

# Student Projects in Computer Networking: Simulation versus Coding

Leann M. CHRISTIANSON  
California State University, East Bay  
Department of Mathematics and Computer Science  
Hayward, CA 94542-3092, USA

and

Kevin A. BROWN  
California State University, East Bay  
Department of Mathematics and Computer Science  
Hayward, CA 94542-3092, USA

## ABSTRACT

*In this paper, we discuss the advantages and disadvantages of using network simulators to teach networking concepts versus having students write programs. The authors have experience developing laboratory exercises for classroom use in both realms. Two case studies, TCP versus UDP protocol performance, and routing algorithm convergence are described. For both studies, students either wrote a program that implemented the protocols being tested or created a simulation using the OPNET Modeler package. Issues specific to these projects are discussed. Student opinion reflected that simulation software allows more focus on the performance of the protocol and visualization of results. Programming, though viewed by students as more difficult, allows them to practice and gain skills that will be useful in their future careers. The authors feel that a balance of both programming and simulation activities is the best practice.*

**Keywords:** Simulation, network, teaching, OPNET

## 1. Introduction

When teaching computer networking it is important to design activities for students that illustrate the concepts covered in lecture. In this

paper, we consider the pros and cons of using network simulators to teach computer networking concepts versus having students write programs. The authors have experience developing laboratory exercises for classroom use in both realms [1-3]. Issues that impact the development of effective networking assignments are presented along with case studies and student feedback.

Computer networking is an integral part of computer science education today. Creating practical and compelling student assignments in networking, however, is challenging. This is due to the fact that networking concepts and algorithms often involve many entities or are designed to work in environments that are not readily available for classroom use. Some of the issues to consider when designing computer-networking exercises are listed below:

- a. Many protocols differ in how they respond to **corruption, errors, and loss**, which occur uncommonly and inconsistently in real networks making replication of results difficult.
- b. The performance of data-link layer algorithms may only be apparent under **high load** which may negatively impact other users of the network.
- c. The differences in the performance of transport algorithms may only be apparent if the transmitter and receiver nodes are separated by **large distances** or if a large number of machines are linked to the network.

- d. Routing algorithms require communication with tens to hundreds of other nodes.
- e. *Security concerns* within a university may make it difficult to provide access to live servers with which to interact (e.g., to test a student web client).
- f. *Super-user* access may be needed to use commands that report on network performance
- g. Much network code is at the *kernel-level* requiring recompilation of the kernel in order to modify.

Computer networking algorithms are complex, therefore, it is beneficial to provide more than a description and explanation of them in a lecture setting. Students often learn more by experimenting on their own, by poking and prodding an algorithm to see how it reacts to different stimuli. The ability to experiment may be provided in (at least) two ways. Students can be supplied with a simulation environment, or they can be asked to implement an algorithm themselves and run it on real machines.

## 2. Simulation Versus Programming

In the computer networking area, as in much of Computer Science, it is not considered sufficient to simply teach the theory of a concept. Computer Science may be considered engineering, and as such, Computer Scientists should be able to apply what they learn by building an artifact or program which embodies the concept. In much the same way that a person who builds an automobile will have a better understanding of its workings than a person who only knows how to drive it, our students will understand algorithms more deeply by implementing them. In addition, undergraduate students should be given every opportunity to improve their coding and widen the application of their skills.

There are many practical considerations in computer networking, however, which may preclude a project's successful creation and testing in a real environment. Even with a completely dedicated network laboratory in which students are given super-user access, many experiments may not be conducted due to the need for additional nodes at a distance to act as correspondents. Without a dedicated lab, students may be unable to modify network algorithms residing in the kernel. Due to security concerns, students may not be able to run network interfaces in promiscuous mode in order to monitor packet arrivals on the local LAN.

Testing algorithms presents another issue. Even within a dedicated lab, it is nearly impossible to exactly duplicate the conditions that would be required in order to evaluate an algorithm such that all students would see identical results. Traffic

generators may be used to emulate load, but inducing errors, loss and excess delay is a nontrivial enterprise. Since many algorithms require communication with many nodes over a distance, they would necessarily rely on the traffic, error, and loss patterns seen in the Internet. These patterns are by no means reproducible.

Simulation software as an alternative to programming offers many benefits. Sophisticated simulation software allows for rapid construction of both wide and local area networks, intermediary, and host nodes. Protocols can be easily interchanged and tested. Large distances and multiple nodes can be simulated and errors, losses, and delay rates can be defined to follow a chosen distribution pattern. This allows for reproducible results. Simulation software applications do not require super user access or a dedicated lab. Students can experiment without administrative fear of a compromised network. Timeliness is also a factor as long time periods (hours and weeks) can be simulated much more quickly than experimenting in real time.

There are negative aspects of using simulation software as well. Valuable class time will need to be spent learning how to use the software. Some simulation applications do not allow the user to modify all simulation entities. This means that it might be impossible to demonstrate a particular event such as a pattern of movement of mobile nodes for example. Additionally, simulation results will depend on the simulation model which may not accurately reflect real traffic patterns. Cost, support, and license management of simulation software will also need to be considered. Lastly, and most important, the students will not gain experience writing actual networking code and interacting with live standards-based servers.

## 3. Case Study: TCP Versus UDP

The effect of packet size on Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) round-trip delay is a concept presented in most beginning network courses. We have assigned this exercise as both a simulation and a programming assignment. Using simulation software such as OPNET Modeler [4], students can easily “drag and drop” networking components to a workspace to create a network model. Components can be linked together using a variety of link speeds, and nodes can be configured to use the TCP or UDP protocol via drop down menus. Figure 1 illustrates a wide area network with a File Transfer Protocol (FTP) client and FTP server. A base packet size and traffic pattern can be configured from a list of options. The end nodes can be configured with a drop-down menu to use TCP or UDP packets when simulating a file request.

The simulation can be rerun with different packet sizes, and results can be graphed for easy comparison. Students should see that TCP packet delay is longer than UDP packet delay due to the added complexity of the protocol. Additionally, students should note that the delay to process each TCP packet increases with packet size. Figure 2

illustrates a graph of TCP and UDP delay times for a file transfer.

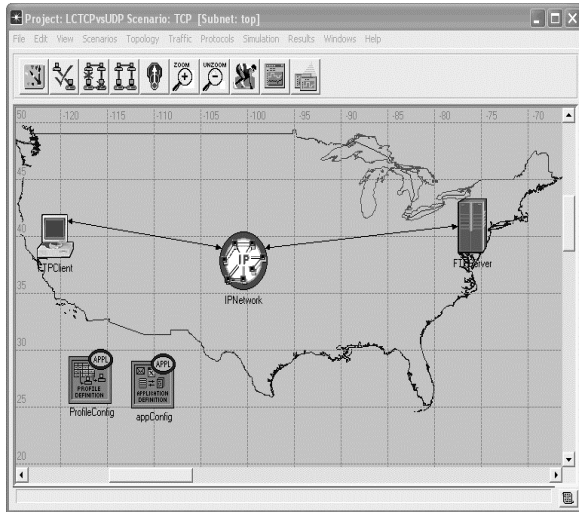


Figure 1. OPNET model of a wide area network with an FTP client and FTP server.

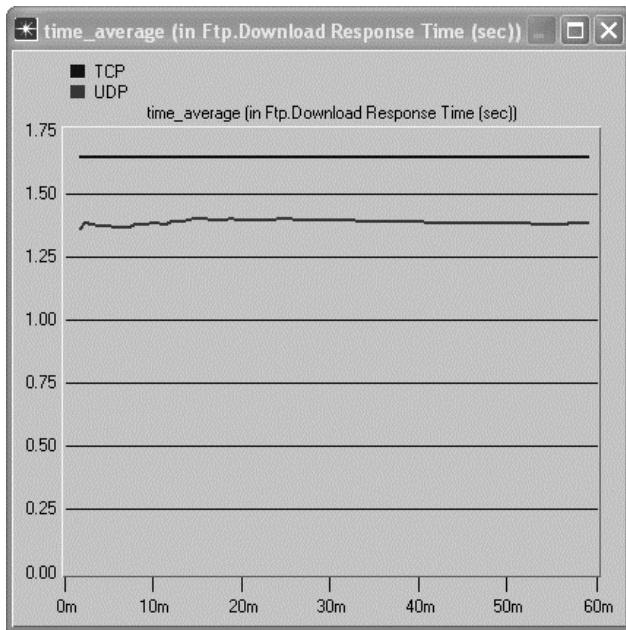


Figure 2. Graph of TCP versus UDP packet delay for a file transfer.

A benefit of this type of simulation is that once the initial simulation has been completed, students may change parameters such as link speed, packet size, and background traffic load to observe how these affect their initial results. Various types of application background traffic such as web, database, or email can be chosen. We have found that the ability to easily change factors which affect performance, the quickness in which each simulation runs, and the ability to view graphically the results all

serve to encourage the students to play with their simulation, discovering and learning on their own. A disadvantage, however, is that the students will not gain valuable experience in writing programs that use communication protocols. This is certainly an important skill in which computer science students should be accomplished

As an alternative, an instructor may instead choose to have students write their own programs for this experiment [3]. This involves writing a client program and a server program that communicate via TCP or UDP sockets, running multiple tests, exporting results to a file and graphing them via Excel. A description of the programming assignment is below:

*TCP vs UDP: UDP is a simple best-effort, demultiplexor protocol. TCP, on the other hand, is reliable and connection-oriented. TCP, therefore, will go through a connection set up phase prior to sending a packet. In this assignment, you will compare delay values for each protocol. This can be accomplished by writing 2 pairs of programs which will exchange information using TCP and UDP and sockets.*

*The first pair of programs (a sender and a receiver) should use UDP to send a packet of set size back and forth between the source and destination 10,000 times. You will want to calculate how long the 10,000 round trips take, therefore, the source must time the process. The source should check the time when it first sends the packet and again check the time when the packet comes back the 10,000th time. Next, your program should calculate the difference, and use this difference to calculate the delay for a single, one-way trip. Note that the delay consists of processing delay, queuing delay, transmission delay, and propagation delay. The last three components of this delay should be the same for both UDP and TCP (on the average). The second pair of programs should do the same thing as the first pair, but use the TCP protocol instead. Run each pair of programs 100 times and create a histogram that displays the outcomes.*

This assignment appears fairly straightforward, but there are some issues that can make the testing complex. Packet size can be adjusted easily, but it is difficult to generate a consistent level of background traffic. Measuring results in a real system is difficult as there is no way to control the actions of other users competing for network resources. Programming languages (C++ or Java) and operating systems (Unix or Windows) will affect delay times, and results among students will vary. It is our experience that students become confused when their results are inconsistent with what they have learned in class. For instance, due to unseen circumstances, TCP delay may appear to be less than that of UDP, contradicting what the experiment was meant to illustrate. Students will have trouble discerning whether the problem lies in their code or in the network conditions at the time of the test. The programs will also need to be run repeatedly, hundreds of times in order to get a good test sample. This takes much longer (20 minutes per run versus 1 minute) amount of time than an OPNET simulation. Finally, a big disadvantage is the lack of immediate graphical output. Program results will need to be output to a file and later imported to a spreadsheet for

graphing. Many students have trouble with this and forego it all together.

Student comments reflect that the experiment's goals and outcomes are clearer when simulation software is used. Students enjoy playing with the simulation and tweaking model components. The students do feel, however, that the experience of coding and using sockets for program communication is worthwhile. For many, this is the first time that they have been exposed to socket programming. Though students do enjoy flexibility, some problem issues of this exercise can be assuaged by having all the students use the same programming language, operating system and network.

#### 4. Case Study: Routing Convergence

Routing algorithm convergence time is another lesson that students can explore through simulation or programming. Convergence time is the time it takes for routes within a routing table to stabilize after start up or after a link has gone down. Common routing algorithms such as Open Shortest Path First (OSPF), the Routing Information Protocol (RIP), or others can be used. Using OPNET Modeler to simulate this exercise is straightforward. A group of routing nodes can be connected into a network of a given topology. Figure 3 illustrates a simple topology with five nodes. Each node can be configured to run the routing protocol of choice.

OPNET provides the capability to export routing tables to a file as well as to allow the student to "break a link" after a certain amount of simulation time. Students can then study the routing tables to find the new path that was propagated to the routing tables after the link went down. This exercise gives the student experience with router interfaces and interpretation of routing tables. Students can then study the simulation and routes and calculate the convergence time for the original network and then again for the network after one link goes down. Figure 4 illustrates a routing table for node 0 at 8 seconds into the simulation.

The advantage of using a simulation for this study is again, rapid configuration and design. The routing protocol, number of nodes, and interfaces can all be configured easily. Results can be viewed graphically in a quick and easy manner, and routing tables are automatically created. Students enjoy creating a simulation for this exercise. Many have stated that they understand the routing concepts discussed in lecture much better after having completed this assignment. In particular, they feel that they have a better understanding of routing table organization, and how to interpret the routes created by a particular routing protocol. They have also gained increased

understanding of routing table convergence and its implications.

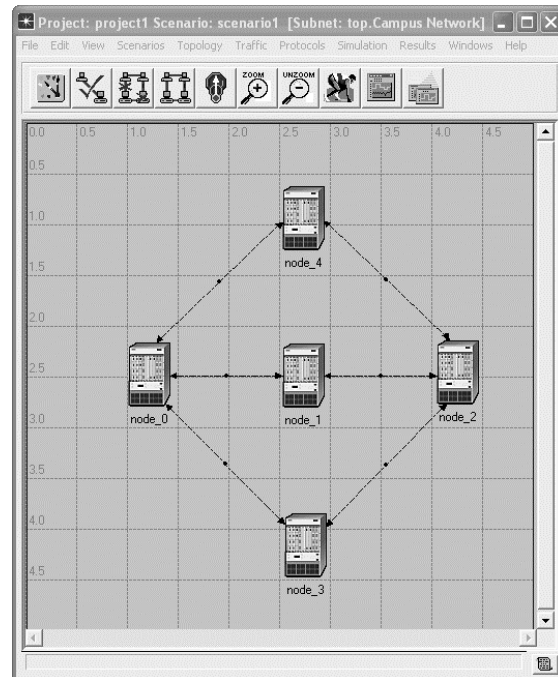


Figure 3. Simple topology for routing convergence.

Dest. Name	Address	Metric	Subnet Mask	Protocol	Next Hop	Insertion Time	Interface
0	192.0.0.0	0	255.255.255.0	Direct	192.0.0.1	0.000	IF0
0	192.0.1.0	0	255.255.255.0	Direct	192.0.1.1	0.000	IF1
0	192.0.2.0	0	255.255.255.0	Direct	192.0.2.1	0.000	IF2
0	192.0.3.0	0	255.255.255.0	Direct	192.0.3.1	0.000	Loopback
1	192.0.4.0	5	255.255.255.0	RIP	192.0.0.2	5.001	IF0
1	192.0.5.0	5	255.255.255.0	RIP	192.0.0.2	5.001	IF0
1	192.0.6.0	5	255.255.255.0	RIP	192.0.2.2	5.001	IF2
1	192.0.10.0	5	255.255.255.0	RIP	192.0.2.2	5.001	IF2
1	192.0.7.0	5	255.255.255.0	RIP	192.0.1.2	5.001	IF1
1	192.0.9.0	5	255.255.255.0	RIP	192.0.1.2	5.001	IF1
2	192.0.8.0	6	255.255.255.0	RIP	192.0.1.2	6.619	IF1

Figure 4. Routing table for node 0 after 8 seconds of simulation time.

Coding a program that simulates a routing protocol, creates routing tables, and allows a user to follow a route from one node to another is a rather complex task. An input file of nodes and weights that will represent the network topology will need to be provided by the instructor. The student's program should read this file, simulate the routing algorithm,

and create routing tables. The students will need to write another program to follow the routing tables from a source node to a destination node to confirm the performance of their program. A description of the programming assignment follows:

*The Distance Vector Routing algorithm was one of the first routing algorithms used on the Internet and is part of the RIP protocol. For this assignment, you will need to write a program which simulates the behavior of the Distance Vector Routing Algorithm. Details are below: Assume that there are 5 nodes in the graph. You will be supplied with an input file which will have a triple <nodeA> <nodeB> <weight> on each line. This will indicate the links between nodes of the graph and their weight. Assume that links are bi-directional with the same weight in each direction. If a link does not exist, assign it a weight of infinity. To begin you will need to read this file and store the information regarding the initial shape and link values of the graph. Next, you will need to run the Distance Vector Routing algorithm until it converges (routes stop changing).*

*After convergence, your program should print the following: the routing table for each node, the number of cycles it took to converge. Your program should then query the user for a source and destination node. Given these 2 nodes, you should then inspect your routing tables and print out the route from the source to the destination and the weight of that route.*

From our experience, the disadvantage of coding this particular problem is that the problem is complex enough that it will take a good investment of student time in order to complete the project. Students often get bogged down with the coding details and lose sight of the purpose of the assignment. Many will not be able to write programs that generate correct results. One technique that we find aids success is to convince the students to draw the graph and run the routing algorithm by hand prior to coding. This helps them visualize what their program needs to do and thus makes design and coding a bit easier. Students who complete this project successfully have recounted that they feel they have gained coding experience, but have not gained any insight on routing and convergence. For this particular project, a simulation might appear to be better at teaching the concepts. The simulation allows the students to focus on the main points of the lesson, routing and convergence, without getting lost in the coding details.

## 4. Conclusion

We have discussed the advantages and disadvantages of simulation versus programming for computer networking courses. Two case study

activities were described: the first, compares the round-trip packet delay time for both TCP and UDP protocols, the second looks at routing protocol convergence time. For both studies, students either wrote a program that implemented the protocols being tested or created a simulation using OPNET Modeler. Student opinion reflects that simulation software allows more concentration on the performance of the protocol and allows for an easier visualization of results. With simulations, students are more apt to “play” with their simulations and experiment beyond the bounds of the assignment. Programming, though viewed by students as more difficult, allows them to practice and gain skills that will be useful in their future careers. The authors feel that a balance of both programming and simulation activities is the best practice.

## 5. References

- [1] Christianson and Brown, *OPNET Lab Manual to Accompany Business Data Communications* by William Stallings, 5th Edition, Pearson Prentice Hall 2005.
- [2] Brown and Christianson, *OPNET Lab Manual to Accompany Data and Computer Communications 7<sup>th</sup> Edition and Computer Networking with Internet Protocols and Technology*, 4th Edition, by William Stallings, 5th Edition, Pearson Prentice Hall 2005.
- [3] Brown and Christianson, *Networking Lab Exercise for The Networking and Data Communications Laboratory Manual*, edited by Frances S. Grodzinsky, published by Prentice Hall.  
<http://www.prenhall.com/grodzinsky>.
- [4] OPNET IT Guru <http://www.opnet.com>