A NETWORK SIMULATOR USING STEP EXECUTION FUNCTION

Hnin Cherry
Lecturer, University of Computer Studies, Yangon, Faculty of Computer Systems and Technologies,
Shwe Pyi Thar Township, Yangon, 11411, Myanmar
hnincherry@ucsy.edu.mm
http://ucsy.edu.mm

Khaing Khaing Wai
Professor, University of Computer Studies, Yangon, Department of Information Technology Support and Maintenance, Shwe Pyi Thar Township, Yangon, 11411, Myanmar
khaingkhaingwai@ucsy.edu.mm
http://ucsy.edu.mm

Abstract

Many students find it challenging to comprehend the communication mechanism that is a part of TCP/IP because learning about it through textbooks and lectures is not enough. Network simulators are helpful resources for computer network education. Because the complex operations that occur in the computer network during data exchange between network devices are completed in incredibly tiny time slices, the learner is unable to follow and understand them. The main objective of our simulator is to display visually these processes. Our suggested system allows students to play with network activity to help them understand it. Using specific packets, step counts, and the network configuration setting as input, our approach calculates packet transmission step-by-step. Our method enables the students to comprehend network behavior. With our simulator, students may analyze network behavior in a variety of settings without requiring specialist gear.

***Keywords*: Network simulators; Communication mechanism; TCP/IP.**

1. Introduction

The development of networking abilities is essential for people in the current digital era. However, knowledge of network communication, packet sending and receiving between devices, device statuses, and network configuration settings are necessary to function as a network engineer. Network simulators provide sophisticated methods for understanding and instructing on the complex concepts of computer networks. They also make it easier to visualize and simulate networks of any topology.

Therefore, implementing a simulator that provides a way to solve difficulties linked to understanding computer networks would be very beneficial to students. This is because students can practice on their own at home and the teacher has an extra resource to use when teaching networking subjects. Simulators are the ideal tool for identifying concepts in computer design, even though network communication processes between devices are hard to understand without extensive study and experience.

A commercial simulator called OPNET Modeler [1] has more than 400 functionali-ties for modeling network protocols, network parts, and the dynamic behavior of the network. It supports a wide range of LAN and WAN networks, including ATM, IP, mobile networks, wireless LAN, etc.

Using the Java programming language, J-Sim [11] builds an autonomous architec-ture based on components (Autonomous Component Architecture, ACA). J-Sim pre-sents the idea of centralized management for independent simulation execution in-stances (background thread manager), enabling worldwide simulation time monitor-ing for all active instances.

NS-2 [4] is a discrete event network simulator that offers strong support for wiring and wireless network simulation of TCP, routing, and multicast protocols. The C++ programming language is used to develop simulator that provides users with access to a front-end API. Since the oTCL language is used for this front-end API, it is also used for the implementation of the network model and simulation control.

NS3 [9] simulator is developed in order to meet the demands of the expanding community of network researchers and developers. It also provides a tool for building highly customizable network simulations.

KivaNS [3] is an intuitively designed, free, open-source program. Java is used in the programming of KivaNS. The purpose is to simulate TCP-IP computer networks and to provide students with an independent virtual laboratory where they can do experiments related to IP routing without requiring expensive and complex physical equipment like routers, switches, hubs, PCs, etc.

NetSim [8] is also used in network lab research and experimentation to help the user learn the Cisco IOS command structure. A Windows-based program called NetSim imitates many different Cisco router models with many different protocols such as RIP, IGRP, EIGRP, BGP, and OSPF. It also supports several LAN/WAN protocols such as Frame Relay, ISDN, and PPP/CHAP.

The traditional approach to teaching network communication mechanisms consists primarily of lectures and texts, supplemented by tangible tools. It is challenging traditional teaching methods to enable the following:

- When students create a network, they communicate based on the timing and information included in the packets they want to send.
- Examine how the results vary depending on different inputs. Upon receiving each packet, the student verifies delivery; and
- Checks its internal status, looking at the MAC address table and ARP table.

To solve the problems, we propose a network simulator with the following features:

- The inputs are customized packets, network configurations, and the order in which they are transmitted;
- It delivers, receives, and sends packets by step-by-step computations.

2. Literature Review

One network simulator suggested by George F. Riley [10] is intended for use in classrooms. In particular, this applied to the graduate-level subject "CAD for Computer Networks." The lesson aims to accomplish two principal goals. Using one or more network simulation tools is the initial step. The second is to use these tools to observe how various network topologies—wired and wireless networks included—perform in various contexts. The learning experiences of students in network courses were helped by this study. Students need to comprehend the procedure of sending packets as part of our education.

Network simulators are utilized in networking courses, according to Qian Liu [5], to help students gain skills in customized learning. The goal of the study was to suggest a learner-centered environment in which students may choose and use the finest technologies to direct their own education at their own pace. With the help of this technique, students have more opportunities to investigate topics of interest and gain a detailed understanding of procedures and mechanisms. They can help students comprehend how networks behave and how to compare results in various contexts. Our study enables students to learn network communication with step-by-step, allowing them to work quickly.

Muhammad Wannous [12] used open source technologies for virtual network computing and virtualization in a networking virtual web-based laboratory. Students can use this system to create, configure, test, and debug a virtual network running on a host computer. As part of the evaluation procedure, their answers to a level exam were compared after the exercise was completed. This showed that the effectiveness of the system and the students' test scores both increased. Though it cannot help students grasp the packet delivery method, this research does let students practice and exercise computer networks.

According to the study [6], Packet Tracer could improve the practical networking activities that are used to teach computer-networking concepts. They show that while employing physical hardware with Packet Tracer does improve student performance, teachers must continue working with students to close certain significant gaps. Although this subject helps students build networks, it does not help them understand TCP/IP. This course teaches students how to send packets from one host to another without the use of physical devices by studying the operation of the network's communication mechanism in a virtual environment.

The paper [7] describes a web-based tool for modeling and visualizing computer networks. This system is implemented using the Java programming language, which enables the simulation and visualization of processes in computer networks with any topology. The purpose of this system is to help students grasp the principles of using a TCP/IP computer network. Due to the learner's inability to comprehend what is happening in a computer network when data exchange occurs in extremely short time slices between network devices. However, they are unable to pinpoint the exact packet that is being delivered between devices step-by-step. Our simulator aids to understand information in the packets between devices.

The study [2] developed and constructed the new education-based computer network simulator after assessing several popular network simulator kinds. With this simulator, students can conduct experiments on PCs that mimic real-world network equipment. It was created especially for certification tests, including the Cisco Certified Network Associate (CCNA), Cisco Certified Internetwork Expert (CCIE), and Cisco Certified Network Professional (CCNP). Students can get the relevant network course for the network experiment through this study. However, they cannot determine the precise packet transit between devices. With the help of our method, students will be able to comprehend packet information at all network connection stages.
Major headings should be typeset in boldface with the first letter of important words capitalized.

3. Methodology

This section describes how students to learn computer networks can use our system. With a visual representation of packet processing in devices, our simulator aids students in understanding the communication mechanism involved in sending packets over a network.

3.1. *Network communication mechanism*

Our system helps learners comprehend packet transport by using basic network properties like IP fragmentation, error checking, traffic management techniques, network size, topology, and communications protocols to organize traffic. For example, a device obtains the MAC address associated with the IP address from the ARP cache when an ICMP echo request is received. Students can understand why the ARP cache exists by seeing the actions. In order to capture packets' information sending between devices, the size of the network can be changed on the traffic control technique and network topology.

3.2. *Simulator's learning flow*

To understand the communication mechanism, take the following steps:

Step 1: Students set up a network configuration that consists of default gateways, IP addresses, MAC addresses, device links, and subnet masks. It also shows which packets are delivered at particular step values in the scenario. The packet parameters that are included in the scenario are the source IP address and destination IP address of an ICMP-Echo packet or the source IP address, source MAC address, destination IP address, and destination MAC address of an ARP packet. Along with the stages and types of packets that the devices communicate, it also includes the devices that send the packets.

Step 2: Our simulator simulates network communication (ICMP and ARP) for packet delivery using the network configuration and network scenario.

Step 3: Students use a visualizer to observe the network's behavior. The four main buttons (shown in Fig. 3) that are used to display the network in the visualizer are start, stop, next, and previous. Students can click the "start" button to open the network visualization. Additionally, they are able to observe that the network is continuously visualized from the scenario's start number to its finish number with each step. In the upper-left corner of the user interface, students may see how many steps there are currently. Students can access device information at every phase, including packet construction and protocol state, by touching the red square button. With a frame-by-frame representation of network states, including packet position, the visualizer generates animation. The packet icon that travels over the network in the animation is represented by the blue square. For instance, IP addresses, ethernet headers, source and destination hosts, and ICMP metadata may all be included in a packet transmitted by a device via ICMP. Users can instantly halt the animation by using the stop button; they can view the network state for the current step and the previous phase by using the next and previous buttons, respectively.

4. Implementation

4.1. *System flow*

The simulator's device controller, scenario interpreter, and step counter are depicted in Fig. 1.

It has mostly used two inputs, a network simulation scenario and a network configuration, to simulate network states. Device information, such as the quantity of devices, their ID, IP address, MAC address, subnet mask, default gateway, and cable connections between them, are included in the network setup file that is prepared. The components of a scenario are the IP addresses of the source and destination devices, the step at which packets are delivered, the devices transmitting them, the types of packets, and the end step number designating the end of the simulation.

The scenario interpreter requests packet transmission from the device controller based on the obtained step number of the step counter and the scenario. If the step count on the step counter and the step number in the scenario match, the scenario interpreter requests that the device controller send a packet. The scenario interpreter interprets the packets that will be transferred in the following step, while the step counter increases the step number until the scenario's final step number.

To govern packet forwarding, the device controller supports device attributes such as the cable connection between devices, device ID, IP address, MAC address, default gateway, and subnet mask. The packet data and network state are recorded in a log file once the network has reached a certain step number.

The step counter increases the step number one at a time until the scenario interpreter has completed interpreting. After a packet transmission at the scenario-corresponding phase is complete, the step number is increased by one.

Every communication step where packets are delivered and received is recorded in a log file, which records network circumstances. Every device object is printed out in the log file, which is saved in the ".dat" format, using the serialization technique.

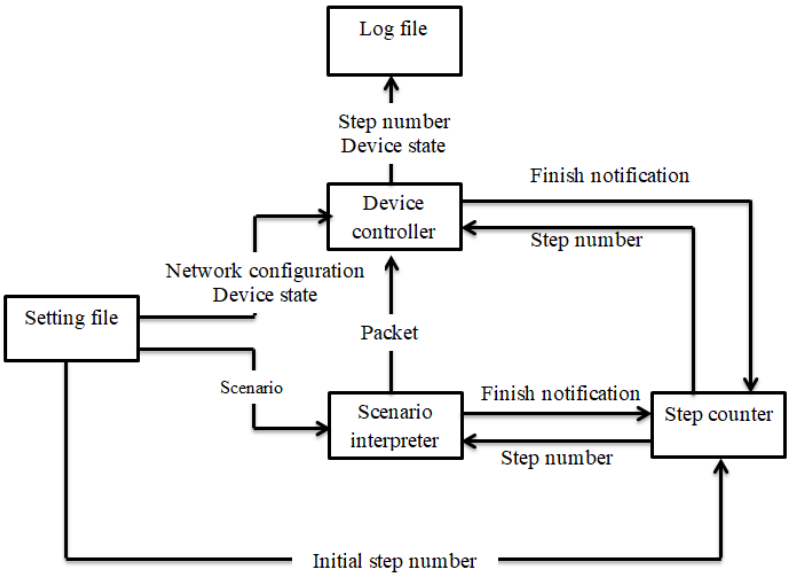
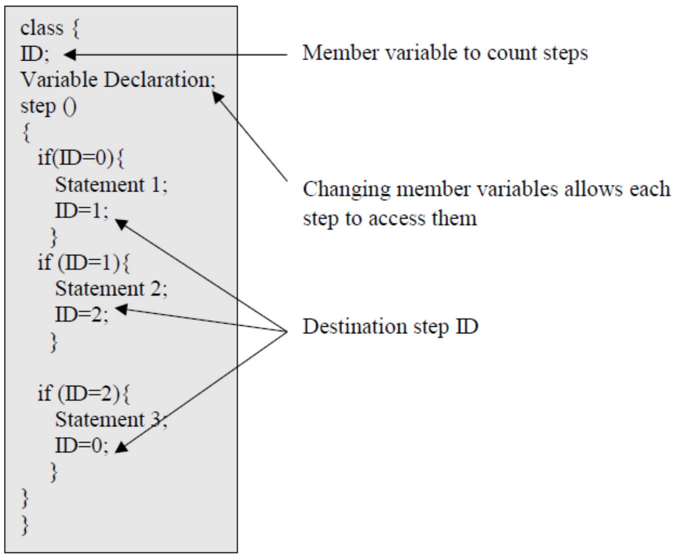


Fig. 1. System flow diagram for network simulator

4.2. *Step execution function*

We divided the packet processing processes across network cables and devices. Included is the procedure for going to the next phase at the conclusion of each phase. Each device's device controller completes the current step before moving on to the next one when the step counter has produced a step number. At each phase of execution, the network state is modified by one step.

Fig. 2 displays a sample model of step division. When the network communication technique is done, it can change based on the step number. A process's step number is increased for the subsequent step number when it is completed at that step number. ID is updated following the execution of Statement 1, Statement 2, or Statement 3 that correspond to ID using the step () function.



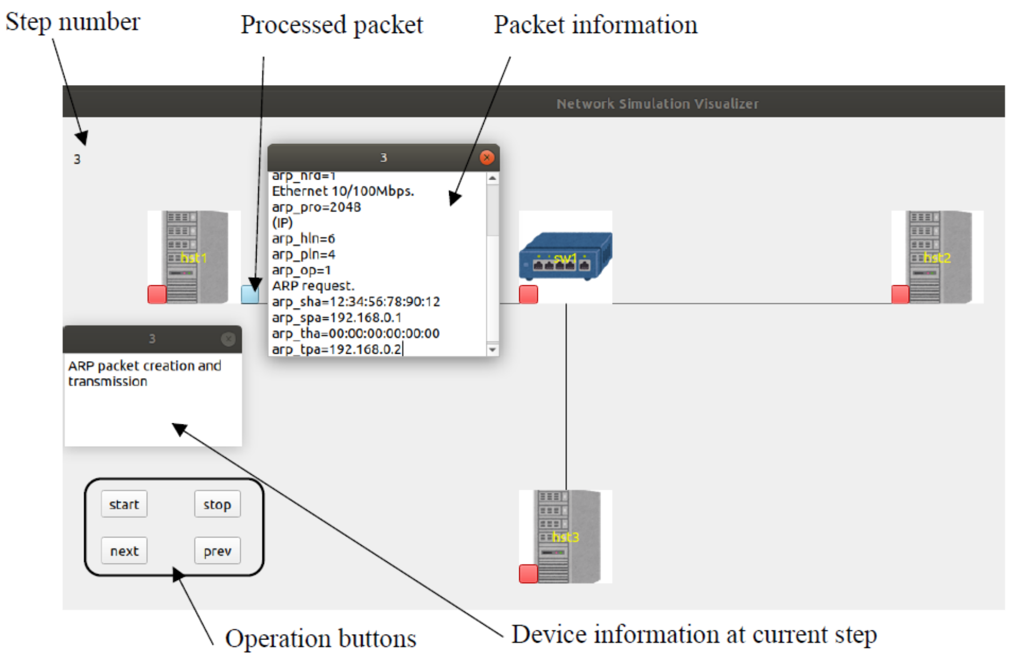
```
class {
ID;
Variable Declaration;
step ()
{
  if(ID=0){
    Statement 1;
    ID=1;
    }
  if (ID=1){
    Statement 2;
    ID=2;
    }

  if (ID=2){
    Statement 3;
    ID=0;
    }
}
}
```

Member variable to count steps
Changing member variables allows each step to access them
Destination step ID

Fig. 2. Step division model

4.3. *Network simulation visualizer*

We create a visualizer in Fig.3 that shows the current step number, the device description of the process, information about packet processing, and detailed instructions. Visualizer displays the network state animation in a step-by-step manner. Steps are raised on a regular interval when the visualizer loads the log file. Step number increases are stopped when the stop button is pressed. The next and previous buttons allow you to increase and decrease the step number with each step, respectively. The number of steps affects how network states are visualized.



```
arp_hrd=1
Ethernet 10/100Mbps.
arp_pro=2048
(IP)
arp_hln=6
arp_pln=4
arp_op=1
ARP request.
arp_sha=12:34:56:78:90:12
arp_spa=192.168.0.1
arp_tha=00:00:00:00:00:00
arp_tpa=192.168.0.2
```

ARP packet creation and transmission

Fig. 3. Network simulation visualizer

5. Experiment

For the network simulation experiment, two network topologies are set up, and the step numbers determine how the findings are analyzed. The simulator is implemented using the C++ programming language, and the visualizer is implemented using Qt. The research was conducted using an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz processor, 12GB RAM, and a VMware Workstation Player 16 virtual system.

Our simulator offers two distinct network configurations. Three hosts and one switch make up the basic configuration, as shown in Fig. 4. The configuration file and scenario file for this configuration are shown in Figs. 6 and 7. The first line in Fig. 6 lists the host, switching hub, and cable numbers. Lines 2-4 contain a host ID, IP address, MAC address, default gateway, and subnet mask. The switching hub ID is defined in the fifth line. Lines 6–8 include the following data: destination device ID, destination port number, source device ID, and source device ID. Lines 1-3 of Fig. 7 consist of a field data array, a protocol name, a source host ID, and a step number for packet transmission. The source and destination IP addresses in the array correspond to the ICMP protocol name. When using the ARP protocol, the array consists of the source, destination, and destination MAC addresses as IP addresses. The fourth line defines an end step number. The second configuration, with two switches and four hosts, is shown in Fig. 5. The configuration file and scenario file for this arrangement are shown in Figures 8 and 9.

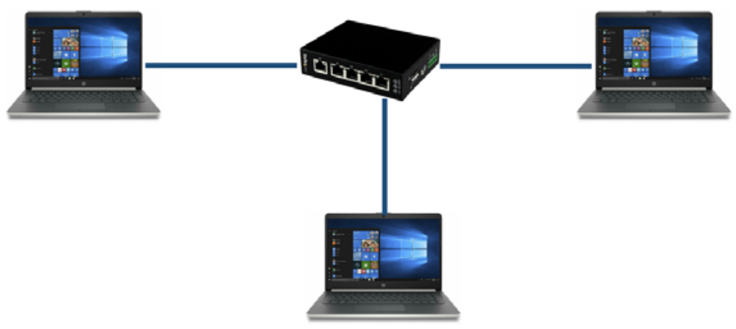


Fig. 4. A Network in which one switch is connected to three hosts

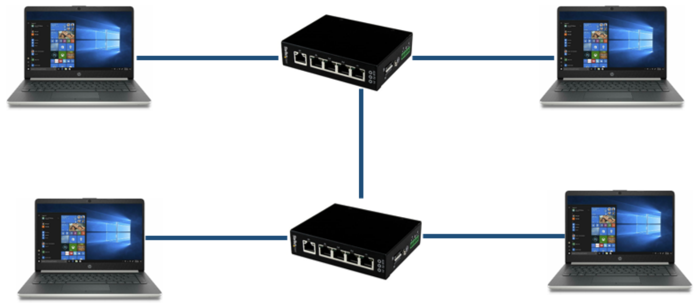
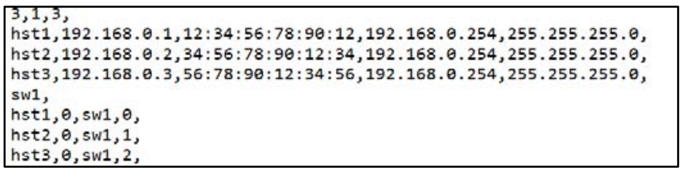
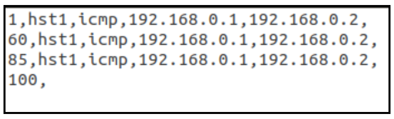


Fig. 5. Two switches linked to each two hosts that are connected together



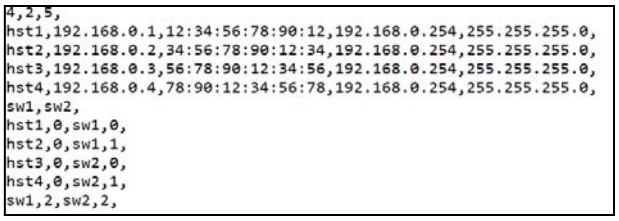
```
3,1,3,
hst1,192.168.0.1,12:34:56:78:90:12,192.168.0.254,255.255.255.0,
hst2,192.168.0.2,34:56:78:90:12:34,192.168.0.254,255.255.255.0,
hst3,192.168.0.3,56:78:90:12:34:56,192.168.0.254,255.255.255.0,
sw1,
hst1,0,sw1,0,
hst2,0,sw1,1,
hst3,0,sw1,2,
```

Fig. 6. Network configuration setting for Fig. 4



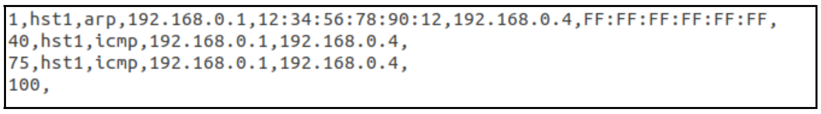
```
1,hst1,icmp,192.168.0.1,192.168.0.2,
60,hst1,icmp,192.168.0.1,192.168.0.2,
85,hst1,icmp,192.168.0.1,192.168.0.2,
100,
```

Fig. 7. Network scenario for Fig. 4



```
4,2,5,
hst1,192.168.0.1,12:34:56:78:90:12,192.168.0.254,255.255.255.0,
hst2,192.168.0.2,34:56:78:90:12:34,192.168.0.254,255.255.255.0,
hst3,192.168.0.3,56:78:90:12:34:56,192.168.0.254,255.255.255.0,
hst4,192.168.0.4,78:90:12:34:56:78,192.168.0.254,255.255.255.0,
sw1,sw2,
hst1,0,sw1,0,
hst2,0,sw1,1,
hst3,0,sw2,0,
hst4,0,sw2,1,
sw1,2,sw2,2,
```

Fig. 8. Network simulator configuration setting for Fig. 5



```
1,hst1,arp,192.168.0.1,12:34:56:78:90:12,192.168.0.4,FF:FF:FF:FF:FF:FF,
40,hst1,icmp,192.168.0.1,192.168.0.4,
75,hst1,icmp,192.168.0.1,192.168.0.4,
100,
```

Fig. 9. Network scenario for Fig. 5

5.1. *Experimental result*

Performance was assessed using the following metrics: resident set size (KB), CPU utilization (MB), log file size (MB), and execution time (s). The latter two were dependent on the total number of steps. Between 200 and 1000 steps, we tested with the two types of network topologies. The execution time and log file size increase linearly with the number of steps, as shown in the following Table 1.

There is minimal difference in the resident set size (RSS) when the number of steps is altered. The CPU use was measured with a different number of steps. The CPU use of our simulator does not change much as the number of steps rises. The number of devices employed in topology is independent of step numbers. The length of time it takes for each device to send a packet determines how many steps there are.

Total number of step	Log Size (MB)	CPU Usage (MB)	RSS (KB)	Execution Time (s)
200	20.1	114	11075	2.6
300	36.3	114	11043	3.2
800	98.5	114	11013	8.4
1000	121.0	114	11051	11.2

Table 1. Network simulation analysis result.

6. Future work

The simulator is equipped with a variety of network topologies and protocols, such as TCP and UDP. Furthermore, in order to interface with other devices, like routers, the current protocol implementations must be changed. Resume function may be implemented to make network simulation depends on changing network communication. Even if the performance analysis showed how scalable the simulator is, our goals remain to minimize resource usage and speed up execution times.

7. Conclusion

While students possess a high degree of practical skills, specialized network teaching laboratories are expensive. Alternatively, providing students with access to network simulators will be taken into consideration. Our simulator applies and replicates basic computer networking concepts. The network simulator is designed to be executed step-by-step communication over the network. At any time, the network can be restarted by reverting to its previous state. With a step number, it generates a log file to restart the network state and is easy to use and adaptive in simulation. The simulator is implemented in C++, while the visualizer is made with the Qt Library. It might make it possible for us to achieve our goals.

Conflict of Interest

Declared conflicts of interest do not exist for the authors. There are no financial conflicts to disclose, and all co-authors have reviewed and approved the manuscript's content. We attest that the submission is our original work and isn't currently being considered by another publisher.

Acknowledgments

I would like to thank Professor Khaing Khaing Wai, Department of Information Technology Support and Maintenance from University of Computer Studies, Yangon, Myanmar for supervising to complete and publish this work.

References

[1] "Introduction to Using OPNET Modeler", OPNETWORK 2002, Simulation and Modeling, SYSC 4005/5001.
[2] Cai W.X., Li G. S., Chen X. H., Hong C. Q., Zhu S. Z., Wu Q. H., Chen R.. Education Based New Computer Network Simulator Design and Implementation .The 11th International Conference on Computer Science & Education (ICCSE 2016) August 23-25, 2016. Nagoya University, Japan.
[3] Candelas F.A. and Gil P., "Practical experiments with KivaNS: virtual laboratory for simulating IP routing in Computer Networks," Research Reflections and Innovations in IICT in Education, vol 3, pp. 1415-1418, April 2009.
[4] Ciraci, Selim, and Bora Akyol, An evaluation of the network simulators in large-scale distributed simulations, Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid. ACM, 2011.
[5] Liu Q., "Applying simulators in computer networks education to encourage personalised learning", Global Journal of Engineering Education Volume 21, Number 2, 2019.
[6] Marquardson J.,Gomillion D. L., "Simulation for Network Education: Transferring Networking Skills Between Simulated to Physical Environments", Information Systems Education Journal (ISEDJ), February 2019.
[7] Nenad J., Zoran J.; Oliver P.; Ivan S.; Aleksandar Z., "Computer network simulation and visualization tool for educational purpose", 2013 11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS).
[8] NetSim 8.0 User Manual - Boson Software, 1998-2003, http://www.boson.com/files/support/
[9] NS-3 development team. Ns-3 network simulator. http://www.nsnam.org/.
[10] Riley G. F., "Using Network Simulation in Classroom Education", Proceedings of the 2012 Winter Simulation Conference.
[11] SOBEIH, Ahmed, et al. J-sim: A simulation environment for wireless sensor networks. In: Proceedings of the 38th annual Symposium on Simulation. IEEE Computer Society, 2005. p. 175-187.
[12] Wannous M., Student Member, IEEE, and H. Nakano . NVLab, a Networking Virtual Web-Based Laboratory that Implements Virtualization and Virtual Network Computing Technologies. IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, VOL. 3, NO. 2, APRIL-JUNE 2010.

Authors Profile

Hnin Cherry, received the M.C.Sc. degree from the University of Computer Studies, Pathein, Myanmar in 2010. She is currently working as a Lecturer in the University of Computer Studies, Pathein, Myanmar. Her other research interests are Computer Networking, Data Mining, Machine Learning and Deep Learning. She can be contacted at email: hnincherry@ucsy.edu.mm.

Khaing Khaing Wai, received the B.Sc. (Hons;), M.Sc. and M.Research. degrees in physics from Yangon University, Myanmar, in 1996, 1999 and 2000, respectively, and the Ph.D. degree in computer hardware technology from University of Computer Studies, Yangon, in 2005. She is currently Head of the Department of Information Technology Support and Maintenance at the University of Computer Studies, Yangon, Myanmar. She is also a Professor of cisco network lab in University of Computer Studies, Yangon. She can be contacted at email: khaingkhaingwai@ucsy.edu.mm.