NETWORK SIMULATION USING NCTUns

Ankit Verma*
Shashi Singh*
Meenakshi Vyas*

1. Introduction: Network simulator is software which is very helpful tool to develop, test, and diagnose any network protocol. To implement any kind of network with any kind of link bandwidth, propagations delay, routers etc we do not need to set up the actual network hence it is very economical and the results obtained are easier to analyze. A simulator needs to simulate various networking devices, application programs, network utility programs hence developing a simulator need great efforts.

To overcome some of the limitation of traditional simulators *S.Y. Wang* proposed a simulation methodology which has two desirable properties as follows.

- It uses the real-life UNIX TCP/IP protocol stack, real-life network application programs, and real-life network utility programs. To generate more accurate result similar to real life implementation..
- Second, it lets the system default UNIX POSIX API (i.e., the standard UNIX system call interface), be provided on every node in a simulated network. Any real-life UNIX
- application program, either existing or to be developed, thus can run normally on any node in a simulated network to generate traffic.[1]

2. The NCTUns Simulator: The NCTUns network simulator and emulator (NCTUns), is a high-fidelity and extensible network simulator capable of simulating various devices and protocols used in both wired and wireless networks. The predecessor of NCTUns is the Harvard network simulator, which Wang authored in 1999.To overcome drawbacks of this simulator, after joining National Chiao Tung University (NCTU),, Taiwan in February 2000, Prof. S.Y. Wang lead his students to develop NCTUns since then. It supports remote simulations and concurrent simulations. It uses an open-system architecture to enable protocol modules to be easily added to the simulator. In addition, it has a highly-integrated GUI environment for editing a network topology, specifying network traffic, plotting performance curves, configuring the protocol stack used inside a network node, and playing back animations of logged packet transfers. NCTUns can generate high-fidelity simulation results at high speeds when the network traffic load is not heavy. NCTUns was first released to the networking community on November 1, 2002. Its web site is set up at http://NSL.csie.nctu.edu.tw/nctuns.html. As of January 12**,** 2010 Initially, NCTUns was developed for the FreeBSD operating system. As the Linux operating system is getting popular, NCTUns now only supports the Linux operating system. Specifically, the version of Linux distribution that NCTUns 6.0 currently supports is Red Hat ‘s Fedora 1 2 with kernel version 2.6.31.6. [2].

3. Features

- It can be easily used as an emulator.
- It supports distributed emulation of a large network over multiple machines

*Maharaja Ranjit Singh College of Professional Sciences, Indore, India

- It directly uses the real-life Linux TCP/IP protocol stack to generate high-fidelity simulation results.
- It can run any real-life UNIX application program on a simulated node without any modification.
- It supports remote and concurrent simulations.
- It provides complete and high-quality documentation.
  It is continuously supported, maintained, and improved [3].

4. Applications:

- As a network-planning tool.
- As a research tool.
- As an application program performance evaluation tool.
- As an education tool [4].

5. Simulation Methodology: NCTUns uses an innovative kernel re-entering methodology to perform network simulations.

Kernel re-entering simulation methodology

It uses the existing real-world Linux protocol stack to generate high-fidelity TCP/IP network simulation results. In the case of Figure 1 the packet sent by the TCP sender passes through the kernel two times. This is the property of the kernel re-entering simulation methodology. By re-entering the kernel multiple times, we can create an illusion that a packet passes through several different hosts (i.e., the packet thinks that it passes through several different TCP/IP protocol stack), Actually, the packet is always in the same machine and passes through the same TCP/IP protocol stack.

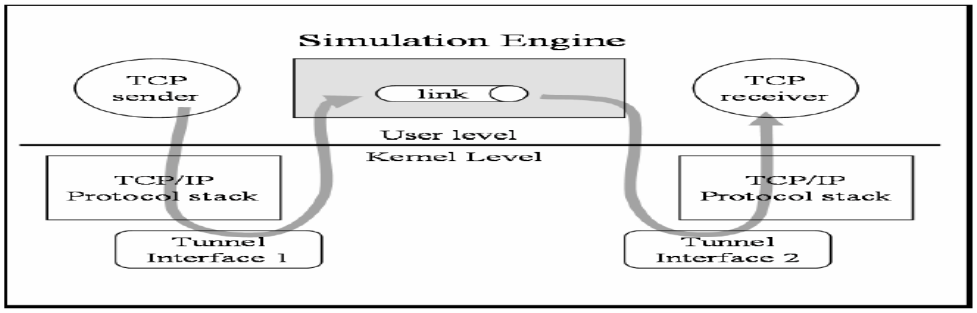

Fig.1 Kernel re-entering simulation methodology

NCTUns uses a pseudo network device driver (called the tunnel device driver in Linux), to virtualized the function of a Network Interface Card (NIC),, which is a set of standard driver functions, such as open(),, read(),, write(),, close(),, etc., that do not perform any real packet transmission/reception operations and are not associated with any real NIC. A pseudo network device driver contains a packet output queue, which is used by NCTUns to temporarily store packets to be transmitted. By properly configuring the IP address and routing entries associated with a pseudo network device driver, NCTUns makes the Linux kernel think that a pseudo network device driver created by NCTUns is a real NIC device driver that controls a real NIC.

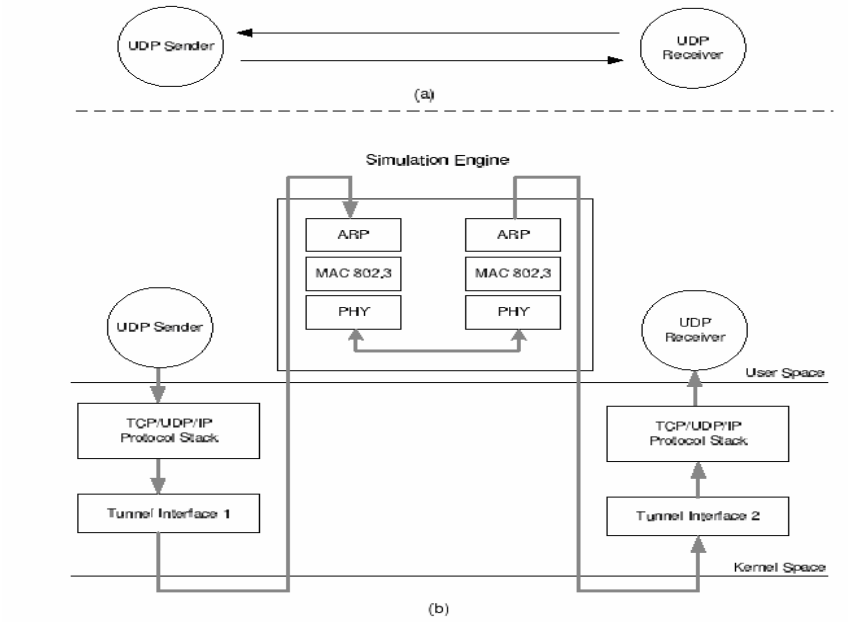

Fig.2 Protocol Stack

Consider a simple network shown in Fig. 2(a),, where a UDP sender program wants to transmit data to a UDP receiver program. These 2 programs run on 2 different machines, connected via a wired link. As shown in Fig. 2(b),, before starting the simulation, NCTUns first creates two tunnel device drivers to represent the NICs of the sending node and the receiving node, respectively. It then sets up the IP addresses of these two tunnel devices. Suppose that the IP address of tunnel device 1 (sender), is set to 1.0.1.10 and that of tunnel device 2 (receiver), is set to 1.0.1.20, respectively. The next step is to add proper routing entries into the kernel routing table so that the Linux kernel will in queue packets that are originated from 1.0.1.10 and destined to 1.0.1.20 into the output queue of tunnel device1 for transmission.

After being forked, the UDP receiver process first uses standard socket APIs, such as socket(), and bind(),, to create a socket structure that is associated with the IP address of tunnel device 2 (i.e., 1.0.1.20), It then uses the recvfrom(), call to wait for packets destined to the IP address 1.0.1.20. On the other hand, after being forked, the UDP sender process first uses the socket(), call to create a socket structure and then uses the sendto(), call to write a segment of data into the socket send buffer, which is in the kernel space. The written data segment will then pass through (be processed by), the UDP/IP protocol stack and be encapsulated into an IP packet, which will then be placed in the output queue of tunnel device 1.Later on, the simulation engine reads these packets out from the output queue of tunnel device 1 and simulates the processing of the ARP, MAC-layer, and physical-layer protocols on node 1. After finding the MAC address for the IP address of the packet's destination node, the ARP module fills out the Ethernet header of the packet and sends it down to the MAC module. Such a sending down process repeats until the packet reaches the PHY module, which simulates the transmission/reception behavior of a packet over a wired link. Finally, the PHY module of node 1 delivers the packet to the PHY module of node 2.

Upon receiving the packet, the PHY module of node 2 first simulates the reception of this packet. If it receives any other packet during the reception of this packet, it drops these two packets

because they are collided with each other. Otherwise, after the reception of this packet has elapsed, the node 2's PHY module delivers it up to the MAC module, which performs the MAC-layer processing for this packet. Similar to the sending down process on the transmitting node, such a sending up process on the receiving node repeats until the received packet reaches the ARP module, where the simulation engine writes the packet into tunnel device 2. After this operation is performed, the Linux kernel invokes the IP-layer receiving routines to process this packet as if this packet was received from a real NIC. Following the normal processing for an incoming IP packet, this packet is then dispatched to the UDP layer. The UDP-layer receiving routines then enquire this packet into the socket-layer receive buffer (which stores the packets destined to the socket created by the UDP receiver process), Upon detecting the arrival of this packet at the socket receive buffer, the Linux kernel then wakes up the UDP receiver process, which then copies the data carried by this packet to its own memory space (i.e., return from the recvfrom(), call),[5].

6. Components and Architecture of NCTUns: NCTUns adopts a distributed architecture. It is a system comprising eight components.
First - A GUI program by which a user edits a network topology, configures the protocol modules used inside a network node, specifies mobile nodes' initial location and moving paths, plots performance graphs, plays back the animation of a packet transfer trace, etc.
Second - The simulation engine program, which provides basic and useful simulation services (e.g., event scheduling, timer management, and packet manipulation, etc.), to protocol modules
Third - The set of various protocol modules, each of which implements a specific protocol or function (e.g., packet scheduling or buffer management),
Fourth - The simulation job dispatcher program that can simultaneously manage and use multiple simulation servers to increase the aggregate simulation throughput.
Fifth - Coordinator program. It registers itself with the dispatcher to join in the dispatcher's simulation server farm. Later on, when the status (idle or busy), of the simulation server changes, it will notify the dispatcher of the new status. This enables the dispatcher to choose an available simulation server from its simulation server farm to service a job.
Sixth - The kernel patches that need to be made to the kernel source code so that a simulation engine process can run on a UNIX machine correctly.
Seventh - Various real-life user-level application programs.
Eighth - Various user-level daemons that are run up for the whole simulation case for a simulated network can be constructed automatically..[1,5,6,7]
The following figure shows the distributed architecture of the NCTUns:

7. Simulation Modes: According to users' common needs, it groups the operations of generating a simulation/emulation case into four modes.

a), The Draw Topology mode: In this mode, one can insert network nodes, create network links, and specify the locations and moving paths of mobile nodes. In addition, the GUI program provides a complete tool kit for users to construct road networks, which is fundamental to wireless vehicular network simulations, where many P2P researchers are proposing to run P2P applications.

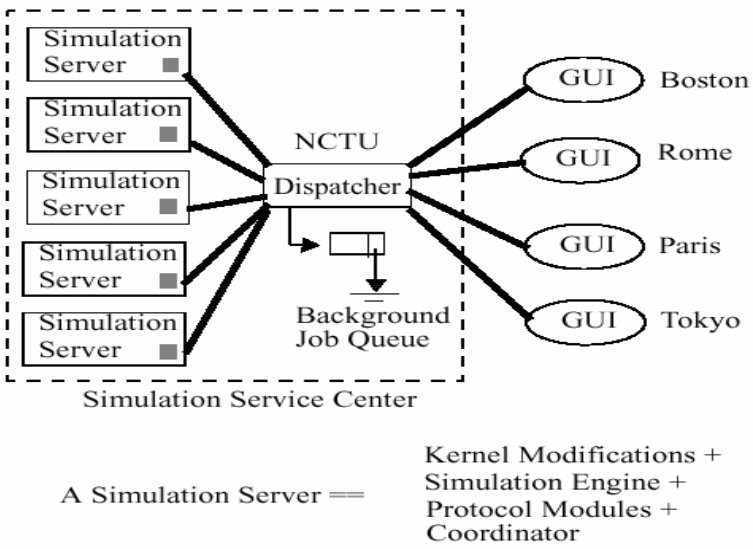

Fig.3 Distributed Architecture of NCTUns

b), The Edit Property mode: In this mode, one can double-click the icon of a network node to configure its properties (e.g., the network protocol stack used in this node, the applications to be run on this node during simulation, and other parameters),

c), The Run Simulation mode: In this mode, the GUI program provides users with a complete set of commands to **start/pause/stop** a simulation. One can easily control the progress of a simulation by simply pressing a button on the GUI control panel.

d), The Play Back mode: After a simulation is finished, the GUI program will automatically switch itself into the Play Back mode and read the packet trace file generated during the simulation. In this mode, one can use the GUI program to replay a node's packet transmission/reception operations in an animated manner.[5].

8. Generate Infrastructure Mobile Nodes' IP and MAC Address: It's easy to build a network topology using Topology Editor. Suppose we would like to establish a simulation with infrastructure wireless network. After building the network topology, next step is to generate mobile nodes' IP and MAC address. Following is the procedure to assign mobile nodes' IP:
1.Switch the operating mode from Draw Topology to Edit Property.
2.Select the mobile nodes and AP which are in the same subnet. (figure 4),
3. Menu->Tools->WLAN Mobile Nodes->Generate Infrastructure Mobile Nodes' IP and Mac Address(pic.2),
4. Set the subnet ID and Gateway IP. (pic.3),

Note: the total number of selected nodes must be less than 255 because a subnet has at most 254 usable IP addresses (X.X.X.255 is used for broadcasting), [8].

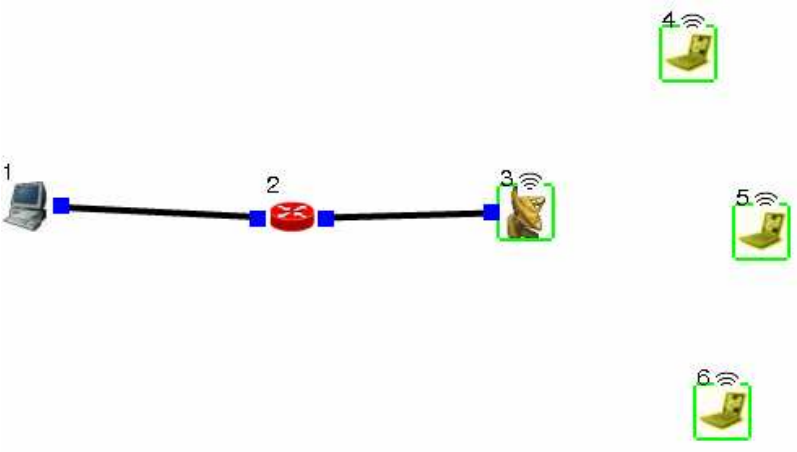

Fig. 4:Example Network

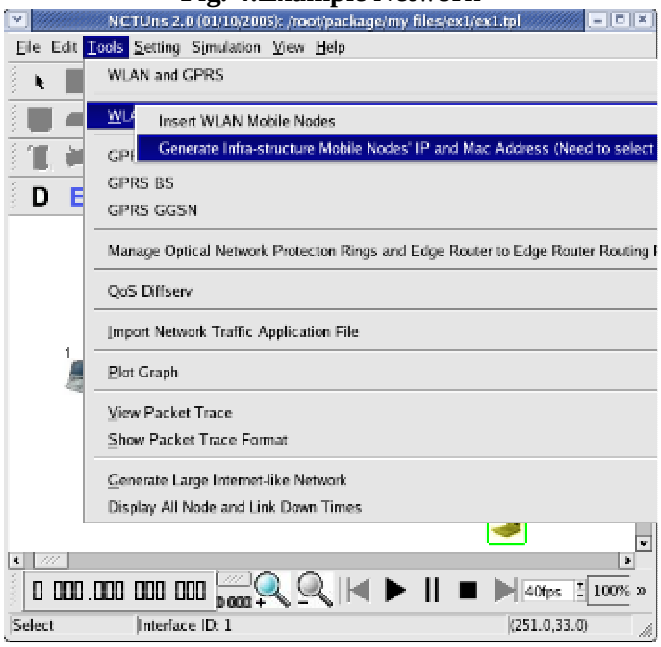

Fig.5 Creating the network

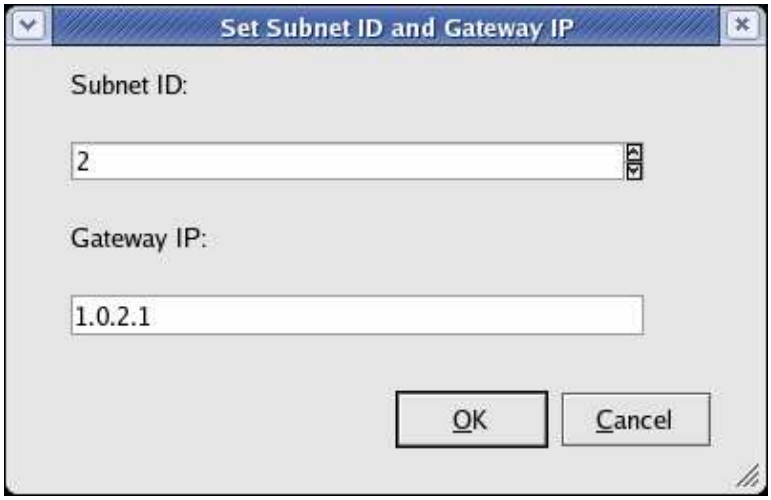

Fig. 6 Setting subnet id and gateway

9. Conclusion: NCTUns is a GUI based network simulator with lot of ease in designing and simulation network no doubt it is user friendly and provide lot of advantages like conviviality of interface. GUI environment for editing a network topology, specifying network traffic, plotting performance curves, configuring the protocol stack used inside a network node, and playing back animations of logged packet transfers .It can automatically generate all related document for simulation like graph routing table etc. .But the result obtained by this simulator vary as compare to the real life result it also sometime generates false result. It is not scalable but despite of this disadvantages. It is much popular among the researches. As the researches going on may its limitation can be overcome by making some changes in simulator design as it allow to add protocol module.

References

1. http://www.csie.nctu.edu.tw/~shieyuan/publications/NCTUNSDesign.pdf.
2. http://nsl10.csie.nctu.edu.tw/support/documentation/NCTUns4.0_ProtocolDeveloperManual_07142007_final.pdf.
3. http://nsl.csie.nctu.edu.tw/nctuns.html.
4. http://nsl10.csie.nctu.edu.tw/
5. S.Y. Wang and R.M. Huang, NCTUns Tool for Innovative Network Emulations, a chapter of the Computer-Aided Design and Other Computing Research Developments book, (ISBN: 978-1-60456-860-8, published by Nova Science Publishers in 2009),
6. S.Y. Wang, C.C. Lin, and C.C. Huang, NCTUns Tool for Evaluating the Performances of Real-life P2P Applications, a chapter of the Peer-to-Peer Networks, Security, Protocols, and Applications book, (to be published by Nova Science Publishers in 2009),
7. S.Y. Wang and R.M. Huang, NCTUns Tool for Innovative Network Emulations, a chapter of the Computer-Aided Design and Other Computing Research Developments book, (ISBN: 978-1-60456-860-8, published by Nova Science Publishers in 2009),
8. http://blog.roodo.com/crusor/archives/cat_91719.html.