Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

NETWORK SIMULATIONS WITH OPNET

Xinjie Chang

Network Technology Research Center
School of EEE
Nanyang Technological University
SINGAPORE 639798

ABSTRACT

Several computer network simulators are compared. One of the most powerful software simulation package-OPNET is introduced in detail. The implementation details of the network models in OPNET are given. Some simulation examples are also illustrated.

1 NETWORK SIMULATION

Simulation Modeling is becoming an increasingly popular method for network performance analysis. Generally, there are two forms of network simulation: analytical modeling and computer simulation. The first is by mathematical analysis that characterizes a network as a set of equations. The main disadvantage is its over simplistic view of the network and inability to simulate the dynamic nature of a network. Thus, the study of a complex system always requires a discrete event simulation package, which can compute the time that would be associated with real events in a real-life situation. Software simulator is a valuable tool especially for today's network with complex architectures and topologies. Designers can test their new ideas and carry out performance related studies, therefore freed from the burden of the "trial and error" hardware implementations.

A typical network simulator can provide the programmer with the abstraction of multiple threads of control and inter-thread communication. Functions and protocols are described either by finite-state machine, native programming code, or a combination of the two. A simulator typically comes with a set of predefined modules and user-friendly GUI. Some network simulators even provide extensive support for visualization and animation. There are also emulator such as the NIST Network Emulation Tool (NIST Net). By operating at the IP level, it can emulate the critical end-to-end performance characteristics imposed by various wide area network situations or by various underlying subnetwork technologies in a lab test-bed environment (NIST NET Homepage).

Some examples of academic simulators include:

REAL: REAL is a simulator for studying the dynamic behavior of flow and congestion control schemes in packet switch data networks. Network topology, protocols, data and control parameters are represented by *Scenario*, which are described using NetLanguage, a simple ASCII representation of the network. About 30 modules are provided which can exactly emulate the actions of several well-known flow control protocols (S. Keshav 1997).

INSANE: INSANE is a network simulator designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements. It's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queueing. A protocol similar to the Real-Time Channel Administration Protocol (RCAP) is implemented for ATM signalling. A Tk-based graphical simulation monitor can provide an easy way to check the progress of multiple running simulation processes (INSANE Homepage).

NetSim: NetSim is intended to offer a very detailed simulation of Ethernet, including realistic modeling of signal propagation, the effect of the relative positions of stations on events on the network, the collision detection and handling process and the transmission deferral mechanism. But it cannot be extended to address modern networks (Lewis, Barnett 1993).

Maisie: Maisie is a C-based language for hierarchical simulation (L. Bagrodia 1991), or more specifically, a language for parallel discrete event simulation. A logical process is used to model one or more physical processes; the events in the physical system are modeled by message exchanges among the corresponding logical processes in the model. User can also migrate into recent extension:

Parsec and MOOSE (an object-orient extension) (Rajive Bagrodia 1995).

Other examples also include ns-2 (ns Network simulator), VINT (VINT homepage), U-Net (T. Von. Eicken, et.al. 1995), USC TCP-Vegas test-bed (J. s.Ahn, et.al. 1995), and Harvard simulator (S. Y.Wang, H.T.Kung 1999). As to commercial simulator, examples include BONeS (Cadence Inc.), COMNET III (CACI) and OPNET (MIL3). BONeS DESIGNER provides lots of building blocks, modeling capabilities, and analysis tools for development and analysis of network products, protocols, and system architectures. With its recent released ATM Verification Environment (AVE), it is specifically targeted for ATM architectural exploration and hardware sizing. COMNET III, a graphical, off-the-shelf package, lets you quickly and easily analyzes and predicts the performance of networks ranging from simple LANs to complex enterprise-wide systems (CACI). Starting with a library of network objects with one COMNET III object representing real world objects, The COMNET III 's object-oriented framework and GUI gives user the flexibility to try an unlimited number of "what if" scenarios.

For maximum effectiveness, a simulation environment should be modular, hierarchical, and take advantage of the graphical capabilities of today's workstations. OPNET (MIL3) is an object-oriented simulation environment that meets all these requirements and is the most powerful general-purpose network simulator. OPNET's comprehensive analysis tool is special ideal for interpreting and synthesizing output data. A discrete-event simulation of the call and routing signaling was developed using a number of OPNET's unique features such as the dynamic allocation of processes to model virtual circuits transiting through an ATM switch. Moreover, its built-in Proto-C language support provides it the ability to realize almost any function and protocol. So that, in the following sections, the software simulation package, OPNET, is discussed.

2 OPNET SIMULATOR

OPNET(Optimized Network Engineering Tool) provides a comprehensive development environment for the specification, simulation and performance analysis of communication networks. A large range of communication systems from a single LAN to global satellite networks can be supported. Discrete event simulations are used as the means of analyzing system performance and their behavior. The key features of OPNET are summarized here as:

- **Modeling and Simulation Cycle** OPNET provides powerful tools to assist user to go through three out of the five phases in a design circle(i.e. the building of models, the execution of a simulation and the analysis of the output data), see Figure 1.
- **Hierarchical Modeling** OPNET employs a hierarchical structure to modeling. Each level of the hierarchy describes different aspects of the complete model being simulated.
- **Specialized in communication networks** Detailed library models provide support for existing protocols and allow researchers and developers to either modify these existing models or develop new models of their own.
- **Automatic simulation generation** OPNET models can be compiled into executable code. An executable discrete-event simulation can be debugged or simply executed, resulting in output data.

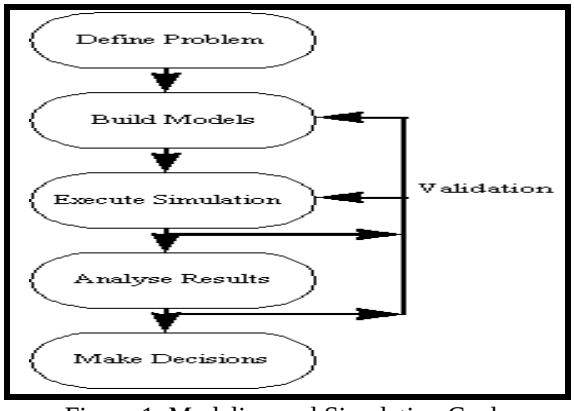


Figure 1: Modeling and Simulation Cycle

This sophisticated package comes complete with a range of tools which allows developers specify models in great detail, identify the elements of the model of interest, execute the simulation and analyze the generated output data:

- Hierarchical Model Building
  - Network Editor - network topology models
  - Node Editor - data flow models define
  - Process Editor - control flow models
- Running Simulations
  - Simulation Tool - define and run simulation
  - Debugging Tool - interact with running simulations
- Analyzing Results
  - Probe Editor –data need to be collected
  - Analysis Tool – statistical results
  - Filter Tool – date processing
  - Animation Viewer – dynamic behavior

2.1 Hierarchical Modeling

OPNET provides four tools called editors to develop a representation of a system being modeled. These editors, the Network, Node, Process and Parameter Editors, are organized in a hierarchical fashion, which supports the concept of model level reuse. Models developed at one layer can be used by another model at a higher layer. Figure 2 portrays this hierarchical organization. The following sections introduce each of the modeling domains. The Parameter Editor is always seen as a utility editor, and not considered a modeling domain.

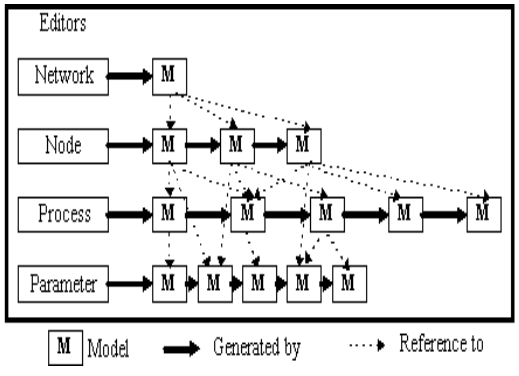


Figure 2: Hierarchical Organization of Editors

2.1.1 Network Model

Network Editor is used to specify the physical topology of a communications network, which define the position and interconnection of communicating entities, i.e., *node* and *link*. The specific capabilities of each node are realized in the underlying *model*. A set of parameters or characteristics is attached with each model that can be set to customize the node's behavior. A node can either be fixed, mobile or satellite. Simplex (unidirectional) or duplex (bi-directional) point-to-point links connects pairs of nodes. A *bus link* provides a broadcast medium for an arbitrary number of attached devices. Mobile communication is supported by *radio links*. Links can also be customized to simulate the actual communication channels.

The complexity of a network model would be unmanageable where numerous networks were being modeled as part of a single system. This complexity is eliminated by an abstraction known as a *subnetwork*. A subnetwork may contain many subnetworks, at the lowest level, a subnetwork is composed only of nodes and links Communications links facilitate communication between subnetworks.

2.1.2 Node Model

Communication devices created and interconnected at the network level need to be specified in the node domain using the Node Editor. Node models are expressed as interconnected *modules*. These modules can be grouped into two distinct categories. The first set is modules that have predefined characteristics and a set of built-in parameters. Examples are packet generators, point-to-point transmitters and radio receivers. The second group contains highly programmable modules. These modules referred to as *processors* and *queues*, rely on process model specifications.

Each node is described by a block structured data flow diagram. Each programmable block in a Node Model has its functionality defined by a Process Model. Modules are interconnected by either *packet streams* or *statistic wires*. Packets are transferred between modules using packet streams. Statistic wires could be used to convey numeric signals.

2.1.3 Process Model

Process models, created using the process editor, are used to describe the logic flow and behavior of processor and queue modules. Communication between process is supported by *interrupts*. Process models are expressed in a language called Proto-C, which consists of state transition diagrams(STDs), a library of kernel procedures, and the standard C programming language. The OPNET Process Editor uses a powerful state-transition diagram approach to support specification of any type of protocol, resource, application, algorithm, or queueing policy. States and transitions graphically define the progression of a process in response to events. Within each state, general logic can be specified using a library of predefined functions and even the full flexibility of the C language. Process may create new processes(*child process*) to perform sub-tasks and thus is called the *parent process.*

2.2 Running Simulation

2.2.1 Simulation Editor

After defining all the models of the network system, we can exercise them in a dynamic simulation in order to study system performance and behavior. Generally, there are three steps for simulations executation and information collection:

1. Specifying Data Collection: Model developers always need to decide which information should be extracted from the simulation, such as application-specific

statistics, behavioral characterizations, and sometimes application-specific visualization. These can take on several different forms including visual animations, time-dependent series of values(vector), and parametric relationships(scalar).

2. Simulation Construction: OPNET simulations are obtained by executing a simulation program, which is an executable file in the host computer's filesystem.
3. Simulation Execution: Simulation execution is the final step in an "iteration" of a modeling experiment. In general, based on the results observed during this step, changes are made to the model's specification or to the probes, and additional simulations are executed. OPNET provides a number of options for running simulations, including internal and external execution, and the ability to configure attributes that affect the simulation's behavior. This section introduces concepts, techniques, and features that support simulation execution.

OPNET simulations can be run independently from the OPNET graphical tool by using the op_runsim utility program. However, you can also run simulations from the Simulation Tool within OPNET, which offers the convenience of a graphical interface. The Simulation Tool provides the following services: 1) specification of simulation sequences consisting of an ordered list of simulations and associated attribute values 2) execution of simulation sequences 3) storage of simulation sequences in files for later use.

2.3 Data Generation

2.3.1 Probe Editor

Most OPNET models that contain objects that are capable of generating vast amounts of output data during simulations. The sources of output data include pre-defined and user-defined statistics, automatic animation, and custom-programmed animation. Users can use Probe Editor to specify which data to collect. A probe is defined for each source of data that the user wishes to enable. Probes are grouped into a probe list which, allowing them to be collectively applied to a model when a simulation is executed. Several different probe types are provided by OPNET in order to capture different types of output data. These are:

- **Statistic Probe** This type of probe can be applied to predefined, standard statistics monitoring characteristics such as bit error rates or throughput.
- **Automatic Animation Probe** This type of probe is used to generate animation sequences for a simulation.
- **Custom Animation Probe** Process and link models also support the creation of custom animations. The actual specification of the animation's characteristics is defined within the user's code.
- **Coupled Statistic Probe** This type of probes generates output data as the statistic probe does but, in addition, a *primary module* and a *coupled module* need to be defined. Some statistical data is generated at the primary module. This data is only generated when changes to the statistic are due to interactions with the coupled module. This type of probe is only used for radio receiver.

2.3.2 Analysis Tool

Simulations can be used to generate a number of different forms of output, as described above. These forms include several types of numerical data, animation, and detailed traces provided by the OPNET debugger. In addition, because OPNET simulations support open interfaces to the C language, and the host computer's operating system, simulation developers may generate proprietary forms of output ranging from messages printed in the console window, to generation of ASCII or binary files, and even live interactions with other programs. However, the most commonly used forms of output data are those that are directly supported by Simulation Kernel interfaces for collection, and by existing tools for viewing and analysis. Both animation data and numerical statistics fall into this category. Animation data is generated either by using automatic animation probes or by developing custom animations with the KP's of the Simulation Ker-nel's Anim package; the m3_vuanim utility is then used to view the animations. Sim-ilarly, statistic data is generated by setting statistic probes, and/or by the KP's of the Kernel's Stat package; OPNET's Analysis Tool can then be used to view and manip-ulate the statistical data.

The service provided by the Analysis Tool is to display information in the form of graphs. Graphs are presented within rectangular areas called analysis panels. A number of different operations can be used to create analysis panels, all of which have as their basic purpose to display a new set of data, or to transform an existing one. An analysis panel consists of a plotting area, with two numbered axes, generally referred to as the abscissa axis (horizontal), and the ordinate axis (vertical). The plotting

area can contain one or more graphs describing relationships between vari-ables mapped to the two axes. For example, the graph in the panel below shows how the size of a queue varies as a function of time.

2.3.3 Filter Tool

OPNET's Analysis Tool allows the user to extract data from simulation output files and to display this data in various forms, as described in Chapter Datan of the OPNET Modeling Manual. The Analysis Tool also supports several mechanisms for numerically processing the data and generating new data sets that can also be plotted. These include computing probability density functions and cumulative distribution functions, as well as generating histograms. The data presented in the Analysis Tool may also be operated on by numeric filters. These are constructed from a pre-defined set of filter elements in the Filter Editor.

Filter models are represented as block diagrams consisting of interconnected filter elements. Filter elements may be either built-in numeric processing elements, or references to other filter models. Thus, filter models are hierarchical, in that they may be composed of other filter models. However, all filter models must be composed at the lowest level of pre-defined filters discussed in Chapter Datan of the OPNET Modeling Manual.

Filters operate on vectors. Vectors are discrete and ordered sets of numeric data which consist of entries, as discussed in Chapter Datan of the OPNET Modeling Manual. Each entry consists of an abscissa and an ordinate value. These are double-precision floating point numbers. A filter model may operate on one or more vectors and combine them to form its output, which must consist of just one vector. The vectors that are fed into the filter are called input vectors; the result of the filter‘s processing is called the filter‘s output vector.

3 SIMULATION EXAMPLE

First, we give out a queueing network example as shown in Figure 3. There are three FIFO queues in tandem and each has several homogeneous sources. We can specify build in parameters for the source and queue, respectively. The source will generate constant length packets in a Poisson manner with a given rate. The buffer capacity and service rate for each queue can also be specified. By changing the buffer capacity and service rate, some simulation results are shown in Figure 4 to Figure 7. Figure 4 and Figure 5 give the end-to-end delay and loss ration against different buffer capacity values. The end-to-end delay and loss ratio against different service rate values are given out in Figure 6 and Figure 7, respectively.

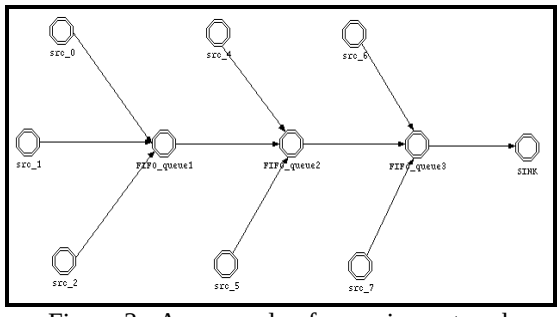


Figure 3: An example of queueing network

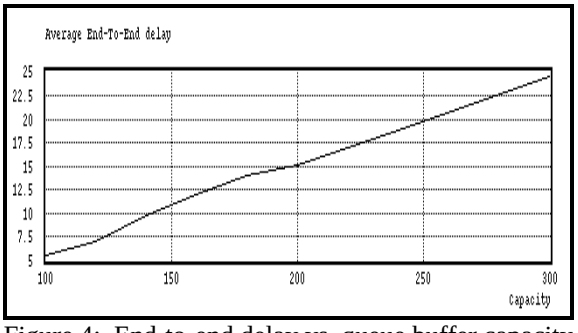


Figure 4: End-to-end delay vs. queue buffer capacity

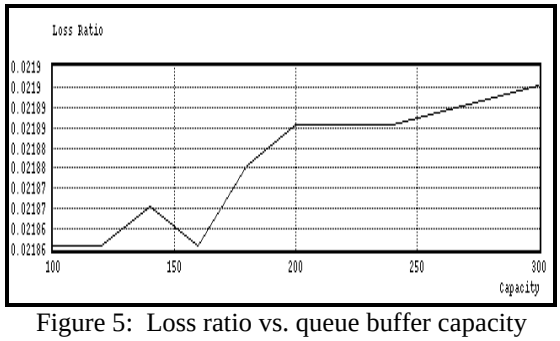


Figure 5: Loss ratio vs. queue buffer capacity

Another example is modeling a scenario of several Ethernet subnets connected by an ATM network backbone. The network topology is shown in Figure 8. Figure 9 gives the internal structure of one of the subnets, which has 20 nodes modeling Ethernet workstation(“node_0” to “node_19”), a Hub and gateway to connect the subnet with one of the switches of the ATM backbone.

On each Ethernet workstation we can specify the type of applications and their correspondent parameters. Here we only use the Email and FTP applications. Parts of simulation results are given in Figure 11 to Figure 14.

4 CONCLUSION

In this article, several computer network simulators has been introduced. The software simulation package, OPNET, which specializes in discrete-event simulation of communication systems, has been presented in detail. The implementation details of the models developed are also given.

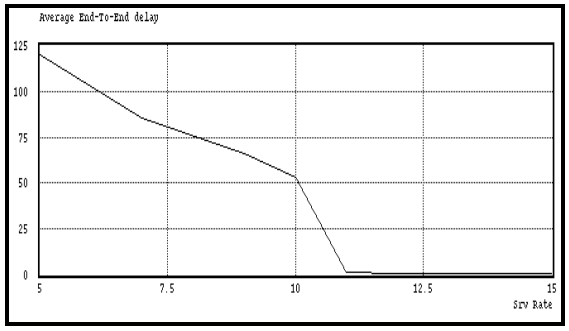


Figure 6: End-to-end delay vs. queue service rate

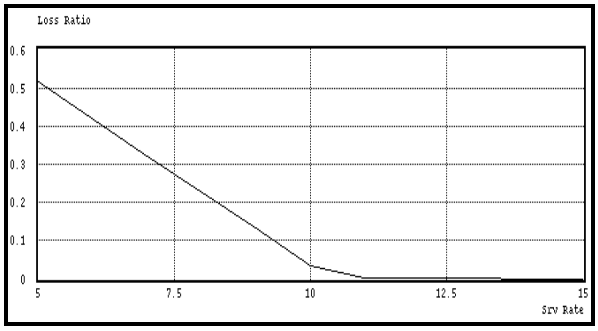


Figure 7: Loss Ratio vs. queue service rate

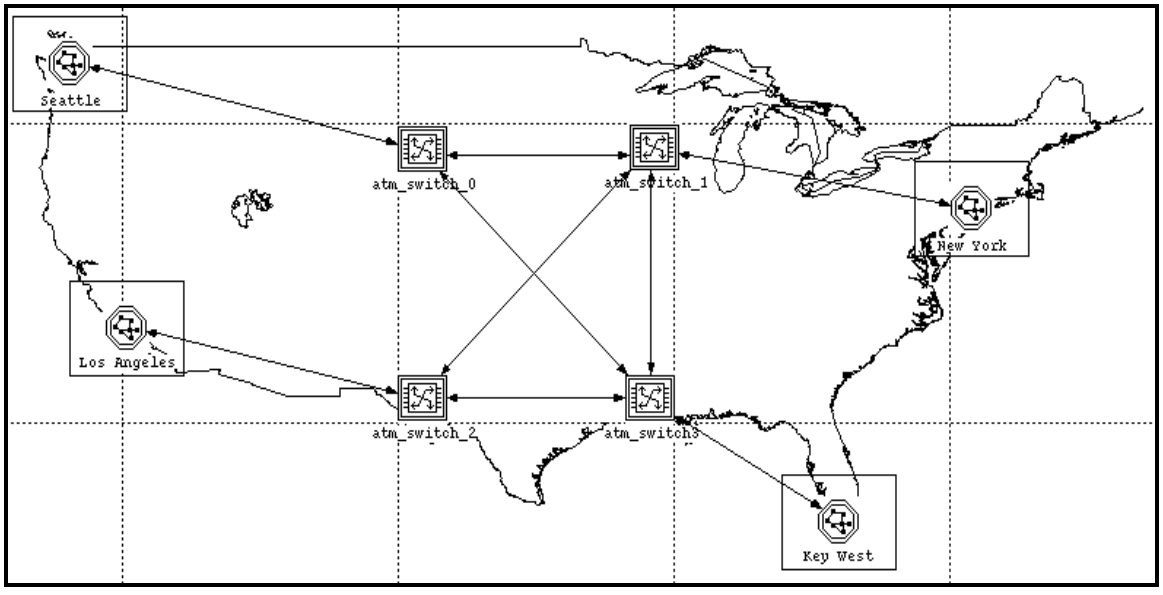


Figure 8: Network Topology of the example network

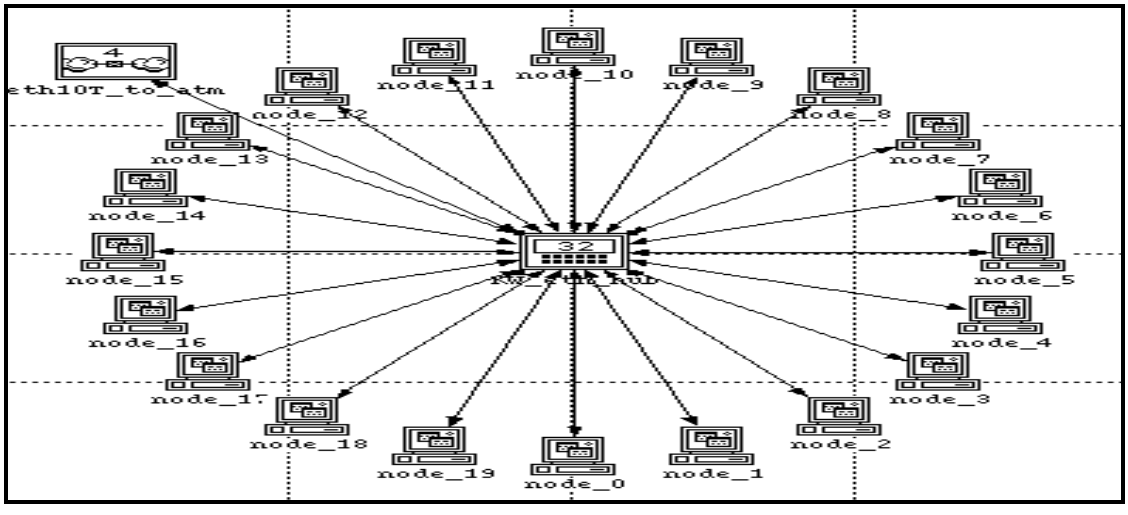
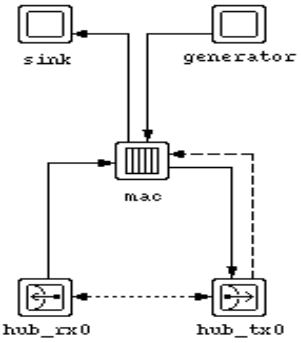


Figure 9: Bay-east subnet



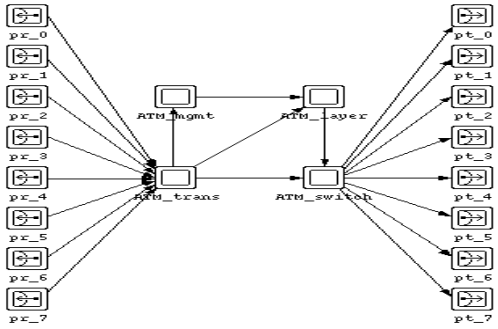


Figure 10: Node of Ethernet workstation and ATM Switch

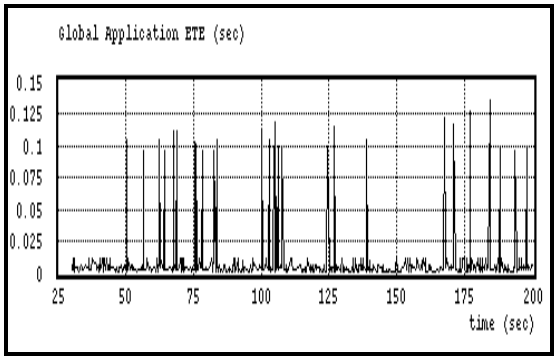


Figure 11: Global End-to-End delay

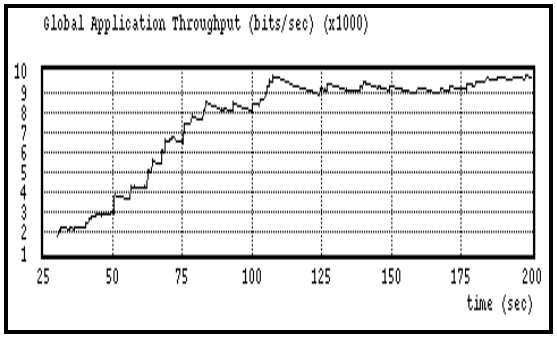


Figure 12: Global Throughput

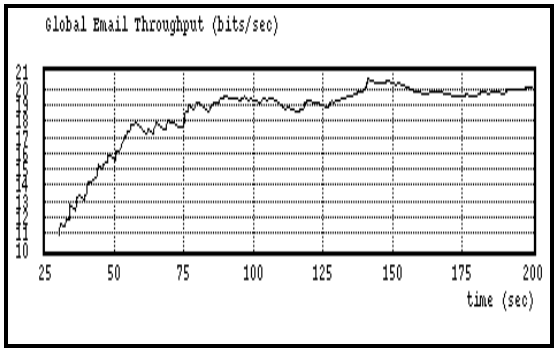


Figure 13: Throughput of the Email applications

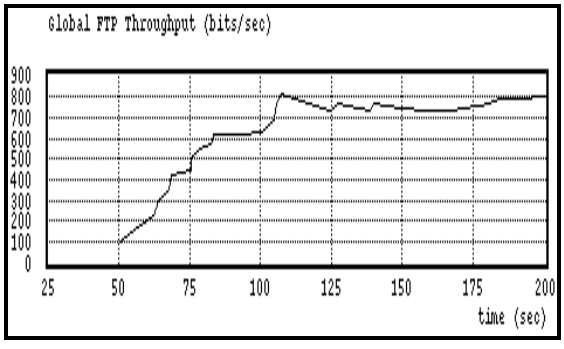


Figure 14: Throughput of the FTP applications

REFERENCES

S. Keshav, REAL 5.0 Overview, Cornell University, Available as:http://www.cs.cornell.edu/skeshav/real

NIST NET Homepage, NIST Network emulator, Available as: http://www.antd.nist.gov/itg/nistnet/

INSANE, An Internet Simulated ATM Networking Environment, Available as: http://www.ca.sandia.gov/~bmah/Software/Insane

Lewis, Barnett, "An Ethernet Performance Simulator for Undergraduate Networking", Proceeding of ACM SIGCSE Technical Symposium, 1993

R. L Bagrodia, "Designing Efficient Simulations Using Maisie", Proceedings of the 1991 Winter Simulation Conference, Dec., 1991, Phoenix, AZ, pp 243-247

Rajive Bagrodia, M. Gerla, Leonard Kleinrock, Joel Short, and T-C. Tsai, "Short language tutorial: A Hierarchical Simulation Environment for Mobile Wireless Networks", Proceedings of the 1995 WSC ns network simulator, Available at: http://www-mash.cs.berkeley.edu/ns

VINT home page, http://netweb.usc.edu/vint

T. von Eicken, A. Basu, V. Buch, W. Vogels, "U-Net: A User-Level Network Interface for Parallel and Distributed Computing", Proc. ACM Symposium on Operating Systems Principles, December 1995

J.S. Ahn, P.B. Danzig, Z. Liu, and L. Yan, “Experience with TCP Vegas: Emulation and Experiment”, Proc. ACM SIGCOMM ‘95, Boston,August 1995

S.Y. Wang and H.T. Kung, “A Simple Methodology for Constructing an Extensible and High-Fidelity TCP/IP Network Simulator”, Proceeding of Infocom 1999

Cadence Inc., Introducing BONeS 4.0, Available at: http://www.cadence.com/alta/products/bonesdat

CACI Products company, COMNET III, Available at: http://www.caciasl.com/COMNET_quick_look.html

MIL3, OPNET, Available at: http://www.mil3.com

AUTHOR BIOGRAPHY

CHANG XINJIE (changxj@ieee.org) received his diploma (B.Sc and M.Sc) in control theory and applications from the Northwestern Polytechnical University, xi’an, China, in 1995 and 1998, respectively. Currently he is a research student for the M.Eng degree in the Network Technology Research Center (NTRC), Nanyang Technological University (NTU), Singapore, where he is working on admission control and dynamic access schemes design for wireless ATM networks.