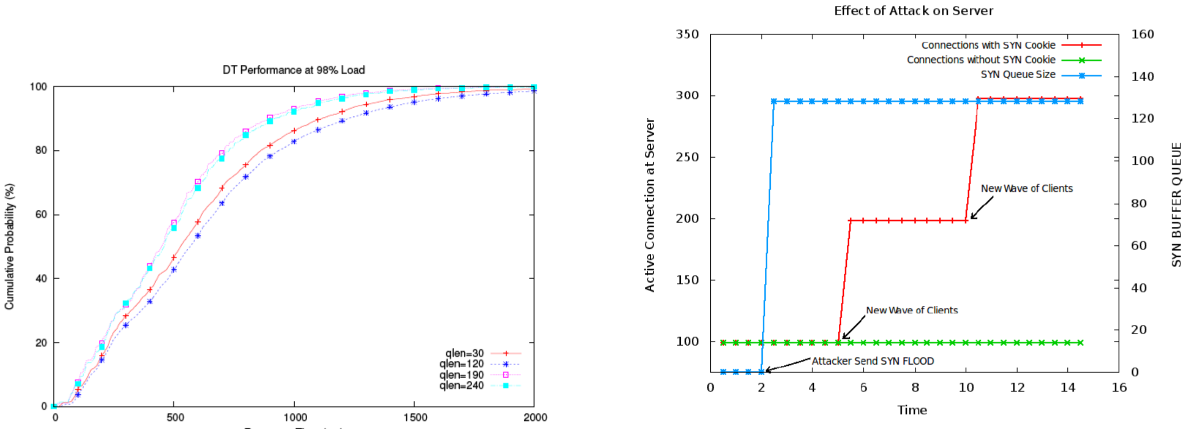
Not surprisingly, the results change considerably as the various network parameters are varied through the reasonable ranges. Figure 1(a) shows variation based on TCP variant (Tahoe and Reno) as well as on segment sizes and receiver window sizes. More telling are the results from the second part of the experiment with 10 simultaneous flows all sharing the same bottleneck link. Prior to running the experiments, most students stated that they thought the throughput for the 10 flows would be nearly the same with somewhat fair sharing of the bottleneck link bandwidth. Figure 1(b) shows considerable variation in performance between the 10 flows, even though the TCP parameters are the same for all flows. Based on these experiments, students get significant insight into the overall behavior and randomness of packet–switched networks.

4 WIRELESS NETWORK EXPERIMENTS

For the wireless experiment, the students created a square 1000 meter by 1000 meter grid, and placed a random number of wireless devices varying from about 100 to about 1000 devices per experiment, and chose random locations for the devices using a uniform distribution. They varied the transmission power for each of the devices from 1 milli-watt to 500 milli-watts, and used two different routing protocols (AODV and OLSR). Finally the *traffic intensity* was varied from 0.1 to 0.9. Here, the term *traffic intensity* means the total network demand in bits per second divided by total network capacity in bits per second. To achieve the demand, each device chose a random peer and generated a stream of packets addressed to that peer using the well-known *On–Off* traffic model. The ratio of the on and off times, as well as the sending rate while in the on state affect the overall network demand for each node. The measured result for this set of experiments was the *network efficiency*. Network efficiency is defined as the total number of packets successfully received at the intended recipient divided by the total number of packets sent at the originators. Clearly the measured network efficiency will vary between 0.0 and 1.0.

Two representative sets of results are shown. Figure 2(a) shows that the network efficiency varies considerably based on traffic intensity, number of nodes, and transmit power. Interestingly, the best efficiency results in low intensity (less overall demand) and smaller transmit power. From this experiment students observe the ill effects of collisions and interference as the traffic intensity approaches 1.0, and the ill effects of too strong transmissions which interfere with a large number of competing transmissions. The second set of results in Figure 2(b) show similar findings presented graphically in a simpler way.

This set of experiments reinforced for the students the difficulty of designing and deploying large–scale wireless networks, such as might be deployed in a military battlefield.



(a) Single Flow Experiments Results.

(b) Ten Flow Experiments Results.

Figure 3: Final Project Results.

5 FINAL PROJECT EXAMPLES

Students were given a list of sample projects that they could choose to count as a *Final Project*. The final project is intended primarily to further understanding of networking behavior by using the *ns-3* simulator. As in prior projects, the use of the simulation tool was incidental to the overall learning activities. The focus was rather on observing network dynamics. The following lists some of the possible projects.

1. Effect of *Random Early Detection* (*RED*) on web browsing response time. In this project the students recreated, with simulated networks and simulated web browsing applications, the results reported in a well–known paper by Jeffay (Christiansen, Jeffay, Ott, and Smith 2000). The Jeffay paper reported on an extensive experimental study comparing web browser response time as a function of a number of queuing parameters such as RED vs. Drop Tail, queue length, and the various RED tuning parameters. The *ns-3* simulator was able to recreate this experiment with minimal additional models to mimic the Jeffay web browsers and servers. The reported results did not agree precisely with those reported by Jeffay, but the overall behavior was the same. Figure 3(a) shows a representative cumulative distribution function of the browser response in one particular case.
2. Modeling *Worm* propagation. In this project, the students created moderately large network topologies of several thousand network nodes, and started with an initial *infected* node. Infected nodes then send additional attack packets to randomly chosen IP addresses, and the overall rate of spread of the worm was measured and reported. Again, *ns-3* had sufficient functionality excepting the addition of the *Worm* application. The measured rate of spread of the worms essentially agreed with previously reported results (Sharif, Riley, and Lee 2004) under similar conditions.
3. Large Scale Denial–of–Service attack. Here the simulator was used to model a single web server with a large number of browsers connecting to and downloading from that server. Next, several thousand *Syn Flood* attacks were initiated which caused the server to eventually become overloaded and run out of available ports. The students also modeled one of several possible defenses to the *Syn Flood* attack and reported on the effectiveness of the defenses. Figure 3(b) shows a representative result from the *Syn Flood* modeling activity.

As can be seen, the focus of all of the projects, and in particular the final projects, was not necessarily about simulation. Rather, the message was to use the simulation tool to gain understanding of network behavior. In some cases, detailed knowledge of the simulator design and implementation was needed if additional applications or networking methods were needed. But in all cases, the focus of the project was understanding networks, not just understanding simulations.

6 SUMMARY

We have nearly 10 years of experience in teaching ECE6110, and it has changed quite a bit over the years as newer and more capable simulation tools have become available. In all cases, the classroom lectures on the simulator itself are just a small part of the overall classroom time. The remainder of the class is devoted primarily to exploring how networks work, reading research papers by others that report on network behavior, and observing network behavior using network animation features of the various simulators.

All of the assignments discussed here can be found on the ECE6110 class web page:
http://users.ece.gatech.edu/~riley/ece6110.

REFERENCES

Christiansen, M., K. Jeffay, D. Ott, and F. D. Smith. 2000, August. “Tuning RED for Web Traffic”. In *Proceedings of ACM SIGCOMM 2000*, 139–150.

S. McCanne and S. Floyd 1997. “The LBNL Network Simulator”. Software on-line: http://www.isi.edu/nsnam. Lawrence Berkeley Laboratory.

Pelkey, J., and G. Riley. 2011, March. “Distributed Simulation with MPI in ns-3”. In *Proceedings Workshop on ns-3*.

Riley, G. F. 2003. “The Georgia Tech Network Simulator”. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 5–12: ACM Press.

Sharif, M. I., G. Riley, and W. Lee. 2004. “Simulating Internet Worms”. In *Proceedings of the Twelvth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS’04)*.

AUTHOR BIOGRAPHIES

GEORGE RILEY is a Professor of Computer Engineering at Georgia Institute of Technology. Dr. Riley received his Ph.D. from the Georgia Tech College of Computing in August 2001, and joined the faculty of Electrical and Computer Engineering at that time. Dr. Riley received a MSCS from Florida Tech in 1996, and a BSEE from University of Alabama in 1972. His research interests include distributed discrete event simulation techniques, large-scale computer networks, operating systems, and multi-core architectures and interconnects. His e-mail address is riley@ece.gatech.edu.