A Sequence of Closed Laboratory Exercises for a Course in Data Communications

Sanjay P. Ahuja
Associate Professor
Department of Computer and Information Sciences
University of North Florida
Jacksonville, FL 32224.
sahuja@unf.edu

Abstract
A sequence of closed laboratory exercises for a course in Data Communications is described in this paper. The exercises are a combination of client-server programming, and simulation. The socket API using C/C++ is used for the client-server programming exercises. This allows students an opportunity to look under the hood and use networking protocols like TCP and UDP to develop their own client-server applications. COMNET III is the tool used to simulate communication networks. It is used to model networks, their control algorithms, and workload. Network simulations add a practical flavor to the course and enable students to observe the operation of an internet comprising dissimilar networks and protocols.
Keywords: Data communications, networks, closed labs, education.

I. Introduction

The field of Computer Networking is making rapid advances. Courses in Data Communications and Networking are now offered on a regular basis in Computer Science programs. The course generally includes topics in data communications, computer networking, and internetworking. Laboratory exercises in support of this course provide a rich learning environment. However, developing and testing laboratory exercises require a substantial time commitment on the part of the instructor wanting to develop this type of pedagogy. This paper describes the lab exercises developed and tested by the author. It is hoped that the lab exercises described here could serve as a starting point to an instructor wanting to develop a lab for a similar course.

In a course on Data Communications and networking, it is essential that students be exposed to the client-server paradigm. Programming exercises are very valuable in this regard. With the advent of Java and its inherent support for networking and distributed computing one possible approach is to

use Java as the basis for client-server programming exercises. Java provides the java.net package to support this. However there is an ongoing debate among faculty whether this approach is the best in a course on Data Communications where the intent is to study networking protocols close up. In the author's opinion, the Berkeley sockets API that was developed for the C programming language, is preferable since it allows a student to look under the hood as it were. 4.3BSD supports the UNIX domain, the Internet domain (TCP/IP and UDP/IP), and the Xerox NS domain communication protocols [1]. The socket API enables students to learn about the socket system calls, the socket address structure, potential byte order differences between architectures and protocols, and the byte-stream oriented nature of TCP and how to deal with it in their applications. Much of this is hidden from view as a result of the abstraction provided by Java's java.net package. While this would be a plus in developing substantial distributed systems it takes away information which is important in the context of a Data Communications course.

Simulations have been shown to be useful in the academic environment [2,3] and allow students to comprehend networks at a deeper level than is possible with mathematical analysis alone. COMNET III is a network simulator and seeks to serve as a teaching aid in a data communications and networks course. It enables the instructor to include realistic network design and optimization as part of the course [4]. COMNET III can be used to model LANs (e.g. Ethernet, Token Ring, and FDDI) and LAN interconnects. Packet switched, circuit switched, and ATM networks can also be modeled.

The rest of the paper is organized as follows. A brief description of COMNET III is included in section 2. Section 3 describes the lab exercises developed by the author. Conclusions are described in section 4.

II. Description of COMNET III

A very brief introduction to COMNET III and its user interface follows. More details are available from the full COMNET III documentation, which can be obtained from CACI Products Company or from an earlier paper by the author [5].

COMNET III's object-oriented design combines LAN, MAN, and WAN performance prediction in a single environment [5]. A user of COMNET III builds models by graphically selecting palette icons representing nodes, links, protocols and traffic, and positioning them on the screen using a mouse. The steps involved in building a COMNET III model follow next. The network topology is generally built first, followed by traffic and workload sources, and setting up the parameters for network operation. COMNET III provides an extensive library of nodes, links, protocols, and traffic objects.

Nodes that represent computers or switches, and links that carry traffic between nodes define the network topology. Traffic can be generated and received by these nodes. Processor utilization, internal storage, bridges, routers, and communications switches (including ATM switches) can be modeled. The library of link objects includes two classes: point-to-point links that represent a channel between nodes, and multi-access links for LANs. Multi-access protocols that can be modeled include CSMA/CD, CSMA, ALOHA, Token Bus, Token Ring, FDDI and Polling.

The sources for network traffic and workload drive the simulation. Network traffic refers to the messages sent between nodes, and workload refers to the internal activities of the node's processors or busses. COMNET III has two kinds of sources: Application Sources that execute commands introduce either traffic into the network or workload inside the node, and Traffic Sources that generate traffic between nodes. Commands that a node can execute include Process Data, Read File, Write File, Transport Message, Answer Message, and Setup Session. Sequences of these commands are associated with Application Sources attached to a node. These applications can be scheduled to occur according to an interarrival time distribution or when a certain incoming message is received.

Messages are transported from source to destination using Transport, Routing, Data Link, and Medium Access Control Protocol objects. Protocol suites such as TCP/IP, IPX, SNA, DECNet can be modeled. Reports produced by COMNET III* include: Node utilization and application delays, link delays and utilization, message delays, packet delays, calls blocked, disconnected and preempted, session setup delay etc.

III. Sequence of Laboratory Exercises

The author has put together a sequence of lab exercises using COMNET III and programming with the socket API. They comprise a 12-week lab schedule. The first 7 labs are based on COMNET III. Labs 8, 9, 10, and 11 are based on socket programming. The complete schedule is as follows.

Lab 1
This is a tutorial on COMNET III. It introduces students to the GUI environment and illustrates what the various icons represent. This includes the various commands, nodes, links, and traffic sources.

Lab 2
This lab studies the performance of a CSMA/CD LAN (i.e. the Ethernet). A model of a 10 Mbps Ethernet LAN is built. Traffic modeled includes email and ftp requests and responses. The performance of the Ethernet is observed and the effect of varying load on the line utilization and the average message delay is noted. Next, the Ethernet is replaced with a Fast Ethernet and the difference in performance observed. The effect of increasing load on a CSMA/CD LAN is noted. This includes a study of the message delays, ftp setup delays, number of collisions, mean number of attempts needed to resolve a collision, link utilization.

Lab 3
This lab exercise studies a round robin protocol based LAN i.e. the Token Ring LAN. The performance of both the 4 Mbps and 16 Mbps Token Ring is observed. A client-server setup is modeled with workstations making requesting files from a file server at random intervals. The file size sent by the server is progressively increased and the effect of this increasing load on the Token Ring is observed. Particularly, the link utilization, token rotation time, and the file transfer delays are noted.

Lab 4
A LAN-LAN internetwork is modeled with an Ethernet being connected to a Token Ring via a

router. Each LAN has its clients and a server. The traffic is modeled such that 70% of a client's requests are for its own server and 30% of its requests are for the remote server. The variance in frame size is observed as a frame makes its way to a dissimilar LAN. A comparison of the link utilization's and average message delays of the two dissimilar LANs is made. This lab exercise allows students to compare two different LANs.

Lab 5

A LAN-WAN internet is modeled. A company has 4 LANs (presently Ethernet), one each at its 4 offices. These LANs are connected via a WAN. The capacity of the line connecting the LAN's router to the WAN is upgraded from an initial 56 Kbps to a T1. The WAN itself uses T1 links. Traffic is modeled such that a client can request a file from its own server or across the WAN to any of the other servers. The impact of upgrading the capacity of the link connecting a router to the WAN is studied. Students had to make the determination whether improved performance achieved by upgrading the LANs to Fast Ethernet's or the access link (from 56 Kbps to T1) to the WAN justified the upgrade. It is not intuitively obvious which upgrade (if any) is required. A simulation study is very helpful in arriving at an informed decision.

Lab 6

The topology used in this lab exercise is the same as in lab 5. The TCP and UDP transport protocol performance is studied. The efficiency of the transport protocol used and form of flow control (sliding window size is varied from 1, 7, to 127) used are evaluated based on the average message delays to send a file, the average message delay to setup the ftp sessions, and the link utilization's.

Lab 7

A campus LAN situation is modeled. A campus has several Ethernet and Fast Ethernet LANs. A decision needs to be made whether to replace the existing FDDI backbone with an ATM network. A performance study is carried out.

Labs 8 and 9

A UDP based connectionless, iterative Echo Server is designed and coded in C/C++. The UDP client process allows the user to connect to the server process running in the background on a remote host at a specific port number. The client reads a line from its standard input, takes a time stamp, and writes it to the server. The server reads a line from its network input echoes this line back to the client. The client reads the echoed line, again takes a time stamp and determines the round trip time (RTT). It also compares the line it sent to the server with the line it received back. It reports the RTT to the user and whether there were any errors in transmission of the line.

A pair of client-server programs to echo input lines is a good example of a network application. All the steps usually required to implement any client-server are illustrated by this example. This echo example can be easily expanded into other iterative applications by changing what the server does with the input received from its clients.

Labs 10 and 11

A TCP based connection oriented, concurrent simple FTP client-server is designed and coded in

C/C++. The application behaves as a simple FTP system with the client allowing the user to connect to a remote host to retrieve or send files. The commands supported include SEND, GET, and BYE. The server process is concurrent in nature and forks a child process to service each client request. This illustrates the principles of concurrency and allows the student to compare this with the iterative server. The FTP is an application that students are familiar with and they derive satisfaction of having developed a working, albeit simpler, version of it. Students get a clear understanding of the byte-stream nature of TCP. After having completed labs 8, 9, 10, and 11, students have a solid exposure to socket system calls, and the client-server paradigm.

Lab 12

This lab deals with UNIX networking utilities like *ping, traceroute, netstat, and nslookup.* Students are asked to read the man pages to learn about each of these utilities. They use *ping* to determine the RTT to various hosts in the Internet. Students use *netstat* to display the current IP routing table on their host. The utility *traceroute* is used to determine the number of hops from a student's host to other hosts in the Internet. Students use *nslookup* to determine the sequence of name servers to contact to resolve a specific domain name of a host in the Internet.

COMNET III is presently running on a Windows platform. The department purchased a multi-site license for approximately $3000. COMNET III is also available for many UNIX flavors.

Students worked in pairs in all the labs. Their feedback on COMNET III was solicited in the form of a questionnaire towards the end of the semester and has been encouraging. Student comments have indicated that COMNET III is a good visual tool for observing the flow of packets and frames, covers nearly all kinds of networks, and is an excellent aid in observing the operation of an internet comprising dissimilar networks and protocols. Asked to rate their overall experience with COMNET III, the majority of students rated their experience with COMNET III as being good.

The author has observed that it would be helpful to explain to students how to read, interpret, and draw conclusions from the reports generated by a simulation study carried out using COMNET III. It is also helpful to hand out labs a couple of days prior to the lab itself. This gives students a chance to familiarize themselves with the lab. Further, a demo using a LCD panel would be helpful. Care must be taken while designing and adding details to labs so that the simulations do not run very slowly given that several simulation runs typically need to be made in a 2-hour lab session. It is to be noted that adding too much detail to a simulation can be counterproductive. COMNET III's object-oriented GUI increases ease of use and reduces time spent entering network description from the student's perspective.

Prior to beginning the labs on socket programming it is imperative that the instructor discuss in class the socket API with all the relevant socket system calls (socket(), bind(), connect() etc.), the client-server paradigm, connectionless versus connection oriented servers, iterative versus concurrent servers, and the pros and cons of TCP and UDP. The byte-stream orientation of TCP should be also explained.

IV. Conclusions

The author has developed a lab sequence for the Data Communications and Networks class in

which both simulation and programming play an important part. The programming exercises based on the socket API have played a significant role in student comprehension of client-server application development. Should a student go on to develop large scale distributed applications using a language like Java, he/she would already have a good understanding of the protocols involved and an appreciation for the abstraction provided by a language like Java.

Simulations are a very useful tool in the academic environment. COMNET III, a network simulator, is an effective teaching aid in a communications networks course. It has been the author's experience that exposure to COMNET III has significantly enhanced both student comprehension and interest in the subject matter. This has been evidenced by student feedback

References

[1] Stevens, Richard, "Unix Network Programming", *Prentice-Hall*, pp. 258-298, 1990.

[2] McDonald, C.S., "A Network Specification Language and Execution Environment for Undergraduate Teaching", *Proceedings of the ACM Computer Science Education Technical Symposium 1991*, San Antonio, TX, SIGCSE Bulletin 23(1), pp. 25-34, March 1991.

[3] Barnett, B. L. III, "A Visual Simulator for a Simple Machine and Assembly Language", *SIGCSE Bulletin*, 27(1), pp. 233-237.

[4] "Major Applications of COMNET II.5, Network Analysis and Capacity Planning Through Simulation", *CACI Products Company*, pp. 22-23, August 1993.

[5] Ahuja, S. P., "COMNET III: A Network Simulation Laboratory Environment For A Course In Communications Networks", *Proceedings of the Frontiers In Education Conference (FIE 98)*, Tempe, AZ, November 1998.

SANJAY P. AHUJA

Sanjay P. Ahuja is an Associate Professor of Computer Science in the Department of Computer and Information Sciences at the University of North Florida. His research interests include Computer Networks, Distributed Systems, Applications of Genetic Algorithms, and Computer Science education. He is a senior member of the IEEE, and an active member of the IEEE Computer Society, ACM, and ACM SIGCSE. Dr. Ahuja received a Ph.D. in Computer Science and Engineering at the University of Louisville in 1993.

* COMNET III is a registered trademark of CACI Products Company.