Design of Innovative Computer Networking Labs for Senior and Graduate Networking Courses and Supporting Research Projects

Peter S. Lau[1]

Abstract - In computer networking technology courses, experiments with large networks are essential for students to understand the problems and complexity of real world networks, the Internet, and the underlying solutions and technologies that make our Internet so useful, scalable, and reliable. Networking equipment are very expensive and few laboratories have the funding to purchase enough equipment to build a large network. An alternative for large network experiments is to use computer simulations. This paper describes a computer simulation program written in C++ which currently supports a limited set of functions. The program was used in senior computer networking courses and as a research tool. This paper also presents a content distribution application of the program showing how a novel multi-source, multi-root multicast tree supports content servers redundancy. The source code is open to the students and faculty for further development of the program.

Keywords: computer network, computer simulation.

INTRODUCTION

The Computer Engineering program in the department of Engineering Technology offers two computer networking courses, Internet Technology and Computer Network Technology, for senior and graduate students. These two courses together give a comprehensive introduction to modern communications networks, covering all five layers of the Internet Protocol Layering Model, namely, application, transport, network, data link, and physical layer. Student labs based on physical networking devices that support different network protocols are indispensable for hands-on experience. Due to the high cost of networking equipment, only networks consisting of a few nodes are possible in our networking lab. While small networks supporting a limited number and type of topologies are suitable for certain labs such as RIP (Routing Information Protocol) in a router network to demonstrate hop-count based path selection and PPP protocol over a serial link to demonstrate robust and reliable communication between the two endpoints of a link, large networks with different topologies are necessary for studying protocols such as multicast and OSPF (Open Shortest Path First) and for evaluating performance in terms of delay, delay variation, and packet loss ratio. In addition, a large network topology is usually required to test and verify new protocols, new technology, and network configurations such as path redundancy.

An alternative to building a large physical network lab is computer network simulation. Network simulation is an economic way, often the only viable way, to evaluate the performance of large networks, to explore different network configurations and solutions, to test new protocols and new network technologies before the protocols and technologies are verified in an expensive network testbed, if any, or before committing the protocols and technologies to a production network. Network simulation is not intended to replace, but rather to complement, physical network labs.

Though there are commercial network simulation packages like the OPNET and qualNet, and open source network simulation project such ns-2, we decide to develop our own network simulation software. Beside the first hand experience of building a large computer simulation software, we can freely modify the source code without any restriction and the requirement to publish any new or novel implementation as required by the open source public license. The network simulation software will serve the dual purpose as undergraduate and graduate networking labs and research projects for graduate students in the field of computer networking. Faculties and students at the

[1]. The University of Memphis, 203 Eng. Tech. Bldg, Memphis, TN, 38152, peterlau@memphis.edu

University of Memphis will have access to the source code and are free to modify the source code. Ultimately, it is our intention to make it an open source project to the academic and Internet communities.

DESCRIPTION OF SIMULATION SOFTWARE

The simulation software is written in C++ [Deitel, 1] and consists of nine classes: packet, packet switch, lookup table entry, lookup table, buffer, input port, output port, traffic source, and traffic analyzer. The design of the different classes described below is based on a multicast content distribution application. Additional classes are currently being developed, such as lookup table for unicast application and a buffer subsystem supporting four queues of different priorities, to support a more generic computer networking simulations. The simulation software does not support GUI interface. A GUI interface for configuring a network and displaying the network map is planned and a GUI for the configuration of other objects such as packet switch object, port objects, and buffer are not yet planned.

Packet Object

Figure 1 shows a packet object which contains a Source IP Address, Group Id, Sequence Number, Priority, Hop Count, Available Bandwidth, Accumulated Delay, Preference, Packet Length, and Data field. Source IP address is the unique IP address of the source that generated the packet. Group Id identifies the multicast group the packet belongs to. The Sequence Number (SN) of a packet identifies the position of the packet in the stream of packets within a connection. The traffic source sequentially numbers the SN of the packets sent into the network. In content distribution applications described in this paper, SN is used for supporting the fault-tolerance capability. Hop Count, Available Bandwidth, and Accumulated Delay are collectively called the HBD performance parameter. Priority and HBD are used to support QoS (Quality-Of-Service) required by the applications. Preference is used to support the multi-source multicast content distribution application. Packet length is the length of the packet including the header and data. The Data field is currently empty.

Source IP Address	Group Id	Sequence Number	Priority	Hop Count	Available Bandwidth	Accumlated Delay	Preference	Packet Length	Data

Figure 1:Packet Format

Packet Switch Object

Figure 2 shows a packet switch object which contains its own coordinates and switch id in the network, a LookUp Table (LUT) object, N input port objects, and N output port objects. A packet switch can accept attachment requests from either a packet switch or traffic source and make attachment requests to other packet switch or traffic analyzer. The packet switch will only accept those packets that belong to one of the registered multicast groups and have nonzero hop count. Accepted packets are dispatched to their respective input ports for processing. After packets are processed by the input ports, the packet switch moves the processed packets from the input ports to the intended output ports. Via the packet switch object, an outbound link can be cut, buffer overflow statistics are recorded, and various thresholds are set.

Lookup Table Entry Object

The LUT object supports a configurable number of entry objects. Figure 2 shows a K-entry LUT. Figure 3 show a lookup table entry object which contains a Valid, Search Index, HBD, Preference, Inbound Group Bitmap, and Outbound Group Bitmap, Performance Count, Timeout, and Root Port field. Valid has 1 bit to indicate if this entry containing valid data. To validate an entry is to assert the Valid bit and vice versa. Search Index is used to identify the entry. Inbound and Outbound Group Bitmap are used for supporting the multicast content distribution application. Performance Count keeps the performance statistics for each input port object. Timeout invalidates the entry if no packets of the multicast group are received by the packet switch for a period exceeding the specified timeout value. This is to ensure the eventual release of the network resource reserved for the multicast forwarding if errors occurred or the multicast group has not been properly teared down. Root Port contains the id of the input port that is receiving packets of the multicast group.

Valid	Search Index	HBD	Preference	Inbound Group Bitmap	Outbound Group Bitmap	Performance Count	Timeout	Root Port

Figure 3:Lookup Table Entry Format

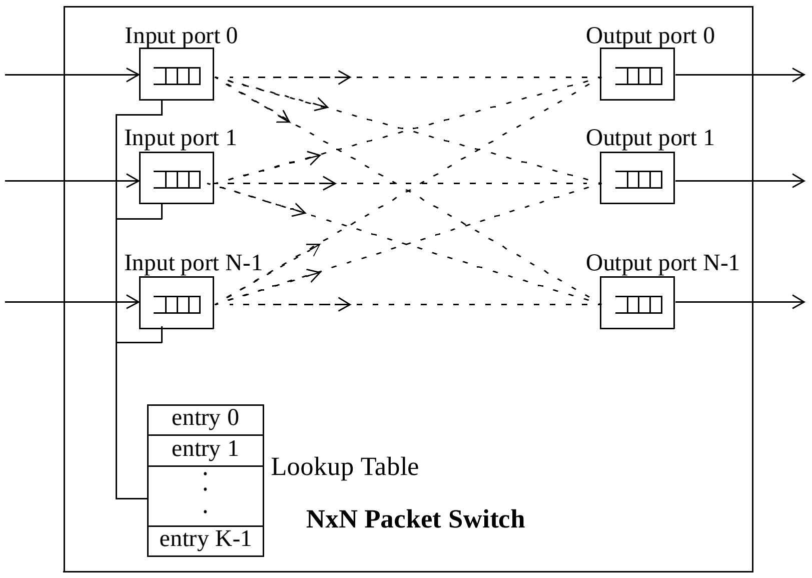

Figure 2:Packet Switch

Lookup Table (LUT) Object

The LUT object has the capability of responding to the addEntry command by validating an unused entry and returning the pointer of the newly created entry to the caller, (2) responding to a deleteEntry command by invalidating the entry specified by a search index, (3) responding to a search command by returning the pointer of the requested entry, and (4) responding to the displayFib command by displaying the entire table.

Input Port Object

An input port object receives packets from the attached packet switch or packet source and stores them in its buffer. The input port extracts the group id from the HOL (head-of-line) packet, uses the group id to search the lookup table to retrieve the entry indexed by the group id. There are four possible outcomes. The action taken by the input port for each outcome are described below. The first case is when no entry is found. The input port issues an addEntry command to the LUT to create a new entry for the multicast group and then populates the fields of the entry with the information contained in the HOL packet. The packet switch is now added as a node to the multicast tree. The second case is when an entry is found and the input port is the root port. The input port updates the entry with the information contained in the HOL packet. The third and fourth case are when an entry is found and the input port is not the root port. The input port extracts the root path's HBD and preference values from the entry, computes the performance of the path on which the packet arrived relative to the root path performance, and write the results to the entry. The third case is when the root path performance is better than the path performance of the packet. The input port simply discards the HOL packet so that the current root port remains to be the root port. The fourth case is when the path performance of the packet is better than the root path performance. The input port replaces the current root port with itself and updates the entry with the information contained in the HOL packet.

Output Port Object

Physical link speed and propagation delay are set in the output port object. Prior to transmitting a packet to the attached packet switch or traffic analyzer, an output port updates the HBD and priority of the packet.

Buffer Object

Each of the input port objects contains a buffer object and each of the output port objects contains the same buffer object as the input port. The buffer object is simply a FIFO (First-In-First-Out). The size of the FIFO is configurable. A more sophisticated buffer implementation can be written as required.

Packet Source Object

A packet source object assembles packets with the desired value for the fields of the packets sent and increments the sequence number of the next packet to be sent.

Traffic Analyzer Object

A traffic analyzer object accepts packets from the attached packet switch. The traffic analyzer computes the average delay and delay variation of the packets and packet loss ratio. Other packet analysis as required by applications can be implemented in the future.

Network Topology Generation

To simulate a computer network, a network topology is required. A network topology consists of a set of nodes interconnected by a set of bidirectional links. Users may obtain a network topology from a real world network or simulate a network topology using the same simulation software. If the network topology is simulated, N nodes are placed randomly on a GxG grid, where N and G are user configurable. A link is placed between two nodes with a probability inversely proportional to the distance between the two nodes. A user configurable connectivity factor, α, controls the probability and hence controls the connectivity of the network topology to be generated. The resulting network topology map is written to an text file.

The simulation software reads the network topology map from the file and instantiates a packet switch object per node. To place a bidirectional link between two nodes according to the network topology, the simulation software selects an output port from the upstream node and configures the output port with the handle (object pointer) of the downstream node and the input port number of the downstream node and vice versa.

STUDENT LABS

Three student labs are designed for the Internet Technology and Computer Networking Technology courses. Students are given the source code of the simulation software and have the option of doing the labs in the scheduled lab sessions or at their home. The first lab introduces students to the simulation software in which students randomly generate network topologies on a GxG grid of N nodes. A suggested value for G and N are 32 and 24 respectively. Students are asked to experiment with network topologies of different sizes and connectivities and to plot one network topology. The student then find the shortest path between three pairs of nodes by hand.   Figure 4 shows an example of network topology generated by the simulation software. The figure also shows the shortest path between nodes 2 and 6, which is $2 \rightarrow 3 \rightarrow 19 \rightarrow 6$.

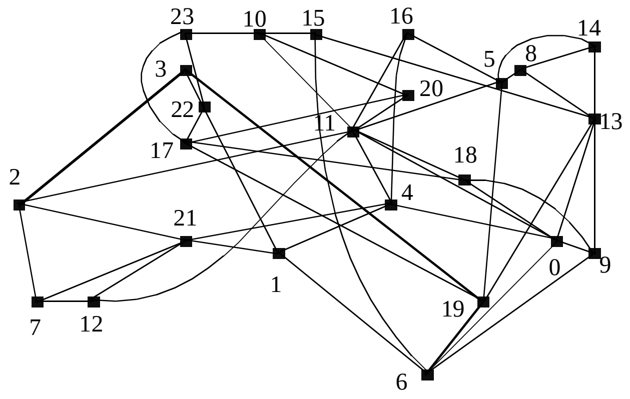

Figure 4:A simulated network of 24 nodes on a 32x32 grid.

In the second lab, each student uses the same network topology he/she used in his first lab. The students are asked to select a node from the network topology and find a multicast tree rooted at the node that spans all other nodes by hand. The students then attach a packet source to the selected packet switch, execute the multicast protocols and algorithm, and then plot the resulting tree. The students are then asked to compare the trees they generated by hand and generated by the multicast algorithm. Figure 5 shows a multicast tree root at node 6.

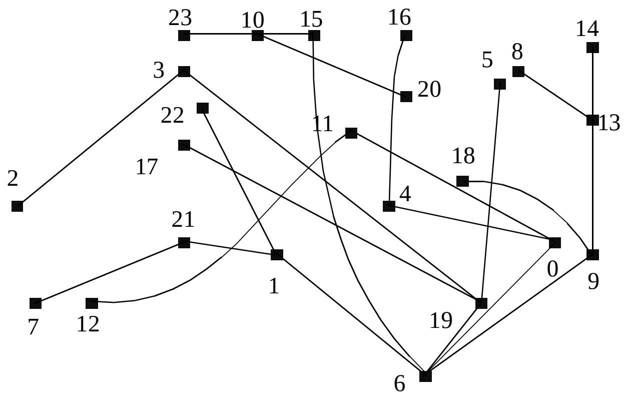

Figure 5:A Tree Rooted at Node 6.

The simulation software will be enhanced to include a new LUT entry class and an enhanced LUT class for unicast forwarding. In the third lab, student will be asked to review the RIP (Routing Information Protocol) example taught in classes and then simulate the RIP and examine the routing table of the nodes. The simulation software will also be enhanced to include a multiple priorities and a round-robin queueing system, a more sophisticated traffic source, and a more sophisticated traffic analyzer. The traffic source will inject packets into the network with Poisson(λ) distribution. The traffic analyzer will be able to collect the statistics of the packets traversing a packet switch and calculate the average delay, delay variation, and packet loss ratio. In the fourth lab, students will be asked to attach one or more traffic sources, with different arrival rates, and a traffic analyzer to the network, run the simulation, and observe the performance.

Students' Experiences

We comment on the students' experience of the simulation software for the first and second labs only because the other two labs were not ready. Unlike the commercially available simulation packages such as OPNET and the free simulation software such as the ns simulator, the simulation software package is very small and targeted for student labs so that the learning curve is not as steep as that of the other simulation packages. After studying the many network topologies generated with different connectivity factors, the students came to appreciate the complexity of large networks.

Students learned the concept of multicasting in a network and a particular multicast tree construction algorithm in the Internet Technology classes. The same multicast algorithm was implemented in the simulation software. In the second lab, each student selected different root nodes on his/her generated network topology and executed the multicast algorithm to see the formation of multicast trees rooted at different nodes. During the experimenting with the multicast algorithm, a small error in the multicast algorithm was discovered and subsequently corrected.

With a particular multicast tree, students randomly cut one or more links and saw how the algorithm automatically repair the tree by routing around the broken links to form a new tree. The students were stimulated when seeing the robustness of the multicast algorithm and developed a deeper understanding of the algorithm and real interest of computer networking protocols. The students started to appreciate the usefulness of the computer simulation technique in verifying protocols.

RESEARCH PROJECTS

The simulation software is also used in a computer networking research project conducted in the department. The project involves a new multicast algorithm for IPTV signal or VoD (Video-on-Demand) video distribution services over the Internet. In distribution services, the data generated by a source is received by two or more recipients. It is most efficient to support the distribution services using multicast forwarding provided by the transport network. Multicast forwarding requires construction of a multicast tree along which the data originated at the root of the tree duplicates at the nodes of the tree while traveling to the leaves of the tree. The set of nodes of the multicast tree must cover the nodes to which the recipients are attached. In distribution applications over the Internet, the content server converts the TV or video signal stream into a sequence of IP packets sent into the Internet. Before the content carrying packets can be transmitted into the Internet, a multicast tree, rooted at the node where the content server resides, must be first established. The multicast tree covers the nodes to which content subscribers are attached.

Fault tolerant distribution applications require fault tolerant transport and switching facility and fault tolerant content server. The former is satisfied by the multicast algorithms proposed in [Lau, 2][Lau, 3][Lau, 4]. A failure in the content server or a failure in the node hosting the content server will cause the distribution application to fail. Though redundant content servers may be hosted on a node to protect server failure, the node itself is still a single point of failure. This argues for two or more content servers hosted in two or more nodes to avoid any single point of failure. To the best of our knowledge, no prior work has been proposed to protect content server by duplicating it at multiple nodes. The multicast algorithms [1][2] readily support hosting of multiple content servers in multiple nodes to thereby providing an arbitrary degree of content server redundancy.

The common wisdom is that a multicast tree has one root. We show here that a multicast tree can have multiple roots obtained as special configurations of the multicast algorithm. Figure 6 shows a 2-root, 2-source tree and Figure 7 shows a 4-root, 4-source tree. Each source multicasts the same content to a disjoint subset of receivers. The set of the subsets of receiver covers the original set of receivers. While a subset of nodes select their respective root ports leading to a source, another subset of nodes select their root port leading to a different source. Therefore, the resulting multicast tree is made up of multiple physically disjoint subtrees each of which serving a subset of receivers. All the sources belong to the same multicast group and hence the same tree, because the packets they are generating have the same group id. It should be noted that despite the appearance of formation of physically disjoint subtrees, the underlying network topology is a single tree since all the nodes share the same group id. If all but one source fails, the tree will fall back to single-source, single-root.

The simulation software not only successfully used to verify the correct operations of the algorithm in different sizes and configurations but also used to evaluate the performance of the algorithm.

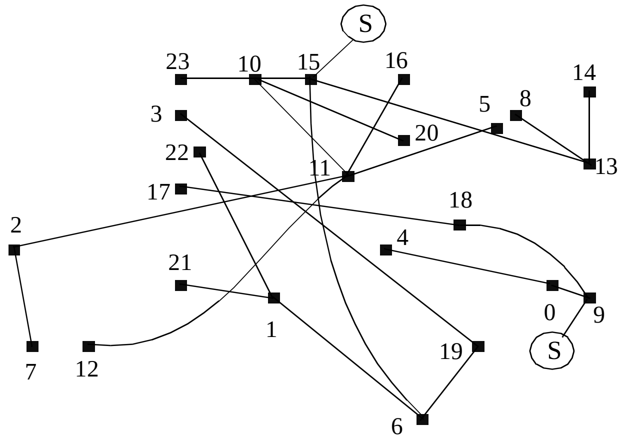

Figure 6:A 2-Source, 2-Root Tree.

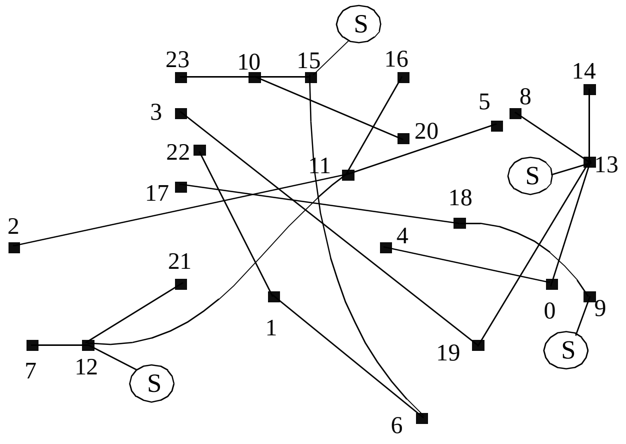

Figure 7: A 4-Source, 4-Root Tree.

CONCLUSIONS

We described the components of the network simulation software, which are based on C++. The simulation software is proved an invaluable complement to the physical network labs and a research tool in computer networking. Students used the simulation software to gain a deeper understanding of computer networking in general and a multicast algorithm in particular. Researchers used the simulation software to verify the correct operations and evaluate the performance of the network protocols.

REFERENCES

[1] Deitel and Deitel, C++ How to Program, Fifth Edition, Prentice Hall, 2005.

[2] Peter S. Lau, “A Best -Effort Multicast Algorithm,” ICCT’06, Nov. 27, 2006.

[3] Peter S. Lau, “A Fault-Tolerance Best-Effort Multicast Algorithm,” ICCT’06, Nov. 27, 2006.

[4] Peter S. Lau, “Application of the Fault-Tolerance Best-Effort Multicast Algorithm,” ICCT’06, Nov. 27, 2006.

Peter S. Lau

Dr. Lau is an Assistant Professor of Engineering Technology at The University of Memphis. He received the B.A.Sc., M.A.Sc., and Ph.D. in Electrical Engineering from The University of Toronto, Canada. Prior joining U. of Memphis, he worked in the telecommunications industry over ten years. His research interests include the applications and security of the Internet and large scale computer network simulations.