INNOVATIVE AND EASY-TO-DEPLOY COMMUNICATION NETWORKING LABORATORY EXPERIMENTS FOR ELECTRICAL AND COMPUTER ENGINEERING STUDENTS

Catherine Rosenberg[1], and Simon G. M. Koo[2]

Abstract *— This paper describes a set of new laboratory experiments developed for the senior level undergraduate networking course at the School of Electrical and Computer Engineering, Purdue University. These experiments differ from existing networking experiments in that they emphasize the understanding of the dynamics of network protocols in addition to their applications, which makes them more suitable for ECE curricula. Students learn through these experiments a wide range of networking aspects including the design and the limitations of protocols, simulation and performance evaluation techniques, interpretation of data, packet analysis and network programming. They are also required to read standardization documents in order to put them in a realistic engineering situation. All the experiments are software based, and the resource required is a workstation with C compiler and some public-domain software like Ethereal, so there is virtually no costs in deploying the experiments, and they can be done in almost any general-purpose computer laboratory.*

Index Terms — Electrical Engineering curricula, Networking experiment, Network Protocols, Simulation.

INTRODUCTION

"Introduction to Computer Communication Networks" is currently an experimental course offered by the School of Electrical and Computer Engineering of Purdue University, West Lafayette. The course was first offered in Spring 2000 and will be continued to offer once a year. It is intended to provide basic and yet essential knowledge about networking for seniors in ECE curriculum. Since networking by itself is a subject, which requires both a solid theoretical foundation and practical skills, the Professor in Charge (PIC) of the course, Dr. Rosenberg decided to add a laboratory component to the course in Fall 2001 [1]. She discussed the project with her PhD students at the beginning of Summer 2001, and one of them (Simon Koo) showed a lot of interest and volunteered to help. The PIC recruited one undergraduate student during that summer to help develop the experiments and the TA for the course tested the experiments and prepared the solutions at the beginning of the Fall semester. Hence these new experiments were designed by a team comprising the PIC, a PhD student, a TA (MS student), and a fourth year undergraduate student who had taken the course the previous semester.

There are a lot of existing laboratory exercises developed for networking class. However, few of them addressed the need of understanding the fundamental ideas. These experiments either focus on implementation (e.g., network programming) or configuration and measurement (e.g., configuring routing tables, measure packet loss ratio). Furthermore, these laboratory exercises usually require expensive hardware, and a lot of laboratory management and configuration.

The approach for teaching Networking in CS departments is usually different from that in ECE departments. For the first course in networking, CS classes usually take the top-down approach and focus on system design and implementation. The efforts are usually put on understanding the higher layer protocols and application of networking. On the other hand, in ECE classes, bottom-up approaches are usually used and focus is put more on lower layer protocols and performance of networks. Understanding the fundamentals is the heart of these courses.

In Purdue, we have the School of Technology in addition to the School of Engineering. The School of Technology offers courses and laboratories classes, which provide practical knowledge on networking, like configuration of a router, setting up servers, etc for students who want to be network administrators. Students who are interested in this type of career can take classes from the Technology School. The ECE course focuses on engineering aspects. However, not many universities have a Technology School so their engineering curriculum in networking has to fulfill the needs of students who are interested in training to become network administrator and of students who are more interested in a career in network engineering.

When we decided to create a set of laboratory experiments for ECE students, we agreed upon a set of characteristics. First the experiments should be easy to deploy, and not limited by the size of the class and availability of equipment. All a student needs is a general-purpose workstation, which has C compiler and a free software called Ethereal [2] pre-installed. Second, the experiments should put the students in a situation as close as possible to a situation that they would encounter in industry and should make them think and read standardization documents. We took the approach not to define the experiments too precisely. We also put a feedback mechanism in place to improve the clarity of the manual and

[1] Catherine Rosenberg, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN 47907 cath@ecn.purdue.edu
[2] Simon G. M. Koo, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN 47907 koo@ecn.purdue.edu

0-7803-7444-4/02/$17.00 © 2002 IEEE

created a website with an online chatboard, where students can post their questions, discuss their problem, both with their fellow classmates and the TA.

The organization of the remaining sections is as follows: We describe the details of each experiment, what we expect from students, and what they can learn through the experiments. We also describe the difficulties that the students encountered and the way the team dealt with these difficulties. The paper continues with a comparison of the performances of students from the previous classes without the laboratory component with the performances of the students from the class that took these experiments. Feedback from students will also be presented. A brief conclusion summarizes the main features of this paper.

CONTENT OF THE EXPERIMENTS

The laboratory experiments offered in Fall 2001 consist of four experiments. They cover the basis of simulation, data link layer and Automatic Repeat reQuest (ARQ) protocols, packet encapsulation and network utilities, and network programming. These experiments briefly covered most areas in networking, so they are useful not only to students who want to work as practicing network engineers but also to those who will need to develop applications or work in domains related to networking, e.g., communications. There was no laboratory periods dedicated to the experiments, they were just assigned and due back according to the schedule, so students could work at their own pace. The only pre-requisites for taking this class were a basic programming knowledge and an introductory course in probability.

Experiment 1: Basis of Simulation

The first experiment requires the students to develop a simulation program to understand the performance of a flow of packets of fixed size L, entering a buffer of size K and which output is a communication link with capacity C (see Figure 1). The system composed of a buffer and a link is really the basic building block of any packet switched network and its behavior should be understood. The experiment consists of developing the simulator, running it for different sets of parameters, collecting data, and very importantly, interpreting the results. The students have to use some engineering common sense to decide the length of a simulation and the validity of their results.

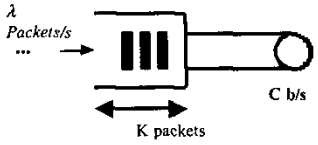

FIGURE. 1
MODEL OF A COMMUNICATION LINK

Simulation is an important tool in engineering. It is often used to evaluate the performance of complex systems, which cannot be evaluated through an exact analysis. Often only an approximate analysis can be performed and the results should be validated by simulation.

Since simulation techniques and methodologies are not usually taught in an ECE curriculum, we though that it would be beneficial to the students to learn the basis of simulation. Details about simulation are discussed in the lab manual but are intended to be minimal as this is not a class on simulation. The intent is to make the students think about what are the issues with creating and using a simulator since several of them would have most probably to create some kind of simulator in their first few years in industry. The manual includes for a brief description of *Discrete Event Simulation (DES)* [3], generation of exponential random variables from uniform random variable, and basic input/output analysis.

Students are expected to use a general-purpose language, e.g., C/C++ in our case, together with the DES framework, to build a simulator and illustrate the behavior of the communication system. They are expected to see how the average sojourn time of a packet in the system, the average buffer length, and the loss probability evolve when parameters like C, L, K and λ (the arrival rate) change. They have to generate a flow of packets using a Poisson Process. They have to explain how to do it and provide test to validate the arrival process they generate. Figure 2 shows an example of a plot that the students have to generate based on their simulation results.

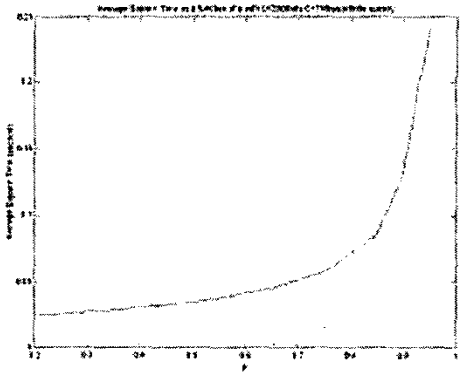

FIGURE. 2
A SAMPLE OF PLOT EXPECTED FROM THE STUDENTS
(AVERAGE SOJOURN TIME VS LOAD IN THE INFINITE BUFFER CASE)

One of the main tasks of the students is to interpret the results they obtained. For example, in one of the questions we ask them to consider the case of an infinite buffer and to explain what happen if the arrival rate of the packets is higher than the capacity. The answer is that it is impossible to obtain a "stable" result, i.e., the average delay increases (as opposed to stabilize) when the simulation run time is increased.

After this experiment, students should understand (i) the basics of simulation, and (ii) the behavior of simple communication links with buffer with different parameters. We believe that this experiment provides a good chance for those who have no simulation experience to get a first exposure in this important domain, for everybody to think in terms of data interpretation as opposed to only thinking of writing a simulation program and let others worry about the use of this program, and to build a foundation for those who would like to continue working in this area.

This experiment serves as a pre-requisite for the next experiment, which objective is to simulate the behavior of different ARQ protocols in data link layer.

Experiment 2: Data Link Layer and ARQ Protocols

The second experiment requires the students to simulate several data link layer ARQ protocols.

In the OSI model, the data link layer (layer 2) handles data transfer over a single, bi-directional communication link without relays or intermediate nodes. The link can be point-to-point, broadcast, or switched, and most local area networks are treated as single links from the point of view of the data link layer (called LLC (Logical Link Control) in that context, performing on top of the MAC (Multiple-Access Control) sub-layer). The data link layer supervises the physical transmission of frames by the physical layer, performs error control over the link (at least error detection), and possibly initiates retransmission when errors are detected and cannot be corrected. The relationship between the layers 1, 2 and 3 is shown in Figure 3. Usually layer 2 protocols are the first protocols seen by students in a networking class.

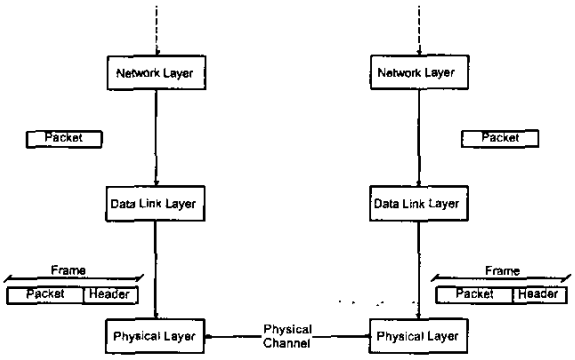

FIGURE. 3
RELATIONSHIP BETWEEN DIFFERENT LAYERS, PACKETS AND FRAMES

There are two main types of error control: *Forward Error Correction (FEC) and Automatic Repeat reQuest (ARQ)*. FEC will embed *error correction code* into the frame and the receiver will try to correct the errors, while ARQ will embed *error detection code* to the frame and the receiver will detect errors and request for retransmission if errors occur. Although ARQ involves extra overhead for retransmission requests and retransmitting frames in error, the total overhead for data links with reasonable error rates is often far less than it would be with FEC.

In this experiment, students are required to investigate three different ARQ protocols, namely, *Alternating Bit Protocol (ABP)*, *Go Back N (GBN)*, and *Selective Repeat Protocol (SRP)*. They will focus on one direction for data transmission, but they are also reminded that in general data flows in both directions.

After the completion of experiment 1, students are assumed to have basic knowledge about DES. For this experiment, they will receive a C library file consisting of a channel function from the instructor. They are then asked to design simulators for the sender side and the receiver side of different ARQ protocols based on DES, and link their simulators to the provided channel function. The channel function will generate random errors and losses according to the parameters pre-set by the students, so the students can see how the protocols perform in terms of efficiency under different error rates and compare the performances of the three protocols. A sample plot required in the lab report is shown in Figure. 4. One of the important objectives of this experiment is to make the students realize that an in-depth understanding of a protocol is needed to design a simulator for that protocol and that a tool to specify a protocol such as a finite state machine can help a lot in designing a protocol simulator.

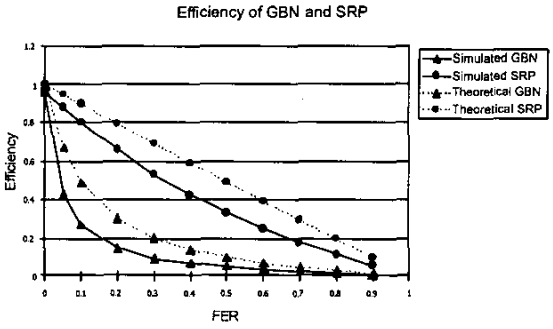

FIGURE. 4
SAMPLE PLOT FOR EXPERIMENT 2 (COMPARING EFFICIENCY BETWEEN GBN AND SRP, SIMULATED VALUES AND THEORETICAL VALUES)

After this experiment, students should have an in-depth understanding of (i) how data link layer and different ARQ protocols work, and (ii) the effect of different parameters on the performance of the protocols, including channel capacity, propagation delay, size of data frame, size of ack frame, and error ratio. We believe that by actually designing the protocols and evaluating their performance; students will have a much better understanding of the protocols than if only learning through lectures. The experiment also

provides a first glance at protocol specification and validation, and furthers the students' simulation knowledge.

Experiment 3: Packet Encapsulation and Network Utilities

Two of the most important concepts of networking are encapsulation and multiplexing. The third experiment consists of three parts. The first part requires students to analyze some typical frames to get a better grasp at encapsulation and multiplexing. In the second part they will be given a trace of several frames and have to analyze the "connection" scenario. The third part requires students to use some basic network utilities to monitor network performance.

In this experiment, we use Ethereal [2], a public-domain software to capture some specific Ethernet frames. These include TCP, UDP, ICMP, and ARP frames. Figure. 5 shows a sample frame. Students are required to interpret (without the help of any software, i.e., "by hand") every byte in the given frames in details.

```
00 00 0c d9 fa 88 00 00 b4 a0 15 c1 08 00 45 00
00 28 04 04 40 00 80 06 42 a0 80 d3 a0 3c 80 0a
13 14 04 3a 00 15 54 f1 f2 09 d6 7d df 9d 50 10
40 5a b9 e8 00 00
```

FIGURE. 5
A SAMPLE CAPTURED ETHERNET FRAME (A TCP ACK FRAME)

In order for the students to be able to interpret the frames given to them, they are required to read some RFCs (Request For Comments) which contain the standards of the protocols. RFCs listed in the manual include RFC768 for UDP [4], RFC791 for IP [5], RFC792 for ICMP [6] and RFC793 for TCP [7]. Each student will be given three different kinds of frames to analyze for this part of the experiment. This gives them an opportunity to visualize the processes of encapsulation and multiplexing.

In the second part of the experiment, students are given a trace captured by Ethereal. The equipment needed is again a general-purpose workstation, PC/Mac/Solaris/Linux based, on which Ethereal is installed. Ethereal is a public-domain software available for various OS which can capture all the frames on the Ethernet segment. Due to security reasons, it requires administrator/root privilege to install and capture frames, but a general user can also use Ethereal to read traffic traces captured by Ethereal. A screen capture is shown in Figure 6.

We currently have three traces prepared. One is a FTP session which also contains DNS (Domain Name System) resolution. The second one is a HTTP connection to the instructor's homepage and the last one is a ping echo-request/echo-reply scenario with ARP. Students are asked to draw the timing diagram of two of the above three scenarios based on the traffic trace, illustrating three-way handshake, showing connection port numbers, etc. They also need to estimate the round-trip-time between the client and the server based on the trace.

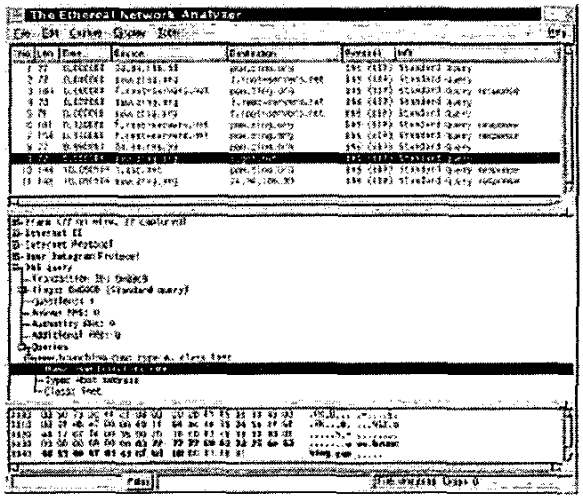

FIGURE. 6
SCREEN CAPTURE OF ETHEREAL
(FROM THE WEBSITE OF ETHEREAL AT WWW.ETHEREAL.COM)

The third part of the experiment involves the use of network utilities. This includes `arp`, `ifconfig`, `nslookup`, `netstat`, `ping` and `traceroute`. Students are required to use these commands to help themselves answer the questions in the manual. Some interesting scenarios are used in the exercises. For example, students are asked to figure out if a satellite link is used in the connection between Purdue University and Indian Institute of Science (http://ece.iisc.ernet.in) by using `traceroute`.

After this experiment, students understand (i) the header format of standard protocols and know how to interpret packet headers by reading RFCs, (ii) how to use basic network utilities to monitor network performance, and (iii) how to use Ethereal to analyze frames. We believe that this experiment provides a solid view of packet encapsulation, and the structure of TCP/IP packet format. This will be useful to anyone whose work involves the Internet. The use of network utilities, on the other hand, gives the students a chance to see how the network is actually performing, and allow them to deduce some implicit information (the satellite link example mentioned earlier) within the network.

Experiment 4: Network Programming and Application Layer Protocol

This experiment requires students to build a simplified HTTP client and server, using network programming. This is just a quick introduction to network programming to make the students understand that they can build network applications without having to bother about the network.

Typical network applications are running in a *client/server* mode. A client usually *initiates* a connection and requests service(s) from the server while a server usually

listens and waits for incoming connection requests and provides services. An example will be a web browser (client) connecting to a web server and asking for an HTML file. The client/server application sits on top of an *application layer protocol*, which defines the messages that are exchanged by the peer processes and the actions taken. HTTP (HyperText Transfer Protocol) [8] is an example of an application layer protocol. It defines the (HTTP) transactions between a web browser and a web server. An application layer process in a remote host (either client or server) uses the services provided by the transport layer, to communicate with its peer process running in the other remote host (either server or client).

In this experiment, students are required to create network applications on top of TCP using network programming (also called *socket programming*) [9]. They will be asked to build a simplified HTTP client, and a simplified HTTP server which can be accessed by a typical web browser.

The equipment required is a workstation with C compiler and socket programming header files. Unix or Linux based machines are preferable (we used Linux), but Windows based machines can also be used if some library functions are given.

The students are asked to read RFC 1945 [8] and are given a binary executable which is a simplified HTTP server. The first task is to create a HTTP client that connects to a server provided by the instructor, and use the command GET to obtain a specific file. Next, they will have to create a simplified HTTP server, which enable a minimum set of HTTP commands defined in the manual, and use the clients they created, as well as a commercial browser to connect to it. They are required to build a server that can accept one connection at a time, however, they are also encouraged to build a server that can accept multiple connections for extra credits.

After this experiment, students are able to (i) understand the client/server paradigm of network applications, (ii) create simple network applications, and (iii) build a simplified HTTP client/server. For many of the EE students, this experiment is the first time they do network programming. We believe that this experiment provides a chance for students to understand and experience the client-server paradigm, and provide experiences for actual networking software development. They can also have a better idea how the application, application layer protocol and the transport layer interact.

HOW WELL DID THEY DO?

Since there is no official lab period for the experiments, we provided a help session after each experiment was assigned, and web-based chatboard for students to discuss and ask questions about the experiments. Here are the most common mistakes or questions that we have encountered:

- (Exp. 1) Some students are not well prepared in basic probability theory. They need to have a solid understanding of cumulative distribution functions, exponential distribution to be able to generate a Poisson arrival process and to test its validity.
- (Exp. 1) Some students generates all the event happening times in a text file and run it later instead of doing it one by one which means that they are very limited in the size (or duration) of their simulations.
- (Exp. 1 & 2) Students failed to recognize that simulation may take a long time to run, and thus spent a long time on program development and not enough time to obtain reasonable amount of statistically valid data.
- (Exp. 4) Students are generally able to do the socket programming part, however, a lot of them have trouble when doing text processing after getting the data from the network.
- (Exp. 4) Some commercial browsers send extra header information, which confused the students. As we are only interested in implementing a partial set of commands, students should ignore the additional ones.

In general most of the students are able to complete the tasks but the level of programming skills varies widely. They perform best in experiment 3, which does not have any programming.

COMPARISON OF PERFORMANCE

The Course Evaluation showed a large increase of satisfaction due to the introduction of the experiments. There were a larger number of students who dropped the course (mostly EE students who were afraid of the programming content of the experiments) but the students who remained showed much more motivation and interest in the class. The class GPA was around 2/4 in Spring 2001 when there were no experiments and was 2.6/4 in Fall 2001 when the experiments were introduced.

FUTURE WORK AND CONCLUSION

After offering these experiments for one semester, we believe that there is a need to continue offering them. The coming version of the manual has been revised based on the feedback of the students. Based on the experiences we have acquired when developing this set of experiments, we are working on developing a set of experiments for the graduate networking class.

ACKNOWLEDGMENT

The authors would like to acknowledge H. K. Chai and Y-. C. Lee for their efforts on testing and preparing the solution for the experiments.

REFERENCES

[1] Rosenberg, C., Koo, S. G. M., Chai, H. K., and Lee Y-C, *Introduction to Computer Communication Network Lab Experiments*, Laboratory Manual, School of ECE, Purdue University, 2001.

[2] The Ethereal Network Analyzer <http://www.ethereal.com/>

[3] Law, A. M., and Kelton, W. D., *Simulation Modeling & Analysis*, McGraw-Hill, 1991.

[4] Postel, J., User Datagram Protocol, RFC 768, Aug 1980.

[5] Postel, J., Internet Protocol, RFC 791, Sep 1981.

[6] Postel, J., Internet Control Message Protocol, RFC 792, Sep 1981.

[7] Postel, J., Transmission Control Protocol, RFC 793, Sep 1981.

[8] Berners-Lee, T., Fielding, R., and Frystyk, H. Hypertext Transfer Protocol -- HTTP/1.0, RFC 1945, May 1996.

[9] Stevens, R., *UNIX Network Programming*, Prentice Hall, 1990.

[10] The Network Simulator – ns2 <http://www.isi.edu/nsnam/>