ATM NETWORK MODELS FOR TRAFFIC MANAGEMENT

By

ZHONGHUI YAO

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of

Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION

ATM NETWORK MODELS FOR TRAFFIC MANAGEMENT

BY

ZHONGHUI YAO

A Thesis/Practicum submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

Asynchronous Transfer Mode (ATM) has quickly become a popular technology in the communication and computer industries since its initial recommendation in 1988 by the International Consultative Committee for Telephone and Telegraph (CCITT). Broadband-Integrated Services Digital Network (B-ISDN) based on ATM and implemented with optical fibers provides huge bandwidth with low error rates, and is able to support a wide variety of multimedia applications such as data, voice and video concurrently.

ATM traffic management refers to a set of traffic management functions and procedures necessary for the completion and delivery of the packets travelling through networks. A properly constructed ATM network is capable of managing traffic fairly and providing effective allocation of the network capacity for different types of applications.

ATM traffic modeling is a rapidly developing research area. In order to test and evaluate the performance of ATM networks and switches, modeling stochastic processes with various distributions is required. Cellular Automaton (CA) can produce effective pseudo-random bit streams since there is little correlation among CA cells and can effectively generate random patterns which have good randomness characteristics.

OPtimized Network Engineering Tool (OPNET) provides a comprehensive development environment supporting the modeling of communication protocols, networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations. OPNET provides an opportunity to examine the higher level and more complex behavior of ATM networks.

In this thesis, OPNET is used to model ATM networks; the modeled ATM networks are used to investigate congestion and OPNET limitation; and CA is used for generating random distributions for ATM traffic. The significance and contributions of this thesis are: (1) making the first step to use CA to generate traffic distributions; (2) designing a model of a real network; and (3) proving that OPNET is a suitable tool for modeling and simulation of ATM networks, and three factors: imbalance traffic load, insufficient buffer size and bursty traffic, can cause congestion.

ACKNOWLEDGMENTS

When I start the composition of this thesis, lots of people appear in my mind. Indeed, this thesis is a result from collaboration and interaction with many people. I would like to take this opportunity to thank all the people who have contributed towards this thesis.

I would first like to thank my advisor, Dr. David C. Blight, for his guidance and encouragement through my academic years as well as his contributions and help with this thesis. Also, I would like to express my appreciation for his kind and generous personality. He deserves credit for this thesis.

I would also like to thank Dr. Robert D. McLeod and Dr. Randal J. Peters, for taking the time to be on my thesis committees, to read my thesis, and for all their effort during this thesis. My thanks also go to Dr. Robert D. McLeod and Dr. Doug J. Thomson, for their recommending me to be a graduate student, which makes my first step to this thesis.

In addition, I would like to express my thanks to following friends, graduate students and staff in TRLabs: Jeff Giesbrecht, Guy Jonatschick, Dan Erickson, Ken Ferens, Martin Meier, Budi Rahardjo, Peter Czezowski, Yair Bourlas, Jose Rueda, Jeff Diamond and Len Dacombe, for their help and discussions with my research. Special thanks go to Ming Zhang and Lihua Zhu, for their friendship and help when I was struggling for my final examination. Thanks also go to Gordon McGonigal, for his friendship and kindness.

Furthermore, I would like to thank my husband, Zaifu Zhang, and my son, Yao Zhang, for their constant encouragement and support. It is Zaifu's encouragement and hard push sometimes that make this and all things possible. It is Yao's seven-year birthday touching story that is always fresh in my mind and moves me all the time. Without their understanding, encouragement and support, I would never have finished this thesis.

Finally I would like to acknowledge the Telecommunications Research Laboratories for their financial support and providing facilities for me to do research. I would also like to acknowledge the support of the Canadian Microelectronics Corporation and Micronet.

ABBREVIATIONS

ABR	Available Bit Rate
AAL	ATM Adaptation Layer
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband-Integrated Services Digital Network
CA	Cellular Automaton
CAC	Call Admission Control
CBR	Constant Bit Rate
CCITT	International Consultative Committee for Telephone and Telegraph
CDV	Cell Delay Variation
CDVT	CDV Tolerance
CLR	Cell Loss Ratio
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTD	Cell Transfer Delay
GCRA	Generic Cell Rate Algorithm
GUI	Graphic User Interface
IP	Internet Protocol
ISO	International Standards Organization
ITU-T	International Telecommunication Union-Telecommunications Sector (Formerly the CCITT)
KP	Kernel Procedure
LAN	Local Area Network
MAC	Medium Access Control
NNI	Network-Network Interface
OAM	Operation, Administration and Maintenance
OPNET	OPtimized Network Engineering Tool
OSI	Open Systems Interconnection
PCR	Peak Cell Rate
PDF	Probability Density Function
PDU	Protocol Data Unit

QoS	Quality of Service
SAR	Segmentation and Reassembly
SCR	Sustained Cell Rate
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SONET	Synchronous Optical NETwork
STD	State Transition Diagram
TCP	Transmission Control Protocol
UBR	Unspecified Bit Rate
UNI	User-Network Interface
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VPC	Virtual Path Connection
WAN	Wide Area Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

B-ISDN supports the transmission of multimedia applications such as data, voice and images concurrently because it uses ATM as the basis and is implemented with the Synchronous Optical NETwork (SONET) technology. ATM is a high-bandwidth, low-delay, connection-oriented cell switching and multiplexing technology. SONET is a physical layer utilizing optic fibers and synchronous Time Division Multiplexing (TDM) and includes Operation, Administration and Maintenance (OAM) functions.

ATM traffic management refers to a set of traffic management functions and procedures necessary for the completion and delivery of the packets travelling through networks. A properly constructed ATM network is capable of managing traffic fairly and providing effective allocation of the network capacity for different applications. It is able to provide cost effective operations relative to the Quality of Service (QoS); support the different delay requirements of the applications; adapt to unforeseen traffic patterns (for example, unusual bursts of traffic from the various end-use devices of applications); shed traffic in certain conditions to prevent or react to congestion; enforce an allowable Peak Cell Rate (PCR) for each VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) connection; as well as monitor traffic and all VPI/VCI connections and verify their correctness.

Due to the increased number of networks in existence and their greater complexity,

designing new systems and improving the performance of existing ones has become more difficult and time consuming, therefore, it is more important to use modeling and simulation tools to deal with this complexity. OPNET is an object-oriented modeling and simulation tool which is specialized in communication protocols and networks. OPNET provides an opportunity to examine the higher level and more complex behavior of ATM networks.

In order to test and evaluate the performance of ATM networks and switches, it is necessary to generate cell distributions closely resembling the traffic existing in actual networks. ATM traffic modeling is a rapidly developing research area. From traditional Markov process models to recent self-similar traffic modeling, researchers are trying every approach (theoretical and practical) to model actual network traffic. However, modeling stochastic processes with various distributions is required no matter which approach is chosen. The key point is to make the model suitable for simulating realistic traffic sources.

A Cellular Automaton (CA) consists of a regular uniform array of any simple combinational logic with nearest-neighbor interconnections. It can produce effective pseudo-random bit streams since there is little correlation among CA cells and can effectively generate random patterns which have good randomness characteristics.

Effective ATM traffic management will require the proper management of all network resources. Comprehensive research in traffic management is required if ATM networks are to meet the anticipated demands of a broadband network. In order to understand how to manage larger ATM networks, it is important to model the key characteristics of these networks as a system, including many different interacting components, subsystems, pro-

tocols, services and environments. These are the questions we are interested in, and they will be answered in chapter 4 and 5:

1. How does an ATM network perform as a whole system?
2. What is the network performance? Are there unexpected results, and why?
3. Is OPNET suitable for modeling ATM networks as a modeling and simulation tool?
4. What is involved in constructing OPNET models for traffic management?
5. How are ATM layer and layer functions implemented in OPNET?
6. Can traffic sources generated by OPNET reflect traffic in real networks?
7. Can some crucial traffic problems, for example congestion, be studied by simulating designed OPNET models?
8. What are the limitations of the OPNET tools and OPNET models?

This thesis focuses on developing ATM network models suitable for traffic management, which involves simultaneous modeling of ATM switches, ATM connections, and ATM traffic streams. In addition, an ATM model in terms of general OPNET model development is presented. Also, a hardware implementation with One-Dimensional Linear Hybrid CA (1-D LHCA) for generating random distributions for ATM traffic sources is designed and simulated.

The thesis consists of six chapters. Chapter 2 gives an introduction to the concept of layered protocols and the OSI reference model, the ATM architecture and the B-ISDN reference model, as well as traffic characteristics and congestion control. This chapter also introduces OPNET architecture, tools and the most essential components of OPNET, Proto-C and Kernel Procedures (KPs).

Chapter 3 presents a hardware design for generating random distributions for ATM traffic. In this chapter, 1-D LHCAs are used to generate required uniform random variables which

are further used to generate a normal random sequence according to the Rejection Method. This chapter also proposes, simulates and compares several other schemes.

Chapter 4 demonstrates a model of an ATM network connected with Ethernet LANs, and presents simulation results of this model. The purpose of this chapter is to gain a general idea of using OPNET to model, simulate and analyze an actual ATM network. The chapter answers questions 1, 2, 3 and 5.

Chapter 5 deals with traffic management issues. In this chapter, three groups of models and simulations are proposed in terms of client-server network models. The first group of simulations looks at the general performance of client-server network models. The second examines the limitations of OPNET for modeling and simulation. The third investigates the most difficult and challenging problem in ATM networks: congestion. This chapter answers questions 4, 6, 7 and 8.

Chapter 6 presents the conclusions, including a summary, contributions and future work.

CHAPTER 2 LITERATURE REVIEW

ATM research activities are based upon many related theoretical concepts. In this chapter, concepts of layered protocols in data communication networks, ATM network architecture, as well as traffic management issues in ATM networks are reviewed. Also, the OPNET tools and modeling steps are introduced.

2.1 Data communication concepts

2.1.1 Layered protocols and the OSI reference model

Communication networks are usually implemented by layered protocols which decompose a complex system into several manageable parts called layers. Each layer has a well-defined interface to the adjacent layers. A layer offers a specific set of services to its higher layer. And at the same time, it receives services provided by its lower layer. Protocols are fundamental to all data communications. A protocol is an agreement between the communicating parties on how communication is to proceed. It defines a set of rules governing the format and meaning of the frames, packets, or messages that are exchanged by the peer entities within a layer [1]-[6].

The OSI (Open Systems Interconnection) reference model was developed by the International Standards Organization (ISO) as an international standard. It provides a conceptual and functional framework which allows international teams of experts to work produc-

tively and independently on the development of standards for each layer [1][2]. The OSI reference model is organized into seven layers as illustrated in Figure 1.

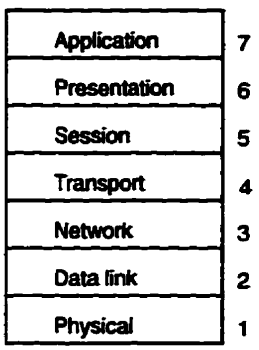

Figure 1. OSI reference model.

Layer	Functions
Physical	Concerns the transmission of unstructured bit streams over a physical medium; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium
Data link	Provides for the reliable transfer of information across the physical link; sends blocks of data (frames) with the necessary synchronization, error control, and flow control
Network	Provides upper layers with independence from the data transmission and switching technologies used to connect systems; responsible for establishing, maintaining, and terminating connections
Transport	Provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control
Session	Provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between cooperating applications
Presentation	Provides independence to the application processes from differences in data representation (syntax)
Application	Provides access to the OSI environment for users and also provides distributed information services

Table 1. OSI layer functions.

Table 1 lists the general functions defined for each layer. Note that the OSI model itself is not a network architecture because it does not specify the exact services and protocols to be used in each layer. It just tells what each layer should do [2]. However, ISO has also produced standards for all the layers, but they are not part of the reference model itself.

2.1.2 ATM and the B-ISDN reference model

ATM architecture: Asynchronous Transfer Mode (ATM) is a network architecture. It is

designed to support the integration of high quality voice, video and high speed data traffic. To the end user, it provides the ability to transport connection oriented and connectionless traffic at constant or variable bit rates. It allows for allocation of bandwidth on demand and intends to provide negotiated QoS. To a network provider, it enables the transport of different traffic types through the same network [6].

ATM networks transmit information by using fixed-size cells which consist of a 48-byte payload and a 5-byte header as depicted in Figure 2. The Generic Flow Control (GFC) field is presented only in cells between a host and the network, that is, the User Network Interface (UNI) interface. The ATM cell may contain user traffic or management/control traffic. These two types of traffic are identified by the Payload Type Identifier (PTI) field. The Cell Loss Priority (CLP) bit indicates the priority of the cell. If CLP=1, the cell is subject to being discarded by the network. Whether the cell is discarded depends on network conditions and the policy of the network administrator. The purpose of the Header Error Control (HEC) field is to check error and correct 1-bit errors [3].

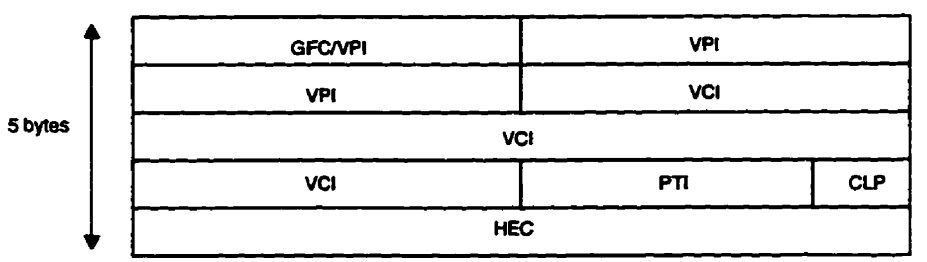

Figure 2. ATM cell header.

The Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) fields are used to identify transmission links. Generally, switches identify cells by VPIs/VCIs and translate incoming VCIs/VPIs into new VCIs/VPIs when forwarding cells to outgoing links [6].

B-ISDN reference model: B-ISDN is a layered protocol reference model specified by the International Telecommunication Union-Telecommunications Sector (ITU-T, formerly the CCITT). It is based on the principles of the OSI reference model, but does not comply with the OSI principles in a number of ways [5]. Unlike the OSI reference model, the B-ISDN ATM reference model is defined as being three-dimensional [2]-[6]. It consists of four layers and three planes as shown in Figure 3.

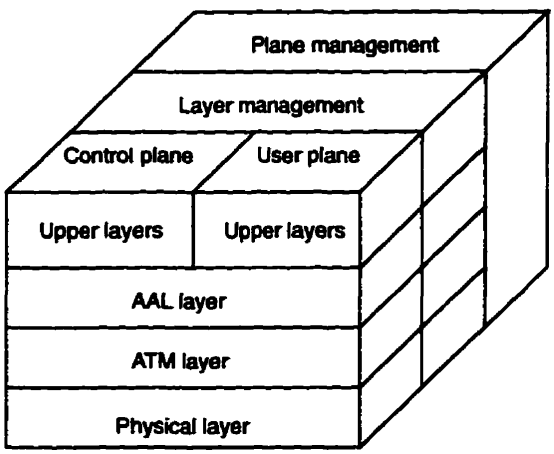

Figure 3. B-ISDN reference model.

The user plane deals with data transport, flow control, error correction and other user functions. The control plane is concerned with connection management. The layer and plane management functions relate to resource management and inter layer coordination. The physical layer deals with the physical medium: voltages, bit timing and various other issues. The physical layer can be implemented with a number of interfaces and protocols. SONET (Synchronous Optical NETwork) is one of such protocols which provides extensive multiplexing as well as Operation, Administration and Maintenance (OAM) functions. The ATM operations reside in the ATM layer and the ATM Adaptation Layer (AAL). The AAL layer is responsible for supporting the different applications in the upper layers, therefore the AAL layer is dependent on the service. At the sending machine, it segments the user traffic into 48-byte Service Data Units (SDUs) and passes

them to the ATM layer. At the receiving machine, it accepts 48-byte SDUs from the ATM layer and reassembles them into the original user traffic syntax. The ATM layer deals with cells and cell transport, including processing the cell header, flow control operations between machines, and processing the various fields in the cell header. Functions in the B-ISDN reference model are given in Table 2.

Layers	Planes/sublayers	Functions
Higher layers	Control plane	Control of connections in the user-plane signaling
	User plane	Operations on user information
AAL	CS (Convergence Sublayer)	Provision of AAL service to higher layers
	SAR (Segmentation & Reassembly)	Segmentation and reassembly of CS-PDUs
ATM		Generic flow control Cell header generation/extraction Cell switching Cell multiplexing and demultiplexing
Physical layer	TC (Transmission Convergence)	Cell rate decoupling HEC header generation/verification Cell delineation Transmission frame adaptation Transmission frame generation/recovery
	PM (Physical Medium)	Bit timing Physical medium

Table 2. Functions in the B-ISDN reference model.

2.1.3 Internet

TCP/IP reference model: The TCP/IP (Transmission Control Protocol/Internet Protocol) reference model, as shown in Figure 4, is used in Internet to connect multiple networks together in a seamless way [2]. The internet layer defines an official packet format and protocol IP, a connectionless network layer protocol. IP functions include fragmentation, reassembly and routing. IP is not designed to support reliability mechanisms such as error recovery and flow control. Those jobs are passed to the next higher layer, the transport layer which is designed to allow peer entities on the source and destination hosts to carry on a conversation. The TCP is defined in this layer and is a widely used connection-ori-

ented transport layer protocol that provides reliable packet delivery over an unreliable network. TCP performs functions, such as, connection establishment and termination, retransmission, re-sequencing and flow control. It is typically used with the IP network layer protocol.

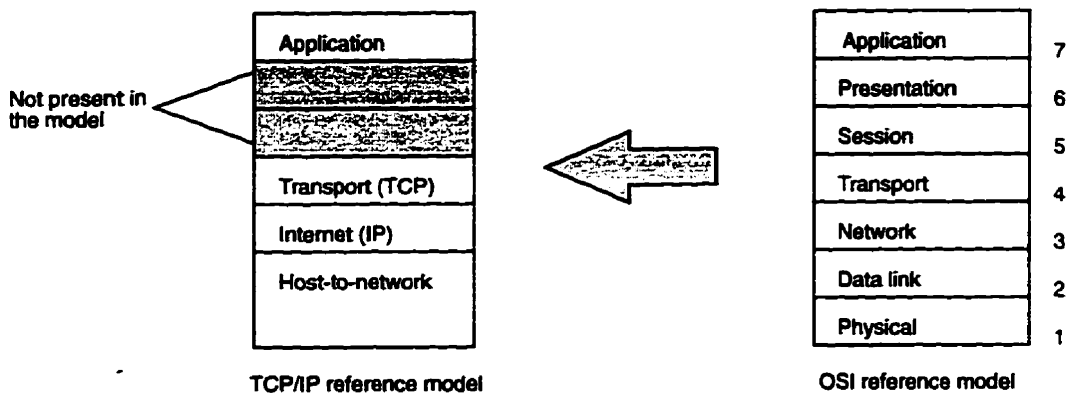

Figure 4. TCP/IP reference model.

The TCP/IP reference model does not have session and presentation layers. The application layer on top contains all the higher-level protocols, such as virtual terminal (TELNET) which allows a user on one machine to log into a distant machine and work there, File Transfer Protocol (FTP) which provides a way to move data efficiently from one machine to another, and the Domain Name Service (DNS) which is used for mapping host names onto their network addresses.

Ethernet LAN: Ethernet is a Local Area Network (LAN) technology. The operation of the Ethernet LAN is managed by the Medium Access Control (MAC) protocol which is based on the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol and has been standardized by IEEE under the name 802.3. The CSMA/CD protocol is designed to provide fair access to the shared communication channel so that all stations

connected to LAN get a chance to use the network [7]. The Ethernet MAC layer accepts data packets from a higher layer protocol, such as IP, and attempts to transmit them at the appropriate time to other stations on the bus. This mechanism is illustrated in Figure 5.

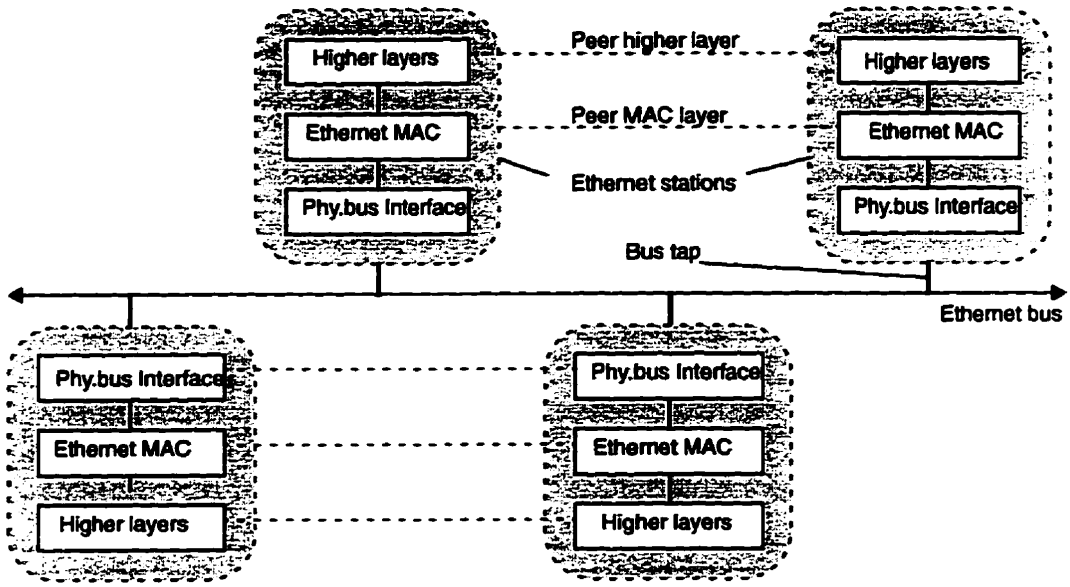

Figure 5. Ethernet MAC protocol.

The idea is very simple. When a station wants to transmit, it listens to the cable. If the cable is busy, the station waits until it goes idle; otherwise it transmits immediately. If two or more stations simultaneously begin transmitting on an idle cable, they will collide. All colliding stations then terminate their transmission, wait a random time, and repeat the whole process all over again [2]. Because the higher layer protocols can forward data at any time and the bus is a broadcast medium, it is possible that several stations attempt to transmit simultaneously, that is, collisions are unavoidable events and necessary on an Ethernet.

Gateway: A gateway on an Internet is a machine that performs relaying functions between networks. It is designed to remain transparent to the end-user applications. The

gateway does not care what type of network is attached to it and is capable of supporting any type of application because the end-user application does not reside in the gateway and the gateway considers the application message as nothing more than a transparent PDU (Protocol Data Unit). The principal purpose of the gateway is to receive a message that contains adequate addressing information and route the message to its final destination or to the next gateway [20].

2.2 Traffic management

2.2.1 Traffic Contracts and Generic Cell Rate Algorithm

When a virtual circuit is established, both the transport layer and the ATM network layer must agree on a contract defining the service [2][9]. The contract between the customer and network has basically two parts:

- The traffic descriptor which characterizes the traffic load to be offered.
- The Quality of Service (QoS) which is desired by the customer from the transport layer and accepted by the carrier from the network layer.

Traffic descriptors: Traffic descriptors define a set of parameters which describe traffic characteristics of an ATM connection. There are two mandatory traffic parameters: the Peak Cell Rate (PCR) and the Cell Delay Variation Tolerance (CDVT). PCR is the maximum rate at which the sender is planning to send cells. This parameter specifies an upper bound on the traffic that can be submitted on an ATM connection. CDVT tells how much variation will be present in cell transmission times. In addition, there are optional traffic parameters: the Sustained Cell Rate (SCR) and the burst tolerance which allow a finer definition of traffic characteristics that enable the network to do more efficient resource allo-

cation. The above four traffic parameters (PCR, CDVT, SCR and the burst tolerance) are defined by the Generic Cell Rate Algorithm (GCRA) which represents a deterministic rule. Since the traffic is defined based on deterministic rules, it does not carry the necessary information to characterize the statistical behavior of the stochastic cell arrival process.

Quality of Service (QoS): To make it possible to have concrete traffic contracts, the ATM standard defines a number of QoS parameters whose values can be negotiated between the customer and the carrier [2]. Three negotiable parameters are the Cell Loss Ratio (CLR), the Cell Transfer Delay (CTD) and the Cell Delay Variation (CDV). These three parameters describe characteristics of the network and are measured at the receiver. CLR measures the fraction of the transmitted cells that are not delivered at all or are delivered so late as to be useless (e.g., for real-time traffic). CTD is the transit time from source to destination. CDV measures how uniformly the cells are delivered. There are also several other QoS parameters which are not negotiable. Their definition can be found in [2].

GCRA: GCRA is implemented as a continuous-state leaky bucket algorithm. It is employed in traffic policing and is a part of the user/network service contract. In the ATM Forum specification, the GCRA consists of two parameters: the increment I and the limit L. The notation GCRA(I,L) means GCRA with the increment parameter set to I and the limit parameter set to L. The increment parameter affects the cell rate. The limit parameter affects cell bursts. The GCRA allows, for each cell arrival, a 1-unit leak out of the bucket per unit of time. In its simplest terms, the bucket has finite capacity, and it leaks out at a continuous rate. Its contents can be filled (incrementally) by I if L is not exceeded. Otherwise, the incoming cell is defined as nonconforming [3].

2.2.2 Service categories

Services in ATM networks are classified as: constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR) and unspecified bit rate (UBR) [2][10]. There is also a block transfer mode under development.

CBR: CBR service refers to a service where fixed-size packets arrive at deterministic intervals. With CBR service, CTD and CDV are tightly constrained and low CLR is required. This service category is suited to interactive audio and video stream.

VBR: VBR service is divided into two subclasses, real time (RT-VBR) and non-real time (NRT-VBR). RT-VBR is intended for services that have variable bit rates combined with stringent real-time requirements such as interactive compressed video (e.g., videoconferencing). With RT-VBR service, statistical multiplexing is used and this may cause a small non-zero random cell loss. CTD and CDV are tightly constrained. NRT-VBR is for traffic where timely delivery is important but certain amount of CDV can be tolerated by the application. An example is multimedia email.

ABR: ABR is designed for bursty traffic whose bandwidth range is known roughly. The ABR service class avoids having to make a long term commitment to a fixed bandwidth. It guarantees services at designed bandwidth and does its best (but does not guarantee) to provide services at the bandwidth higher than the designed bandwidth whenever the network capacity is available. It is the only service where the network provides rate feedback to the sender, asking it to slow down when congestion occurs.

UBR: UBR makes no promises and gives no feedback about congestion. It uses any left-over capacity, no CTD or CDV or CLR is constrained. Example applications are email

and file transfer.

2.2.3 Network congestion and traffic control

Network congestion: Network congestion can occur if a large number of traffic sources become active simultaneously. Under a congestion situation, the queue length may become very large in a short time, resulting in buffer overflow, cell loss and increase of delay [10]. One of the main causes of congestion is that traffic in ATM networks is often bursty [2][11].

Congestion is a global issue, involving the behavior of all the parts of a network [2]. Congestion is also a dynamic problem, static solutions such as, increasing the buffer size, utilizing higher speed links and implementing faster processors, are not sufficient to solve the problem [18]. The lack of the memory space can cause congestion. Increasing the memory space can help in some degree, but can also make the situation worse because of the long queue and long delay introduced by a large memory. The low bandwidth link and the slow processor can also cause congestion. However, upgrading part of system, for example, increasing the bandwidth of the link without upgrading the processor or vice versa, can create a mismatch or imbalance between parts of system which can aggravate the congestion problem [2].

Therefore, congestion control is very important in an ATM network. The objectives of traffic control and congestion control for ATM are to support a set of QoS parameters and classes for all ATM services and minimize network and end-system complexity while maximizing network utilization. Generic functions and procedures for congestion control have been proposed by ATM Forum and studied by many researchers [9]-[18].

Traffic control: Traffic control specifies the actions taken by the network to avoid congestion. Some traffic control functions have been identified as: resource management, CAC (Call Admission Control), UPC (Usage Parameter Control) and congestion control [17]. CAC is a set of procedures that operate at the UNI, encompassing actions taken by the network to grant or deny a connection to a user. It is a major tool for preventing congestion. UPC is designed to monitor and control traffic, and to check the validity of the traffic entering the network. Traffic shaping is about regulating the average rate (and burstiness) of data transmission [3].

The scope of this thesis is modeling and simulation of ATM networks to obtain parameters in terms of QoS, that is, CLR, CTD and CDV. In addition, the congestion situation is examined. Follows are questions we are interested in, and they will be answered in Section 5.2.3.

1. Can the designed model make congestion happen?
2. What are the major factors that cause congestion?
3. What statistics (of the OPNET model) can reflect the congestion problem?

2.3 OPNET

2.3.1 OPNET architecture and tools

OPNET provides a comprehensive development environment supporting the modeling of protocols, communication networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations. The *OPNET Environment* incorporates tools for all phases of a simulation study, including model design, simulation, data collection and analysis [19]-[24].

OPNET architecture: With OPNET, there are three phases of the model development: *Specification, Data Collection and Simulation,* and *Analysis.* These three phases are performed in sequence with a cycle. Specification is divided into two parts: initial specification and re-specification. The latter belonging to the cycle as illustrated in Figure 6.

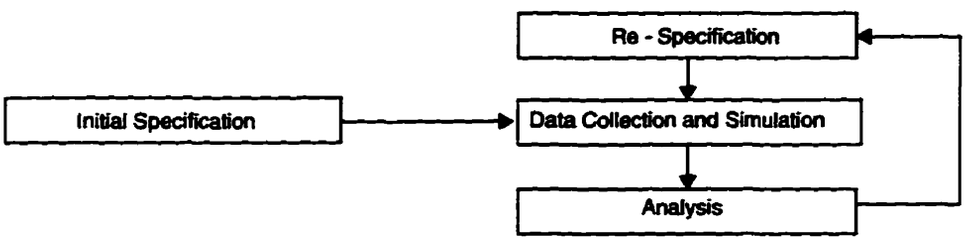

Figure 6. Model development cycle.

Model specification is the task of developing a representation of the system to be studied. Models in OPNET are structured hierarchically. Specialized tools (editors) perform tasks at different levels of the hierarchy. This provides an intuitive modeling environment and also permits re-use of lower level models. OPNET simulations are capable of producing many types of output. In most cases, modelers use the types of data directly supported by OPNET which are *output vectors, output scalars* and *animation.* However, developers are able to define their own types of output files, including text reports, proprietary binary files, etc. because of the general programmability of process and link models. The third phase involves examining data collected during simulations. OPNET provides access to data with the *Analysis Tool* which is essentially a graphing and numerical processing environment.

OPNET Modeler Tool: *OPNET Modeler* is used to study system behavior and performance. *OPNET Planner* is used for delivering a modeling environment to "end-user".

OPNET presents its capabilities in the form of eight distinct tools. Each tool allows the user to perform some set of related OPNET functions within a window of OPNET graphical environment. Table 3 lists *OPNET Modeler's tools* and identifies each one's purpose.

OPNET Tools	Functions
Network Editor	Specify network topology and configure nodes and links.
Node Editor	Create models of nodes by specifying internal structure and capabilities.
Process Editor	Develop models of decision-making processes representing protocols, algorithms, resource managers, operating systems, etc.
Parameter Editor	Specify complex data objects, including packet formats, link models, antenna patterns, modulation curves.
Probe Editor	Identify sources of statistics and animation that are to be make active during a simulation.
Simulation Tool	Design and run sequences of simulations, each potentially configured with different inputs and/or outputs.
Analysis Tool	Plot and process numerical data generated by simulations.
Filter Editor	Define numerical processing that can be applied to data in the Analysis Tool.

Table 3. OPNET modeler tool summary.

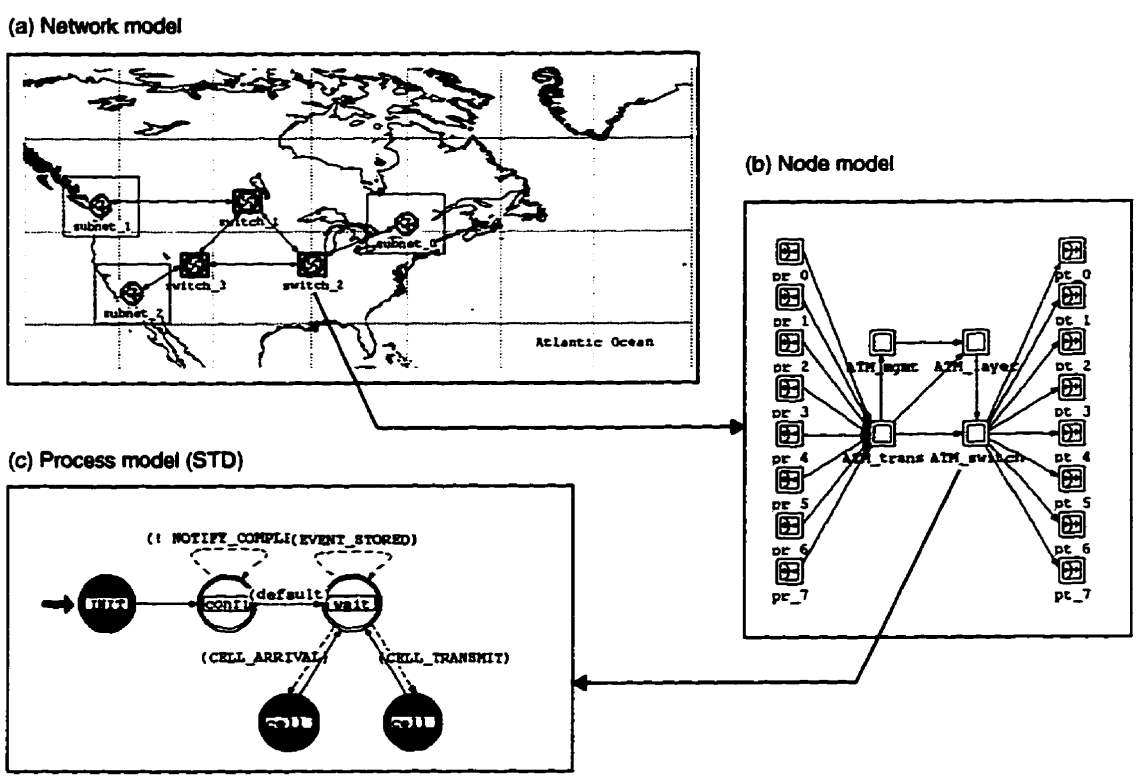

Figure 7. Three level modeling.

Figure 7 shows an example of an OPNET model with three levels. In Figure 7, the net-

work model is specified in the *Network Editor* and it consists of several nodes. Each node is defined in the *Node Editor* and includes several modules. Modules are designed in the *Process Editor*, they could be pre-defined or programmable. For example, *processor* and *queue* modules are programmable, users can design their own procedures for these two modules.

2.3.2 Processes

Process models are key objects of an OPNET model since they are used to define behavior for processor and queue modules. For instance, Figure 7-c represents the process model, *ams_atm_sw.pr.m*. This process model is located in the *ATM_switch* module. It includes five states and defines three interrupt conditions. The certain interrupt conditions invoke certain processes and results the change of the state. The *ams_atm_sw* process receives ATM cells from either the *ATM_layer* or *ATM_trans* modules (Figure 7-b), performs switching functions, queues the cells in an output port buffer, and sends them to the transmitter [19]. Obviously, the network model (Figure 7-a) does not work without behavioral modeling on the process model level (Figure 7-c).

Processes are tasks that processors and queues execute. They may be created and destroyed based on dynamic conditions that are analyzed by the logic of the executing processes. Processes are usually grouped to be executed within the same processor or queue. However, only one process may be executing at any time.

A process is considered to be executing when it is processing new instructions that are part of its process model. When a process begins execution it is said to be invoked. A process that is currently executing can invoke another process in its process group to cause it

to begin executing. When this happens, the invoking process is temporarily suspended until the invoked process blocks. A process blocks by indicating that it has completed its processing for its current invocation. Once the invoked process has blocked, the invoking process resumes execution where it had left off, in a manner similar to the procedure-call mechanism in a programming language such as C.

Processes in OPNET are designed to respond to interrupts and/or invocations. Interrupts typically correspond to events such as messages arriving, timers expiring, resources being released, or state changes in other modules. Once a process has been invoked due to an interrupt, it may invoke other processes in the group and these may in turn invoke other processes, etc. An interrupt's processing is completed when the first process that was invoked blocks.

2.3.3 Proto-C and Kernel Procedures

Processes are expressed in a language called *Proto-C* which is a combination of state transition diagrams (*STDs*), *Kernel Procedures* (*KPs*) and *C* programming language. A process model's STD defines a set of primary states that the process can enter. And for each state, a STD also defines the conditions that would cause the process to move to another state. Transitions in STD describe the possible movement of a process from state to state and the conditions under which such changes may take place. The process model in Figure 7-c is an example of a STD. Proto-C models allow actions to be specified at various points in the *Finite State Machine.*

KP is a library of high level commands (over 300 procedures). They are procedures that can be called from within process models, Transceiver Pipeline stages, C functions which

have been scheduled as interrupts, or simply C functions which are directly or indirectly invoked from one of these contexts. For the most part, the simulation services are accessed through KPs. KPs are categorized by primary functions, based on the types of objects they attempt to manipulate. There are nineteen categories of KPs. The collection of KPs within a category is called a package, and KPs within the same package share a common package keyword in their procedure names. Table 4 lists the capabilities provided by the KP libraries.

Package	Applications
Anim	Support for custom animation development.
Dist	Probability distribution and random number generation.
Ev	Event list and event property query; event cancellation.
Ici	Formal interfaces between processes; association of information with interrupts.
Id	Identification of objects.
Ima	In-simulation query and modification of object attributes.
Intrpt	Query of interrupt properties; control of interrupt handling.
Pk	Creation, destruction, modification, analysis, and transmission of packets.
Prg	Programming support: linked lists, memory, string manipulation, debugging.
Pro	Creation, invocation of processes; process group shared memory.
Q	Queueing statistics; high-level queueing operations.
Rte	Basic routing support for static routing implementations.
Sim	Simulation control: customized messaging, simulation execution control.
Stat	Custom statistic generation; intermodule signalling via statistic wires.
Strm	Communication between modules via packet streams, packet delivery.
Subq	Low-level queueing operations: antenna patterns, modulations.
Tbl	Accessing of tabular data: antenna patterns, modulations.
Td	Setting and getting of transmission data for custom link models.
Topo	Query of model topology (e.g., for automatic discovery and configuration).

Table 4. Major simulation Kernel Procedure packages.

CHAPTER 3 MODELING ATM TRAFFIC DISTRIBUTIONS

This chapter explores a general methodology to use One-Dimensional Linear Hybrid CA (1-D LHCA) to generate random distributions for ATM traffic. CAs are used since they can effectively generate random patterns with good randomness characteristics. The chapter is organized into four sections: Section 3.1 presents the motivation; Section 3.2 is an introduction to CA and the algorithm for generating a random distribution; Section 3.3 deals with CA hardware implementation, simulation and comparison with other proposed schemes; and Section 3.4 is a conclusion. Related work is represented in [25].

3.1 Motivation

As we have already shown, ATM provides for a wide variety of traffic with different characteristics. To test and evaluate the performance of ATM networks and switches, it is necessary to generate cell arrivals closely resembling the traffic existing in actual networks [11][26]. ATM traffic modeling is a rapidly developing research area. From traditional Markov process models to recent self-similar traffic modeling, researchers are trying every method (theoretical and practical) to investigate actual network traffic. Some common approaches for modeling traffic arrivals are Poisson process, exponential distribution, Bernoulli distribution and Markov process [11][26]. The most recent research has

focused on self-similar traffic models [27][28][29]. However, modeling stochastic processes with various distributions is required no matter which approach is chosen. The key point is to make the model suitable for simulating realistic traffic sources.

In general, packet arrivals can be illustrated as Figure 8 shows, where t represents the time, T is the frame interarrival time which reflects burstiness of arrivals and dt is the interval of bits which depends on the bit rate of the n^{th} frame. In Figure 8, with the different type of the traffic sources, both T and dt could be either constants or variables, or one of them could be a constant and the other a variable.

Usually, the variable T and dt are of a certain random distribution, depending on the network applications. Among various ATM traffic sources, an arrival process of cells from a video source is fairly complicated due to the strong correlation among arrivals. As an example, a video traffic source is considered in this thesis.

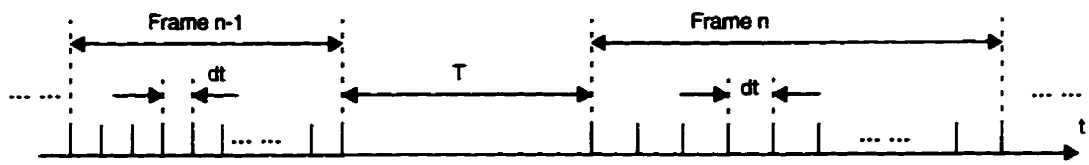

Figure 8. Traffic arrivals.

3.2 General concepts of CA

Cellular Automaton (CA) as a random sequence generator has been studied by many researchers [30]-[36]. These include: the analysis of CA properties; the study of CA behavior; the method of constructing CAs with better randomness characteristics and the algorithm for searching CA rules corresponding to primitive characteristic polynomials. Wolfram first presented a random sequence generator by using 1-D CA with two non-lin-

ear rules (rule 30 and 45 or equivalently rule 86 and 75). Later, most of work in this area is based on 1-D CA with linear rule 90 and 150.

3.2.1 One-Dimensional Linear Hybrid CA (1-D LHCA)

1-D LHCAs used as test pattern generation engines have been extensively investigated in the IC (Integrated Circuit) testing area [31]-[36]. As each cell in 1-D LHCA is connected locally, 1-D LHCA has the advantages of reduced local capacitance and a regular structure which result in efficient layouts and improved operating frequencies. Also, 1-D LHCA can provide effective pseudo-random bit streams since the correlation among CA cells is less than that among the Linear Feedback Shift Register (LFSR). Figure 9 illustrates a 1-D CA with 3-neighborhood dependency.

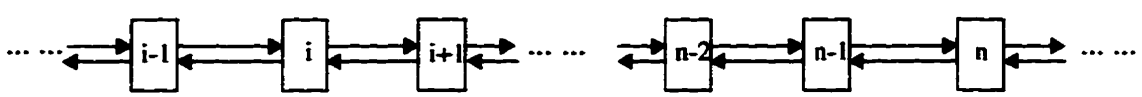

Figure 9. A 1-D CA with 3-neighborhood dependency.

A CA has a regular structure consisting of an array of 1-bit memory elements and a combinational logic determining the next states of the memory elements. In 1-D CA, all the cells are arranged in a linear array. Of particular importance is the 3-neighborhood 1-D CA, where the next state of a cell depends on itself and on its two neighbors (3-neighborhood dependency). Mathematically, the state q of the i^{th} cell at time $t+1$ is denoted as:

$$q_i^{t+1} = g(q_i^t, q_{i-1}^t, q_{i+1}^t) \qquad \text{(Eq-1)}$$

where q_i^t denotes the state of the i^{th} cell at time t, and g is the next state function. Since g is a Boolean function of 3 variables (3-neighborhood dependency), there are 2^{2^3} or 256

possible next-state functions. The next-state functions are usually called rules. If all the CA cells obey the same rule, the CA is said to be uniform; otherwise it is hybrid. If the next-state transition function is expressed algebraically linear (*XOR* function), the rule is said to be additive; otherwise it is nonadditive. Table 5 explains how rules 90 ($90_{(10)}$=$01011010_{(2)}$) and 150 ($150_{(10)}$=$10010110_{(2)}$) are defined.

	3-neighborhood dependency	8 possible states								Rule
Present states	q_{i-1}, q_i, q_{i+1}	111	110	101	100	011	010	001	000	
Next state of q_i	$(q_{i-1}\bar{q}_{i+1}) + (\bar{q}_{i-1}q_{i+1})$	0	1	0	1	1	0	1	0	90
Next state of q_i	$(q_{i-1}\bar{q}_i\bar{q}_{i+1}) + (\bar{q}_{i-1}q_i\bar{q}_{i+1}) + (\bar{q}_{i-1}\bar{q}_iq_{i+1}) + (q_{i-1}q_iq_{i+1})$	1	0	0	1	0	1	1	0	150

Table 5. Rules 90 and 150.

1-D LHCA with rule 90 and 150 is considered in this thesis, since a CA formed from cells with a single rule generally has a sequence length shorter than 2^n-*1* (where *n* is the number of cells in the array). It has been proved that the maximum sequence length 2^n-*1* could be generated for certain combinations of rule 90 and 150 cells [31][32][33][36]. To generate a high quality random sequence, the boundary condition and primitive characteristic polynomial must be concerned.

3.2.2 Boundary conditions

1-D CA can be constructed with null boundary and periodic (cyclic) boundary conditions. Wolfram stated that the boundary conditions have little or no effect on the bit streams [30]. Bardell proposed a conjecture: there are no hybrid 90/150 CA with cyclic boundary conditions that has the primitive characteristic polynomial [33]. Later, S. Nandi and P. P. Chaudhuri gave a formal proof of the conjecture and proposed a maximum sequence CA

with the intermediate boundary condition. This new boundary condition has better randomness characteristics than null boundary condition [35].

When two ending cells of a CA array are connected to constant logic 0 or 1, CA is said to have the Null Boundary (NBCA). When two ending cells are connected cyclically in a chain, the CA has Periodic Boundary (PBCA). The Intermediate Boundary (IBCA) lets two ending cells connect to the next cells of their adjacent cells. Figure 10 illustrates these three boundary conditions.

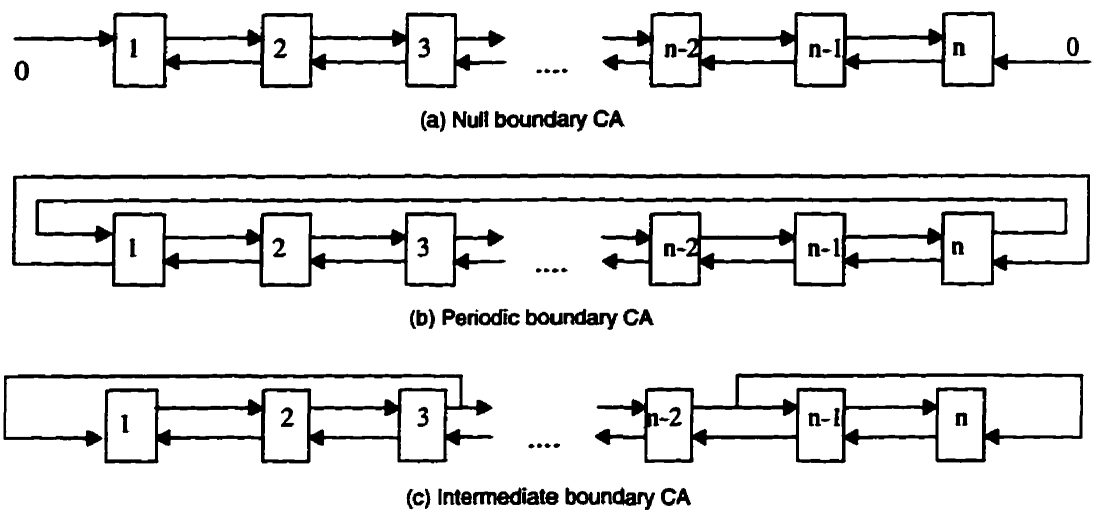

Figure 10. Boundary conditions.

Figure 11 gives an example of a 6-cell NBCA with rule vector <150, 90, 90, 90, 90, 90>. The next state functions of this NBCA are:

$$X_1(t+1) = X_1(t)+X_2(t) \quad \text{(Eq-2)}$$

$$X_2(t+1) = X_1(t)+X_3(t) \quad \text{(Eq-3)}$$

$$X_3(t+1) = X_2(t)+X_4(t) \quad \text{(Eq-4)}$$

$$X_4(t+1) = X_3(t)+X_5(t) \quad \text{(Eq-5)}$$

$$X_5(t+1) = X_4(t)+X_6(t) \quad \text{(Eq-6)}$$

$$X_6(t+1) = X_5(t) \qquad \text{(Eq-7)}$$

Where the addition in equations Eq-2 to Eq-7 is modulo-2. This example will be used later to explain how to obtain the characteristic polynomial of a 1-D LHCA.

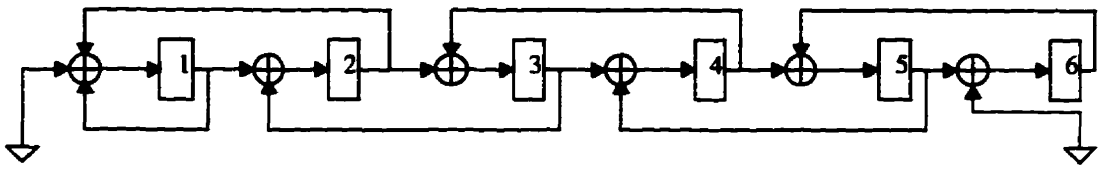

Figure 11. 6-cell NBCA with rule vector <150, 90, 90, 90, 90, 90>.

3.2.3 Primitive characteristic polynomial

The existence of the primitive characteristic polynomial for an n cell CA ensures that the CA will run through the maximum length of 2^n-1 distinct non-zero states. The maximum length CA is the necessary condition for generating high quality pseudo-random patterns. When an n^{th} order characteristic polynomial can only be divided by one and itself, and can only divide $x^{2^m-1}-1$ with m ($>= n$), it is a primitive characteristic polynomial.

Equations Eq-2 to Eq-7 can be written in matrix notation:

$$X(t+1) = AX(t) \qquad \text{(Eq-8)}$$

Where A is the transition matrix and has the form:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{(Eq-9)}$$

The transition matrix A is the next state operator. The next state $X(t+1)$ of CA can be

obtained by multiplying A (modulo-2) with the present state vector $X(t)$. For instance,

if $X(t) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$, then the next state is: $X(t+1)=AX(t)= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$.

By setting the determinant to zero, the characteristic equation for Eq-8 can be obtained,

$$det(A-\lambda I) = 0 \qquad \text{(Eq-10)}$$

The expansion by minors of a determinant of this form [33] can be defined recursively as:

$$M_i = a_{i+1,\,i+1}M_{i+1} + M_{i+2} \qquad \text{(Eq-11)}$$

Where M_i is the i^{th} major diagonal minor with $M_n=1$ and $M_{n+1}=0$. The recursive expansion Eq-12 is obtained after derivation (Appendix I).

$$M_0 = \lambda^6 + \lambda^5 + \lambda^4 + \lambda + 1 \qquad \text{(Eq-12)}$$

Thus the characteristic polynomial for the example shown in Figure 11 is:

$$p(x) = x^6 + x^5 + x^4 + x + 1 \qquad \text{(Eq-13)}$$

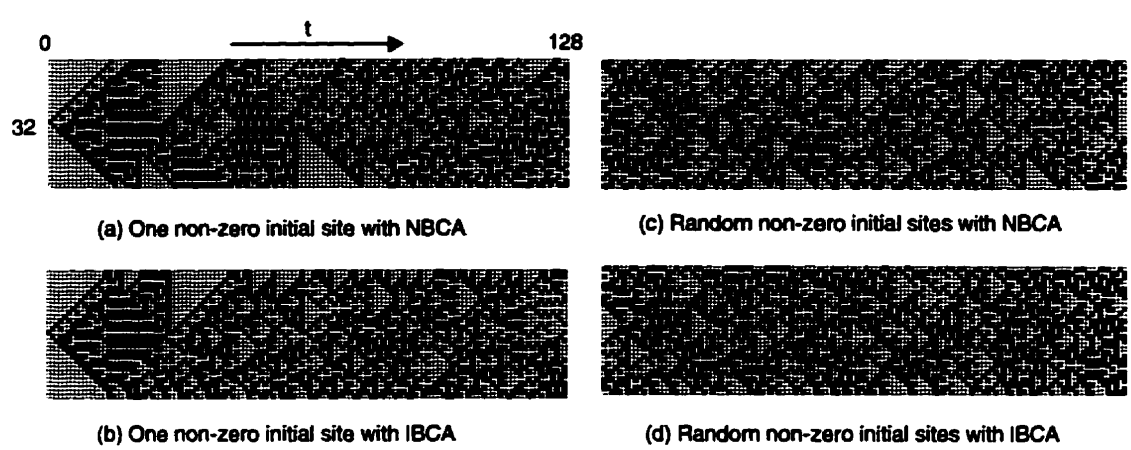

Figure 12. Random patterns generated by 1-D LHCA.

Figure 12 shows random patterns generated by 1-D LHCA with different initial states and boundary conditions. The number of cells chosen is 32 (*n=32*), the sequence length is 128 (*t=128*). The rule vector is chosen as <1000 0000 0000 0010 0000 0000 0000 0000> because the primitive characteristic polynomial exists under this rule [34].

3.3 Generating random distributions with 1-D LHCA

3.3.1 Continuous-state autoregressive Markov model

A video source can be approximated by a continuous-state autoregressive (AR) Markov process which is inferred from the experimental results [37]. Mathematically, an *M* order AR Markov process is described by Eq-14:

$$\lambda(n) = \sum_{m=1}^{M} \alpha_m \cdot \lambda(n-m) + \beta \cdot \varpi(n) \qquad \text{(Eq-14)}$$

Where $\lambda(n)$ represents the source bit rate during the n^{th} frame; *M* is the model order; $\varpi(n)$ is a sequence of independent normal random variables; β and α_m ($m=1, 2,..., M$) are coefficients. This model can describe the cell generation process of a video source accurately. It has been shown that the first-order AR Markov model is sufficient for most purposes [11]. Eq-15 is the first-order AR Markov model derived from Eq-14.

$$\lambda(n) = \alpha_1 \cdot \lambda(n-1) + \beta \cdot \varpi(n) \qquad \text{(Eq-15)}$$

Therefore, if $\varpi(n)$ is obtained, a first-order AR Markov model can be constructed. Furthermore, this model can be used to model a video traffic source to simulate the performance of an ATM network. Obviously, generating $\varpi(n)$, a sequence of normal random variables, is indispensable in this case.

3.3.2 Rejection Method

Generally, the Rejection Method can be used to simulate a random variable with a given density function $g(x)$. This $g(x)$ can be the basis for simulating the continuous distribution with density function $f(x)$. This can be done by simulating a random variable Y from g and then accepting this simulated value with a probability being proportional to $f(Y)/g(Y)$. Let C be a constant, and $f(y)/g(y) \le C$, for all y. Since each iteration during the execution of the algorithm will independently result in an accepted value with probability $P\{U<=f(Y)/Cg(Y)\}=1/C$, it follows that the number of iterations is geometric with mean C [38]. The normal random variable density function with mean 0 and variance 1 is given by:

$$f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, -\infty < x < \infty \qquad \text{(Eq-16)}$$

Let $f(x)=2f_g(x)$ and $0 < x < \infty$, that is,

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-x^2/2}, 0 < x < \infty \qquad \text{(Eq-17)}$$

Considering $g(x) = e^{-x}, 0 < x < \infty$ as the basis for simulating $f(x)$, then

$$f(x)/g(x) = \sqrt{(2e)/\pi} e^{-(x-1)^2/2} \le \sqrt{(2e)/\pi}. \qquad \text{(Eq-18)}$$

Let $C = \sqrt{(2e)/\pi}$, Eq-19 can be obtained.

$$f(x)/g(x) \le C \qquad \text{(Eq-19)}$$

3.3.3 Algorithm and simulation

The algorithm is listed in Table 6 where z is a normal random variable with mean 0 and variance 1. To obtain a normal random variable with mean μ and variance σ^2, z can be

replaced by $\mu+\sigma z$. The other three variables *y1*, *y2* and *y* created in the algorithm are all exponential random variables with rate 1.

Step 1: Generate r1, r2 and u;
Step 2: y1=-log(r1), y2=-log(r2);
Step 3: If $y2-(y1-1)^2/2>0$, set $y=y2-(y1-1)^2/2$, go to step 4; otherwise go to step 1;
Step 4: Set

$$z = \begin{cases} y1 & if \quad u \le 1/2 \\ -y1 & if \quad u > 1/2 \end{cases}$$

Table 6. Algorithm for generating a normal random sequence

The idea is to use 1-D LHCAs to generate three independent uniform variables (*r1*, *r2* and *u*). Based on Transformation Method, if *r* is a uniform (0,1) variable, *y=-log(r)* is an exponential variable with rate 1 [38]. In order to make *r1*, *r2* and *u* independent, several methods could be used, using different boundary conditions; or choosing different generation rules. Here, *r1* and *r2* use the same rule <1000 0000 0000 0010 0000 0000 0000 0000> but different boundary conditions. *r1* is generated with IBCA, and *r2* with NBCA. *u* is generated with null boundary conditions and a different rule: <0111 1101 1110 1111 1001 1110 0100 0101>, as the primitive characteristic polynomial exists under this rule [36].

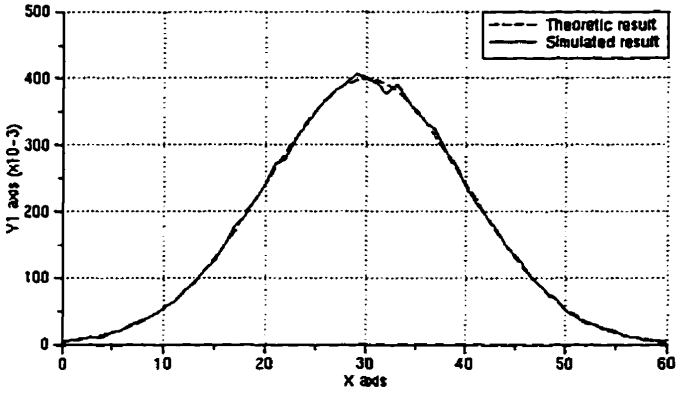

Figure 13. PDF curve of a normal random sequence generated by 1-D LHCA.

Figure 13 shows the simulation result which is the Probability Density Function (PDF) curve of the generated normal random sequence.

3.4 Hardware implementation and simulation

When implementing a normal random sequence generator with a digital circuit, several issues have to be considered: (1) the three generated uniform random variables *r1*, *r2* and *u* must be independent; (2) when calculating an exponential variable from a uniform variable, both computation efficiency and hardware cost need to be traded off; and (3) the least number of steps generating an output *z* is preferred.

3.4.1 1-D LHCA hardware design

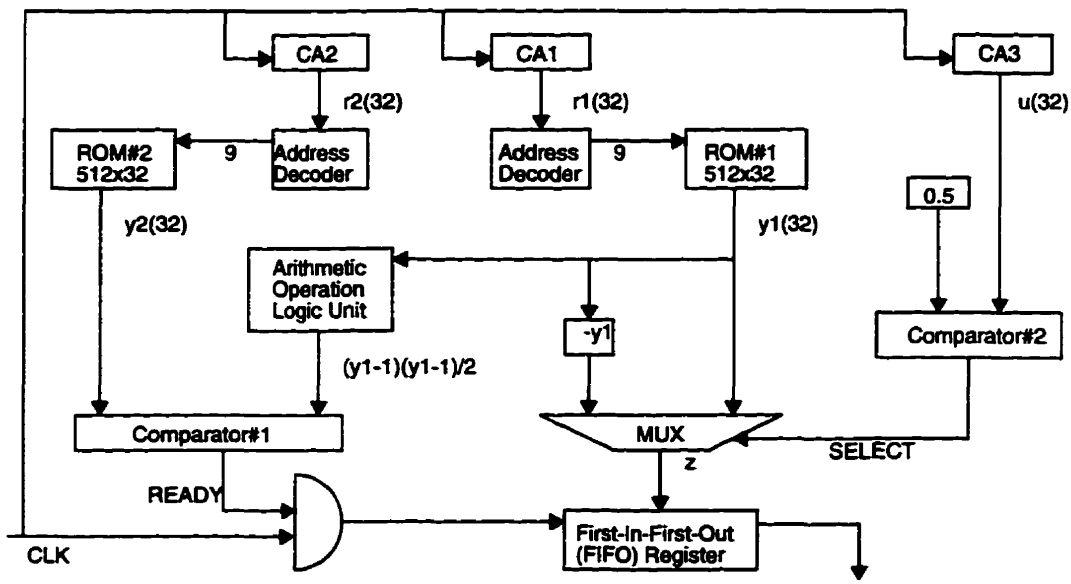

Figure 14. The block diagram of the hardware CA implementation.

The block diagram of the designed hardware CA implementation is shown in Figure 14. Three different properly configured 32-cell 1-D LHCAs generate three independent uniform random variables *r1*, *r2* and *u* in each clock cycle. Two exponential variables are obtained from two ROMs which consist of pre-stored values of the *-log(x)* function. After the Arithmetic Operation Logic unit, *y2* and $(y1-1)^2/2$ are compared. If $y2>=(y1-1)^2/2$, $READY \cdot CLK$ becomes enabled. Meanwhile, *u* is compared with 0.5. If *u<=0.5*, *z=y1*;

otherwise, $z=-y1$. The advantage of using a ROM as a look-up table is that the computation of the *-log(x)* function can be done in one clock cycle. This makes the process of generating a normal random sequence fast. With the implementation in Figure 14, a normal random number can be generated in $C\,(=\sqrt{(2e)/\pi}\equiv 1.32)$ clock cycles on average.

In Figure 14, two ROMs (512x32) are utilized to pre-store the approximated *-log(x)* values of the variable x which uniformly distributes in (0,1). Two address decoders are required to approximate values of x and its neighbors $x\pm dx$ in (0,1). By dividing (0, 1) into m small intervals (linear or non-linear), results of *-log(x)* of each x and its neighbors are approximated by the pre-calculated *-log(x)* values in ROM. This can be implemented by mapping $(x\text{-}dx, x+dx)$ to address y which points to the value of *-log(x)* in hardware. For instance, if choosing the number of dividing points to be 512, $dx = 1/512$. Starting from 0, when $0<x<=dx$, $y=-log(dx)$; $dx<x<=2dx$, $y=-log(2dx)$,..., $idx<x<=(i+1)dx$, $y=-log((i+1)dx)$, $i=0,1,2,...,511$. Figure 15 illustrates a many-to-one mapping. In Figure 15, the values of x and its neighbors $x\pm dx$ are mapped to address y of ROM, where the value of *-log(x)* is stored. Figure 16 shows the simulation results of a many-to-one mapping with SUN built-in function *rand()* when 256, 512 and 1024 are chosen.

Figure 15. A many-to-one mapping.

In Figure 14, a First-In First-Out (FIFO) register is also utilized to store a packet of normal random numbers. Since the Rejection Method performs $C\,(=\sqrt{(2e)/\pi}\equiv 1.32)$ steps on

average before it produces an output, it may need more than one clock cycle to generate one random number. The write enable control signal to the FIFO register is given by *READY·CLK*.

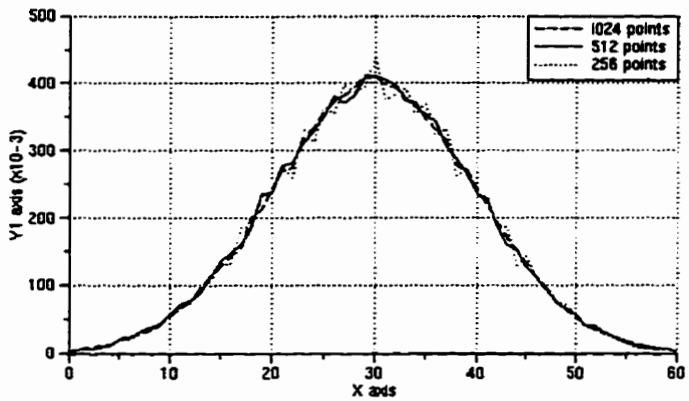

Figure 16. Using 256, 512 and 1024 points to approximate -log(x).

The simulation program is implemented based on the hardware design in Figure 14. The simulation result is shown in Figure 17. It can be seen that two curves (the dotted one is the theoretic and the solid one is the one generated by the design) match very well.

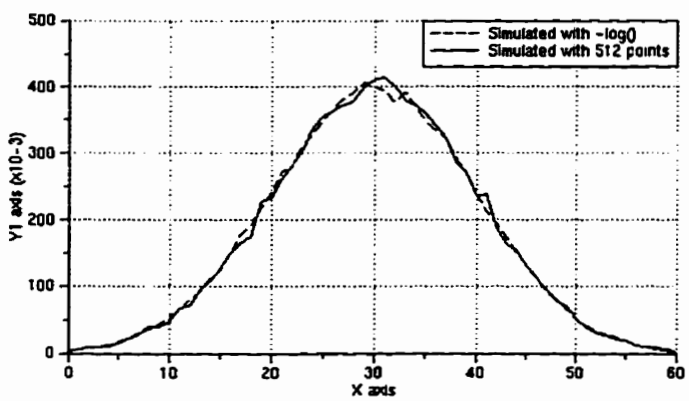

Figure 17. Simulation result of the implementation in Figure 14.

Figure 18 shows the variance-time curve for the generated random sequence. As expected for a normal distribution, the curve takes an approximately linear form. This shows a high

degree of burstiness at fine resolutions, with decreasing burstiness as the time scale increases.

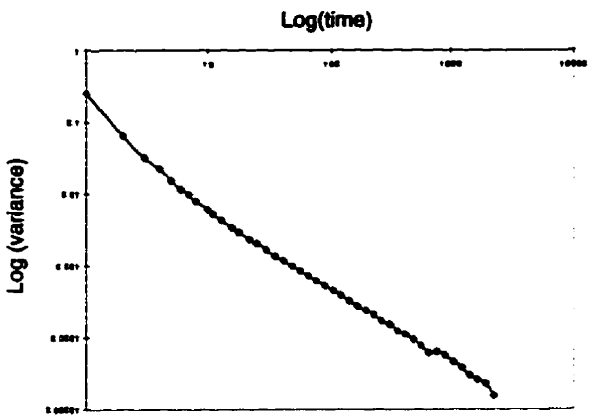

Figure 18. Variance-time curve.

The design shown in Figure 14 can also be downloaded into Filed-Programmable Gate Array (FPGA) chip, for example, the Xilinx Family. As XC4000 FPGA family supports system clock rates of 40 to 50 MHz, this scheme can generate 30M to 38M (*40/C* to *50/C*, $C = \sqrt{(2e)/\pi} = 1.32$) normal random numbers per second on average.

3.4.2 Discussion and comparison

The design in Figure 14 is also compared with several other schemes.

Scheme 1: In this scheme, the subset of *r1* which consists of the failed turns is considered. If let the third uniform random variable *u* be obtained from this subset, *u* is still a uniform variable in (0,1) and is independent of *r1* [38]. Figure 19 presents the simulation result when *u* is obtained from the subset of *r1*.

Scheme 2: The truncated Taylor series can also be considered estimating *log(x)* function of x, that is,

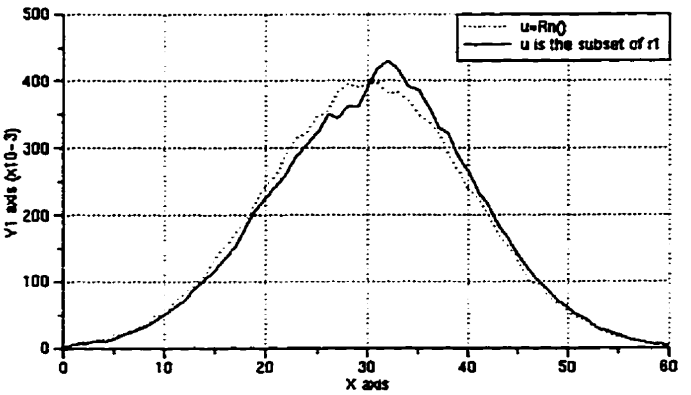

Figure 19. Obtaining u from the subset of r1.

$$log(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 - \ldots \quad 0<x<2 \qquad \text{(Eq-20)}$$

When x is in (0,1),

$$-log(x) = (1-x) + (1-x)^2/2 + (1-x)^3/3 + (1-x)^4/4 + (1-x)^5/5 + \ldots \qquad \text{(Eq-21)}$$

Figure 20 shows the simulation results using the 4^{th} and 5^{th} order truncated Taylor series to approximate $-log(x)$.

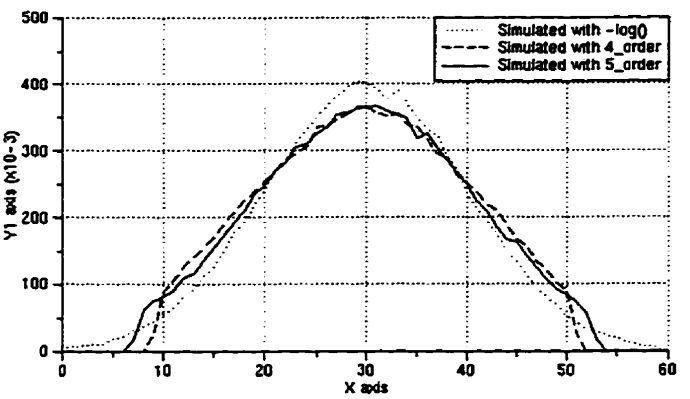

Figure 20. Estimating -log(x) by truncated Taylor series.

Figure 21 displays the outline diagram of the hardware implementation calculating $-log(x)$ function by the truncated Taylor series. Compared to the original design, this scheme needs more clock cycles to generate a normal random number. If more accuracy is expected, more clock cycles are required. In Figure 21, Register #1 is used to accept the

result of the multiplication, the divisors are stored in a ROM, and Register #2 is used to accumulate results. Initially set $A=1$ and $B=0$. After the first step, the output $(1-x)$ is obtained, and after the n^{th} step, the output is:

$$(1-x) + (1-x)^2/2 + \ldots + (1-x)^n/n \qquad \text{(Eq-22)}$$

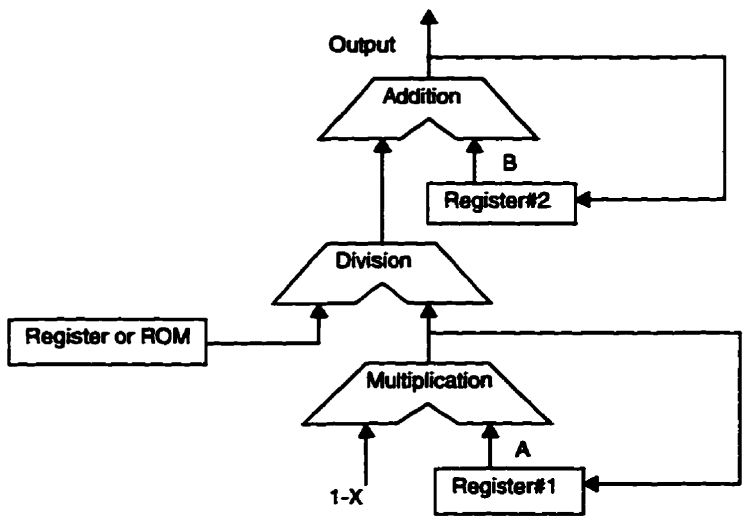

Figure 21. Outline diagram of Taylor series implementation for -log(x).

Scheme 3: Using a piece-wise linear function to approximate $-log(x)$ is also considered.

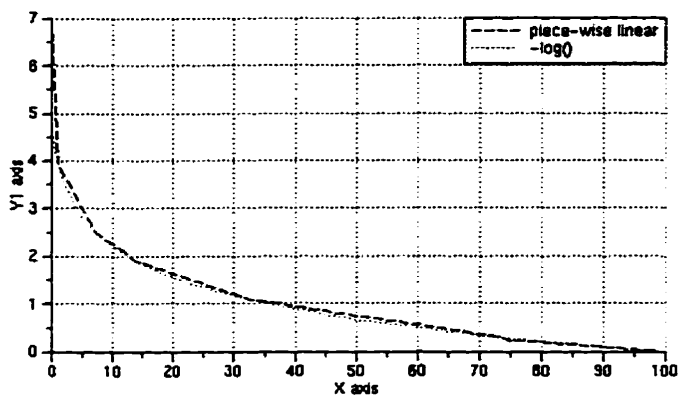

Figure 22. Approximating -log(x) with a piece-wise linear function.

Figure 22 shows a 6-piece linear curve of the piece-wise linear approximation to $-log(x)$ in (0,1), where $x1=0.75$, $x2=0.34$, $x3=0.15$, $x4=0.08$ and $x5=0.02$. Actually, the original

design is a piece-wise linear approximation too, the only difference is that 512 pieces are chosen in that implementation. Figure 23 presents the simulation result of generating a normal random sequence by using a 6-piece linear curve to approximate *-log(x)*.

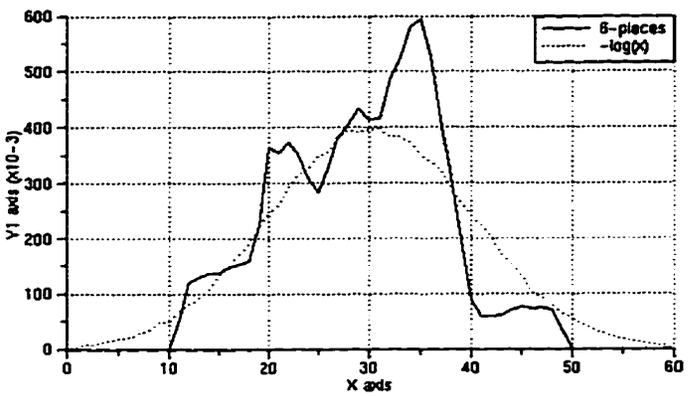

Figure 23. A normal random sequence generated by a 6-piece linear curve.

In general, the original design of Figure 14 is the most efficient among all the schemes considered since this design can produce normal random numbers accurately and quickly.

3.4.3 Conclusions

In this chapter, a 1-D LHCA design for a random pattern generator has been developed; simulation results of using 1-D LHCA to generate a normal random sequence have been presented; several schemes for hardware implementation have been also proposed and compared. The implementation proposed here is efficient and fast. As a general method, it can be used to model ATM traffic sources for evaluating the performance of the ATM networks and switches. The significance of this chapter are: (1) 1-D LHCA is used to generate the required independent uniform random variables; and (2) two ROMs are used to store pre-calculated functions to speed up the process of non-uniform random number generation.

CHAPTER 4 MODELING WITH OPNET

This chapter presents an ATM network model[1] which consists of Ethernet LANs connected to an ATM network. The Ethernet LANs applications are TELNET and FTP. Protocols, such as ATM, TCP/IP and MAC, are used in this model. This chapter also presents the simulation results. The purpose of this study is to investigate (1) how an ATM network works as a whole system; (2) what the network performance is; and (3) how suitable OPNET is as a modeling and simulation tool for ATM networks. Related work can also be found in [39].

4.1 Model description

4.1.1 How does OPNET implement an ATM switch?

ATM layer: The standard ATM switch node is given in Figure 24. It consists of two layers: ATM layer and physical layer. ATM layer is implemented with four processor modules. ATM layer functions are defined in a number of process models which are located in these four processor modules. The physical layer is implemented with a number of point-to-point receiver and transmitter modules. This switch node is used both as the Network-Network Interface (NNI) and the User-Network Interface (UNI). Figure 24-b shows a gateway which is an interface between an Ethernet and the ATM network. The model has

1. This model was designed with OPNET 2.5.B.

three more layers: AAL, IP and MAC.

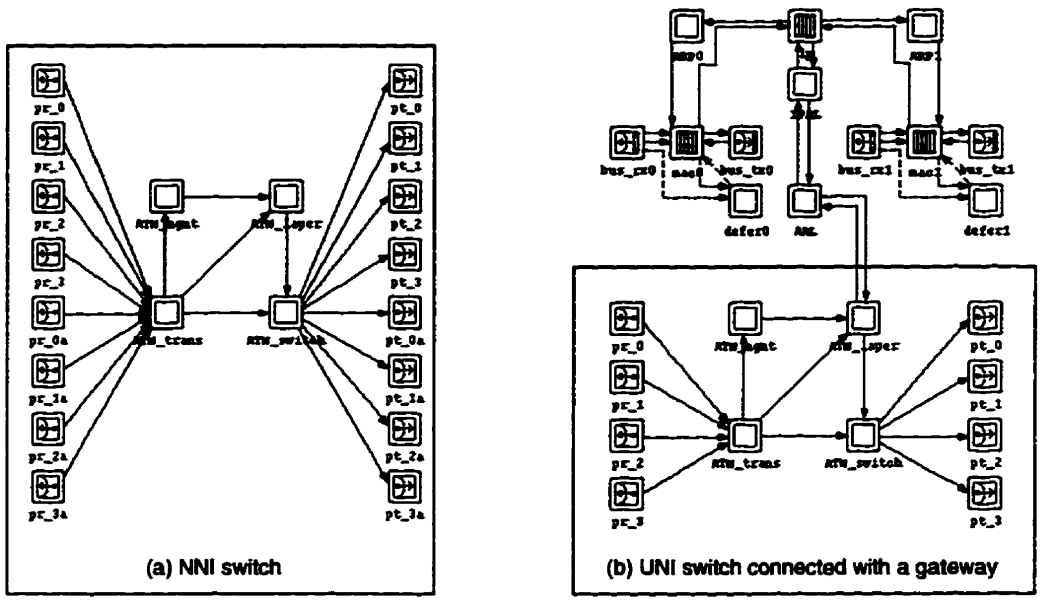

Figure 24. ATM switch node and gateway node.

The node model in Figure 24-a represents an ATM switch which implements VP and VC switching capabilities in an ATM network. It is capable of switching VCCs among eight VP links. It supports the ATM protocol and the port interface consists of eight ATM connections at 155 Mbps. The restriction of this node is that the model can not serve as a source or destination node as it does not perform any AAL functions. A more detailed specification of ATM switch model is given in Table 7.

Because the models are intended for the purpose of simulation, and particularly for performance estimation, certain parts of the protocol have been simplified or omitted [19]. Capabilities implemented for the ATM layer in OPNET are:

1. ATM layer signalling

Dynamic VCC setup and teardown for point-to-point, full-duplex are supported. Point-to-multipoint VCCs are not supported. Status and restart messages are not supported. The

NNI signalling is based on the UNI specification.

Network node	Modules	Process models	Functions
ATM switch	ATM_layer	ams_atm_layer	Encapsulates and forwards data from the AAL layer; decapsulates and forwards data to the AAL layer; forwards cells from the ATM_mgmt module.
	ATM_mgmt	ams_atm_mgmt	Creates and invokes ams_atm_dr to handle call signals and routing messages.
		ams_atm_dr	Builds and updates ATM routing table.
		ams_atm_call_net	Handles signalling to establish and release an ATM connection at a node that is neither the calling nor the called node.
		ams_atm_call_src	Handles signalling at the calling node.
		ams_atm_call_dst	Handles signalling at the called node.
	ATM_trans	ams_atm_trans	Receives incoming cells; translates VPI/VCI values for outgoing cells and forwards cells to appropriate ATM module.
	ATM_switch	ams_atm_sw	Switches cells to output ports.
	pr_x, pt_x		Act as physical input/output ports.

Table 7. ATM switch specification.

2. VPC

VPC specification is based on subscription which is an ATM management function that permits VPCs to be established on a permanent or semi-permanent basis. A VPC is limited to one VP link.

3. VP/VC switching

VP and VC switching are supported. All switching functions are based strictly on the VPI value; the VCI values in VP switched cells are not modified. Input port buffering is not modeled. Output port buffering is modeled. A buffer is available for cells of each QoS class for each output port.

4. ATM traffic control functions

Call Admission Control (CAC) is based on the Peak Cell Rate (PCR) of a call attempting to be established. The CAC algorithm used guarantees that the sum of the PCRs of all

VCCs within a VPC must be less than the total bandwidth of the VPC.

Usage Parameter Control (UPC) monitors cells within a particular flow. If the cells within the flow exceed the traffic contract, actions are taken to prevent the cells from degrading the QoS of other calls. The UPC functions include determining if a cell conforms to the traffic contract as well as discarding non-conforming cells. The conformance definition is based on the GCRA algorithm. The UPC functions are applied only to data cells, that is, cells that do not carry ATM layer signals or routing information.

Procedures for GCRA and UPC are declared in *ams_atm_interfaces.h* and defined in external file *ams_atm_support.ex.c*.

5. Routing

The ATM layer model supports distributed, dynamic routing. The routing table is constructed through a distributed, asynchronous adaptation of the Bellman-Ford shortest path algorithm. Each node in the model maintains its own routing table, which contains records of all known routes to all known nodes for all classes of service. Since there are no public standards for an ATM routing interface, a modular interface is defined that permits a transparent substitution of another routing process within the ATM layer.

AAL layer: The AAL layer is implemented by one processor module as shown in Figure 24-b. The model provides signalling and AAL5 data transfer services. The AAL layer functionality is split into three parts: dispatch, signalling and data transfer. Each part has one or more corresponding process models. Data transfer for AAL5 connections is supported. Data transfer for AAL connections of other types is not implemented. The Segmentation and Reassembly (SAR) function is modeled explicitly in the AAL5 model. The

model supports both simple cell mode and compound cell mode which means that arriving PDUs are sent as a collection of ATM cells. The Cell Delay Variation (CDV) is not computed for ATM cells in compound cell mode.

Packet format: The ATM cell format is defined in *ams_atm_cell.pf.m* by the *Parameter Editor*. The definition of each field is listed in Table 8.

Field index	Field Name	Type	Size (bits)	Default value	Default Set
0	GFC	integer	4	0	set
1	VPI	integer	8	-1	set
2	VCI	integer	16	-1	set
3	PT	integer	3	0	set
4	CLP	integer	1	0	set
5	HEC	information	8		set
6	payload	packet	0		unset
7	payload_bits	information	384		set
8	port	integer	0		unset

Table 8. ATM cell format defined in the standard model.

4.1.2 Assumptions

The network model consists of eleven ATM switches and two types of physical links: T3 (45Mbps) and OC-3 (155Mbps). Assuming that all subnets are Ethernet LANs. The maximum data rate of the LANs is either 15Mbps or 2Mbps. A possible topology of this network is shown in Figure 25. Figure 26 illustrates the relationship between each node according to the layered protocol concepts.

4.1.3 Model specification

Figure 27 shows the OPNET model specified for an ATM network. This network topology reflects an actual ATM network. Ethernet LANs are chosen as sub-networks to model common LAN traffic. They are implemented with a gateway and a number of TCP/IP

workstations. The gateway acts as an interface between Ethernet LANs and an ATM network to support TCP/IP applications over ATM. N60 includes ten workstations as shown in Figure 27. N110 consists of seven workstations. All other LANs have two workstations attached. The data rates of the physical link are defined in a VP configuration table (*my_vp_config.gdf*, see Appendix II). Table 9 lists a more detailed specification of this model.

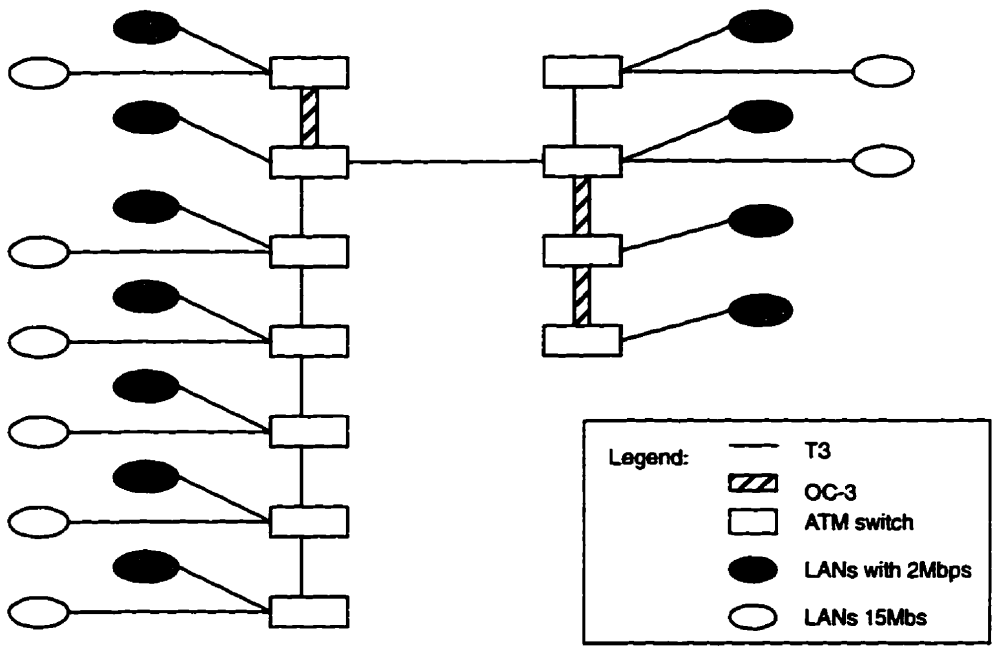

Figure 25. A possible topology.

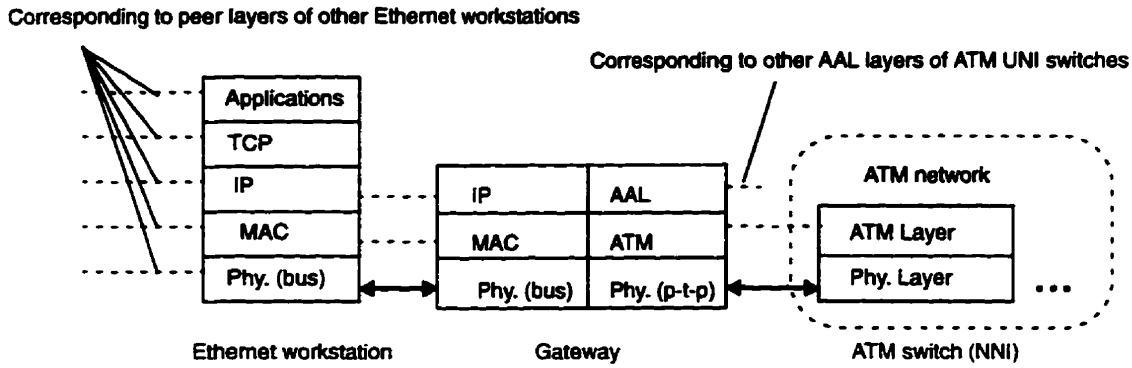

Figure 26. Layered protocols.

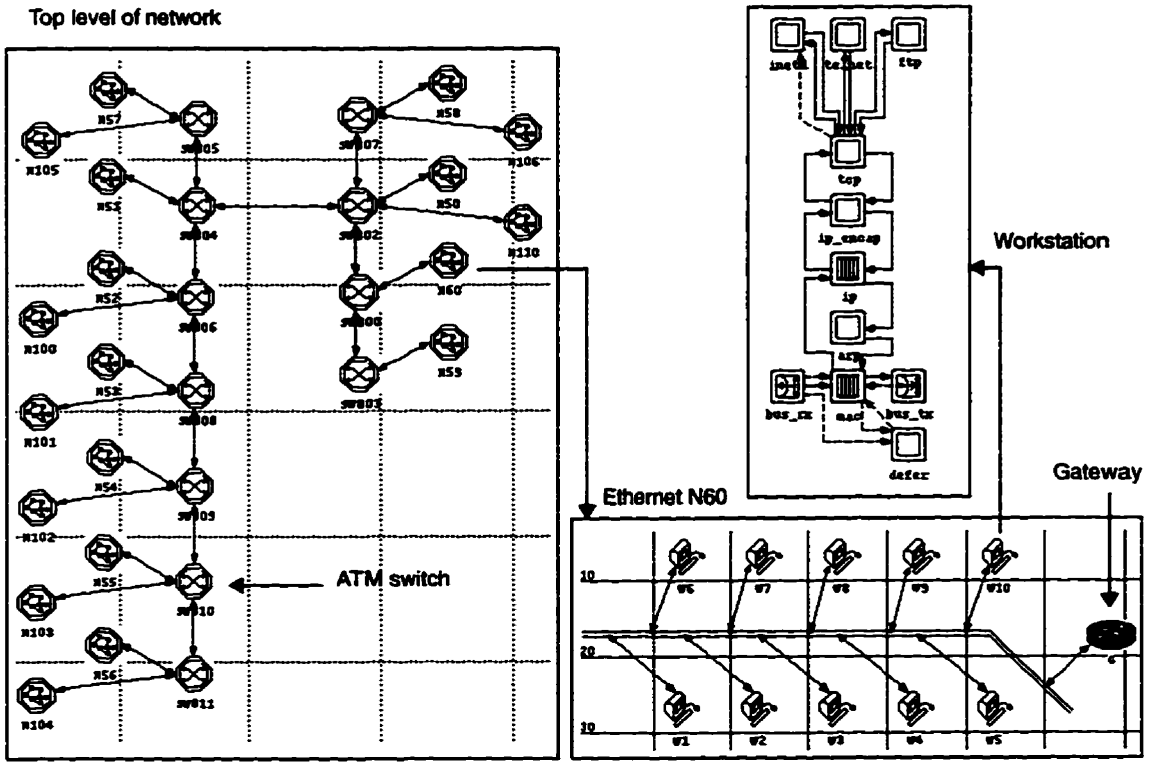

Figure 27. Designed model of an ATM network with Ethernet.

Nodes	Modules	Processes	Functions
Gateway	ATM and Physical layers are the same as in switch nodes		
	AAL	ams_aal_disp	Root process of AAL module; creates & invokes ams_saal.
		ams_saal	Handles all signalling to establish and release an AAL connection.
		ams_aal5_conn	Transports client SDUs for one AAL5 connection.
	IPAL	ams_ipif	Transports IP datagrams across ATM network; establish and releases an AAL connection.
	ip	ams_ip_rte	Routes IP datagrams to appropriate interface; segments and reassembles IP datagrams.
	ARP	ams_ip_arp	Maps IP network and node addresses into lower layer addresses.
	MAC	eth_mac	Accepts data packets from higher layer protocols; encapsulate this data into Ethernet frames; broadcasts these frames in first-in-first-out order to all other stations attached to the bus.
	defer	eth_defer	Performs carrier sensing and computes the deference variable.
	bus_rx, tx		Act as physical bus taps.
Workstation	ip, ARP, MAC, defer and physical layers are the same as in the gateway node		
	ip_encap	ip_encap	Encapsulates higher level data into IP datagrams; decapsulates higher level data from the IP datagrams
	tcp	tcp_manager	Root process of the TCP module; manages a set of connection processes and invokes tcp_conn process

Table 9. Model specification.

4.2 Simulation

4.2.1 Experiment organization

In the experiments, the duration of each simulation is chosen as 20 seconds. The packet size is a uniform distribution in range (a, b) with *a=100* and *b=1000* for TELNET; and *a* and *b* are defined in *att1_pkt.ef* and *att2_pkt.ef* files for FTP (see Appendix II). The queue capacity of the IP module in the gateway node is specified as the default value (infinite) or 32,000 bits and 1,000 packets for the bit and packet capacity, respectively. Specifying that W1 of N5x (x=0,..., 9) and N10y (y=0,..., 6) send packets to W1 of N60 and N110; while Wx of N60 and Wy of N110 send packets to N5x and N10y, respectively. The starting and end time for transferring packets are defined in two environment files: *att1_time.ef* and *att2_time.ef* (see Appendix II).

Figure 28 illustrates the experiments briefly. In both experiments, the starting times are specified from 0 to 10 seconds for the 2Mbps LANs, and from 11 to 18 seconds for the 15Mbps LANs. For instance, an application in N56 sends a packet to an application in N60 at the 6^{th} second after the simulation starts. In Experiment I, all other applications in network other than lower part (below the dash-dot-dot line) shown in Figure 28 are specified to send packets after the 20^{th} second, and are ignored since they do not influence the simulation results in 0-20 second simulation period. In Experiment II, in addition to the specification in Experiment I, N60 and N110 are specified to send packets at 0 and 11 second, respectively. Therefore, at 0 and 11 seconds, the packets being transferred are (11 and 8 times) more then that in Experiment I. This different specification will lead to the different simulation results which will be presented in the next section.

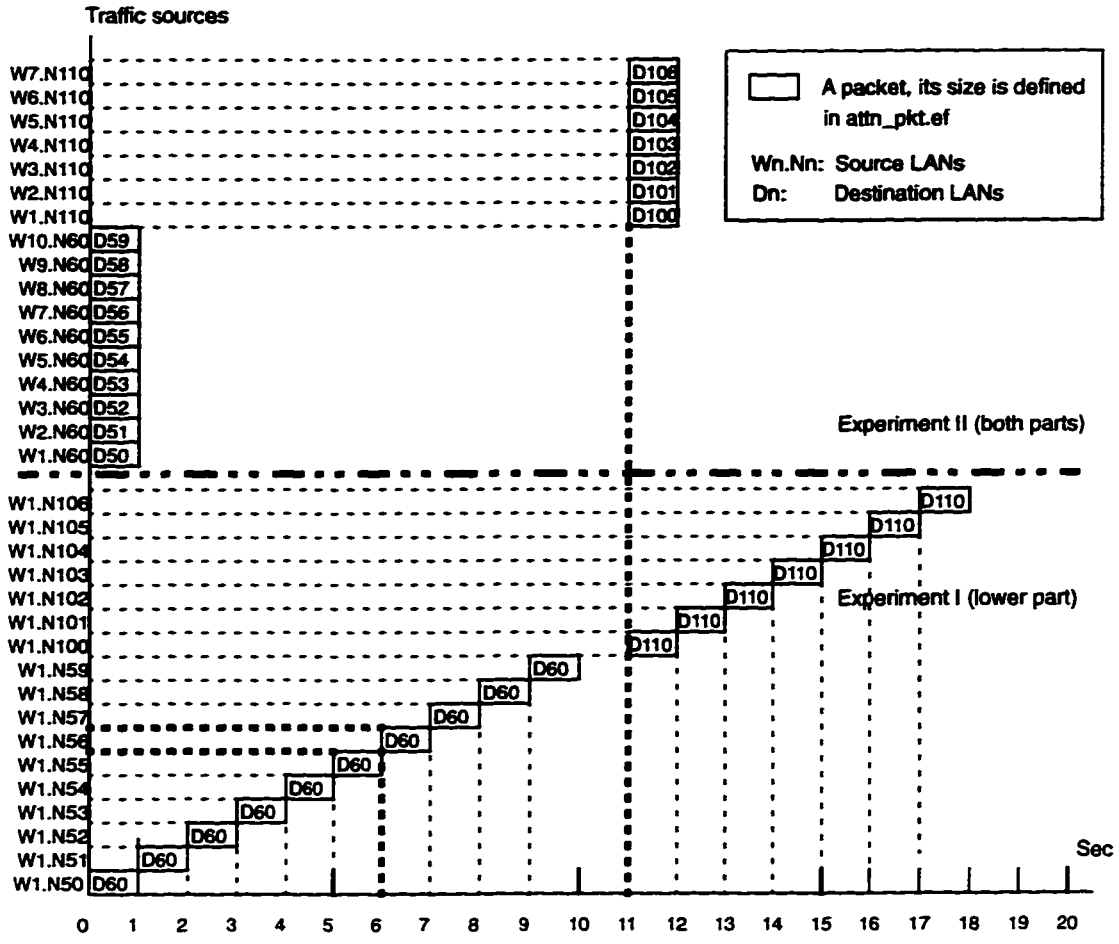

Figure 28. Explanation of experiments.

4.2.2 Collected statistics

Collected statistics are classified as either global or local statistics. The difference between them is that for local statistics, the data source for the statistic output vector is a particular module; whereas for global statistics, all the modules in the particular layer contribute to the same output vector [19]. Four parameters: *Global AAL End to End Delay*, *Global AAL Delay Variation*, *Global ATM End to End Delay*, and *Global ATM CDV*, are to be collected and presented in this chapter.

As an example, the *end to end delay* on AAL layer is computed as the difference between

the time the AAL PDU is created at the source AAL, and the time it is reassembled at the destination AAL, which is shown in Eq-23. Delay variation is the variation in end to end delay. These delay statistics are collected both globally and locally in the *ams_aal5_conn.pr.m* process model.

$$ete\_delay_{AAL} = t_{sAAL} - t_{cAAL} \quad \text{(Eq-23)}$$

Where t_{sAAL} is the current simulation time and is obtained by one of KPs, *op_sim_time()*; t_{cAAL} is the time when the PDU is created at the source AAL and is obtained by another KP, *op_pk_creation_time_get()*. End to end delay (ete_delay) in Eq-23 is updated whenever the *ams_aal5_conn.pr.m* process model is executed.

On the ATM layer, two statistics, the end to end delay and CDV are collected. The end to end delay is computed as the difference between the time ATM cells are sent from the source ATM layer module, and the time they arrive at the destination ATM layer. CDV is the variation of the end to end delay. ATM layer statistics are obtained by similar KPs, but they are collected in process model *ams_atm_layer.pr.m* and external file *ams_atm_support.ex.c*.

4.2.3 Simulation results

Global statistics: After the model specification, models can be simulated and data of the simulation results can be obtained. Two experiments have been done to collect the global statistics. As described in earlier section, more packets are specified in Experiment II. The global statistics obtained are shown in Figure 29 and Figure 30. The values on graphs mean approximate maximum values. For example, the approximate maximum end to end delay on AAL layer is 0.015 second in Experiment I, and 0.15 second in Experiment II.

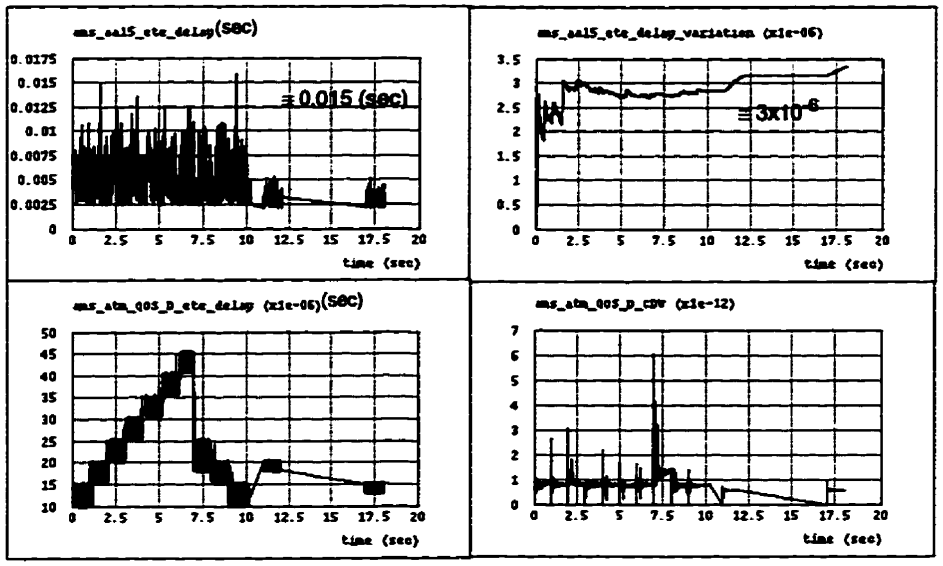

Figure 29. Global statistics obtained from Experiment I.

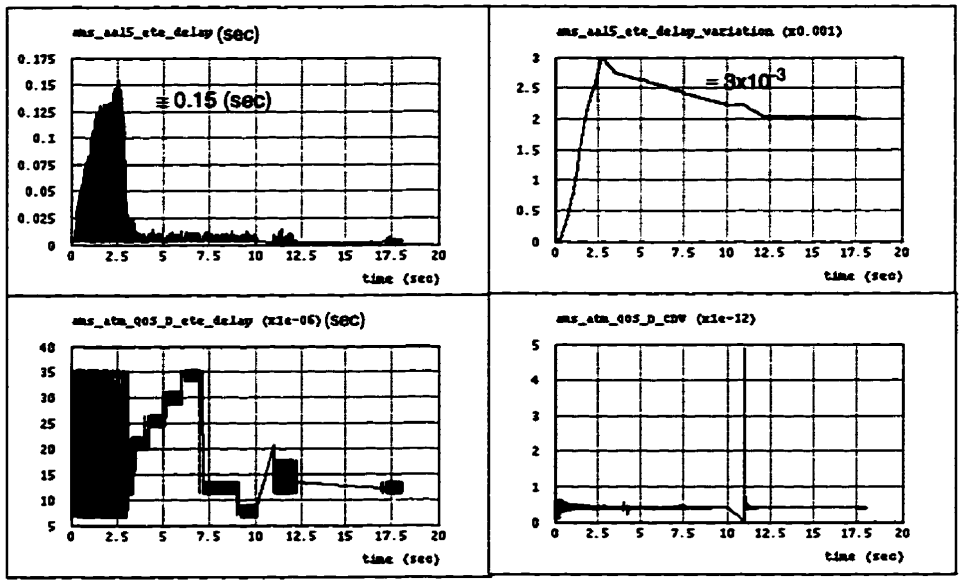

Figure 30. Global statistics obtained from Experiment II.

Statistics show that: (1) with the increase of the packet size, the end to end delay on both AAL and ATM layers are increasing; (2) the shape of the end to end delay and the cell delay variation on the AAL layer and ATM layer in either experiment are different because of the different layer functions; (3) the cell delay variation on the AAL layer in

experiment I is smoother and smaller than that in experiment II, but the CDV on ATM layer in experiment II looks smoother and smaller; also, (4) the end to end delay on the 2Mbps LANs (0 second) is grater than that on the 15Mbps LANs (11 second). This makes sense since the bandwidth of the 15Mbps LANs (15Mbps) is greater than that of the 2Mbps LANs (2Mbps) and the number of 15Mbps LANs (8 LANs) is smaller than that of the 2Mbps LANs (11 LANs).

Local statistics with respect to T3 and OC-3: Creating a probe model in terms of T3 and OC-3 (*my_pb_t3oc3.pb.m*), the utilization, bit-throughput and delay over T3 and OC-3 can be obtained as shown in Figure 31. Similarly, the values on the graphs mean approximate maximum values. For instance, the approximate maximum bit-throughput is 1.75×10^8 bits over T3 and 2.4×10^8 bits over OC-3.

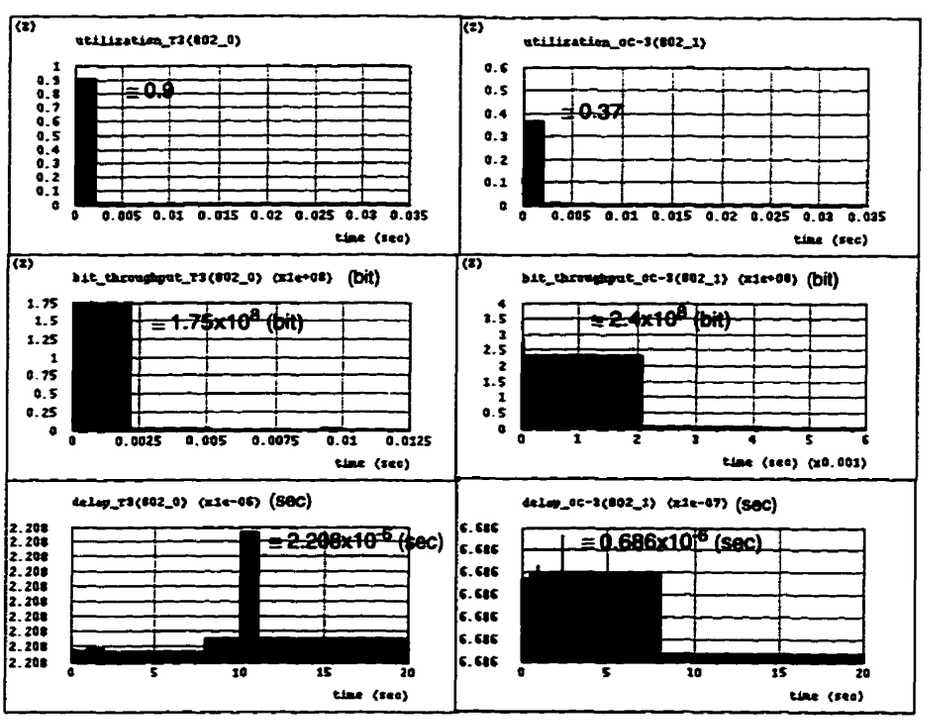

Figure 31. Utilization, bit-throughput and delay over T3 and OC-3.

The collected statistics show that: (1) the bit-throughput over OC-3 is greater than over T3; (2) the delay on OC-3 is smaller than the delay on T3; and (3) the utilization of OC-3 is smaller than the utilization of T3 because the bandwidth of OC-3 (155Mbps) is greater than the bandwidth of T3 (45Mbps).

Local statistics in terms of 2Mbps and 15Mbps LANs: In the simulation results shown in Figure 32, the IP module queue capacity is defined as the default value (infinite). The statistics show that: (1) the delay of 15Mbps LANs (0.01 second) is smaller than the 2Mbps LANs (0.15 second) since the bandwidth of 15Mbps LANs is greater than the 2Mbps LANs; (2) there are no overflow conditions since the queue capacity is infinite.

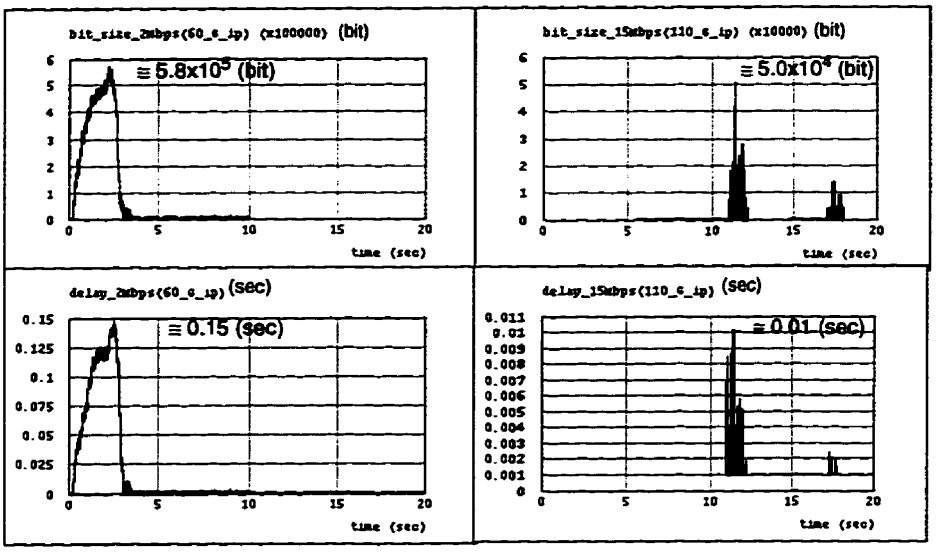

Figure 32. Statistics on the IP queue when the queue capacity is infinite.

In the simulation results of Figure 33 and Figure 34, the IP queue capacity is specified as: 32,000 bits and 1,000 packets. Comparing the statistics in Figure 33, Figure 34 with Figure 32, it can be seen that the overflow occurs. In Figure 32, the bit-size (the packet size measured in bit) can reach up to 580,000 bits while in Figure 33 and Figure 34 the

bit-size can not exceed 32,000 bits, otherwise overflow occurs because of the limited queue capacity.

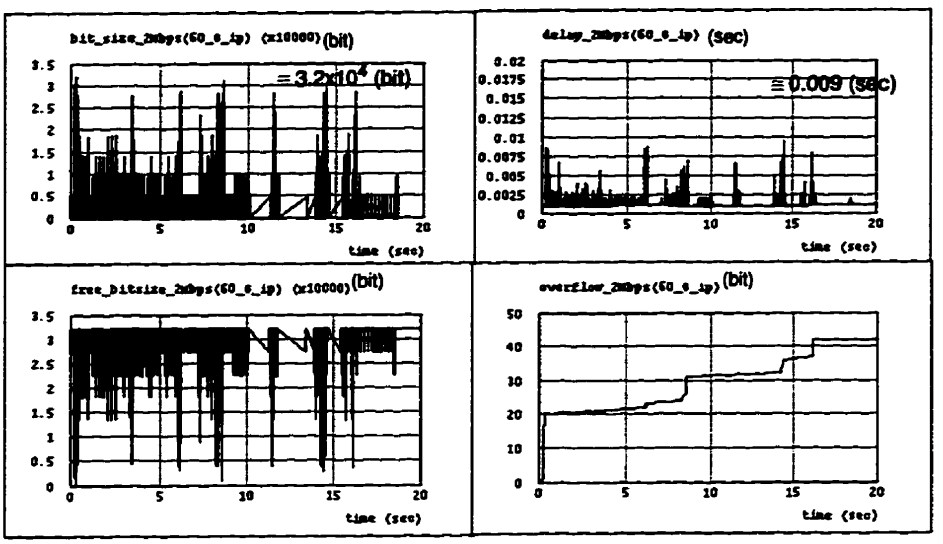

Figure 33. Statistics on 2Mbps LANs when queue is not infinite.

Comparing Figure 32 with Figure 33, the delay on 2Mbps LANs decreases from 0.15 to 0.009 seconds, because the bit-size decreases which is about 580,000 bits in the first case (Figure 32) and 32,000 bits in the second case (Figure 33).

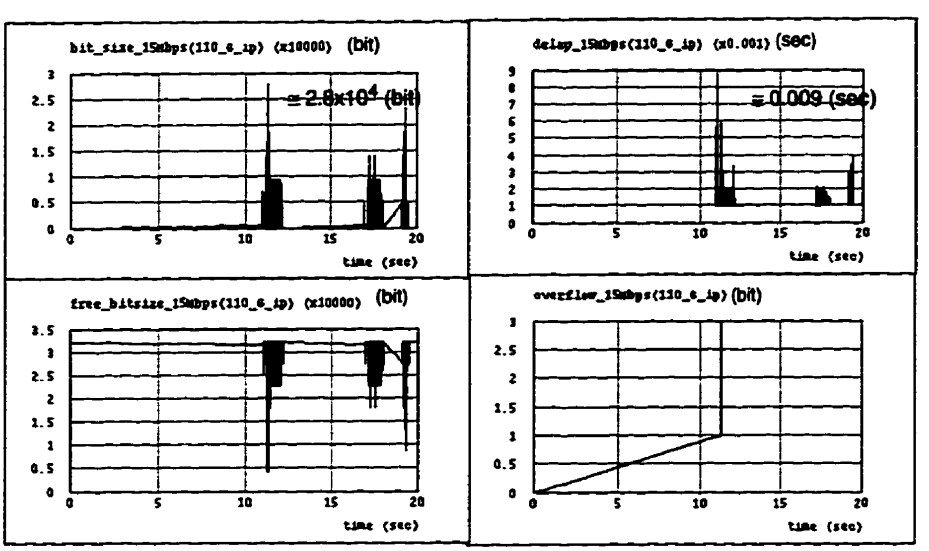

Figure 34. Statistics on 15Mbps LANs when queue is not infinite.

Comparing Figure 32 with Figure 34, delays on 15Mbps LANs are similar, because the bit-size are similar in both cases.

4.3 Summary

In this chapter, an ATM network has been designed and implemented with OPNET. The model consists of eleven ATM switches and nineteen Ethernet LANs. The applications attached to Ethernet LANs are TCP/IP based workstations. The purpose of the model is to gain a general idea how to use OPNET to model, simulate and analyze an actual ATM network.

CHAPTER 5 MODELS FOR TRAFFIC MANAGEMENT

This chapter presents a number of models[1] for three groups of simulations. Because the focus is on investigating traffic management in ATM networks, only client-server systems are considered. The chapter consists of three sections: Section 5.1 describes the model specification, traffic arrivals of five types of applications as well as the parameters and statistics to be collected. Section 5.2 presents three groups of simulations: the first group studies the general performance of the client-server network models, the second examines the limitations of OPNET for modeling and simulation, and the third investigates congestion - the most difficult and challenging problem in ATM networks. Finally Section 5.3 draws conclusions.

5.1 Client-server network models

OPNET provides a large number of standard models for modeling protocols and communication systems. Only client-server network models are considered in this chapter since the goal is to build models for traffic management and the client-server model can model five types of common applications which can generate CBR, VBR and bursty traffic.

1. These models were designed with OPNET 3.0.A.

5.1.1 Model specification

Models constructed in this chapter are modeling WAN networks since the geographic positions of objects are considered. An application of the client-server network is a general model. Its behavior can be modified through parameters and configuration attributes in order to make it act like a wide variety of network applications, including text or graphical virtual terminal sessions, database applications and file transfer utilities. It does not model in detail the behavior of any particular application. All traffic between clients and servers is organized into sessions. A client-server system consists of three types of nodes: server, client and switch. An example of the client-server network is shown in Figure 35. This model consists of twenty clients, one server and five ATM switches. Switches are connected in a star topology.

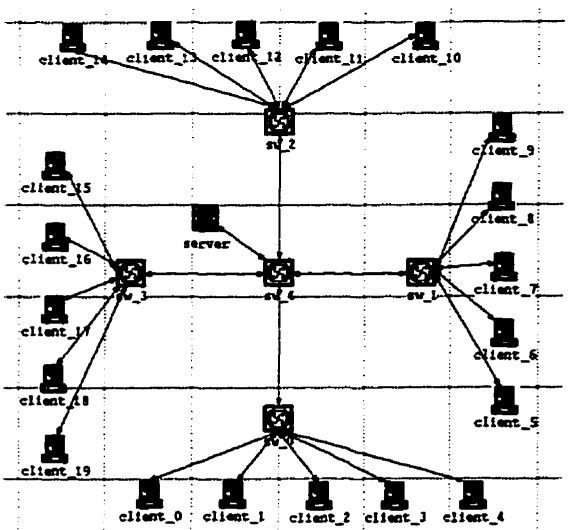

Figure 35. A client-server model example.

Server: The server node modeled in Figure 36-a represents an ATM station with server applications directly using AAL5. It supports ATM and AAL protocols. The parameter, *server configuration table*, is used to specify application servers running on the node.

Another parameter, *transport address*, allows for specifying the address of the node. The server module of the server node is implemented with process model, *net_app_serv*. This process accepts and processes requests from clients at a processing rate that can be specified by the user. A request may ask for zero, or one, or more response packets to be sent from the server to the client. Each response requires a finite amount of server processing time before it can be sent, thus the *net_app_serv* process models a single shared processor processing responses. Processing time is related to the packet size, thus larger response packets require more time to complete. Only one response may be in progress at a time for a single session, but if a server has several active sessions, each of them may have a job in progress. All responses from a server take longer to complete if more than one job is in progress at a time.

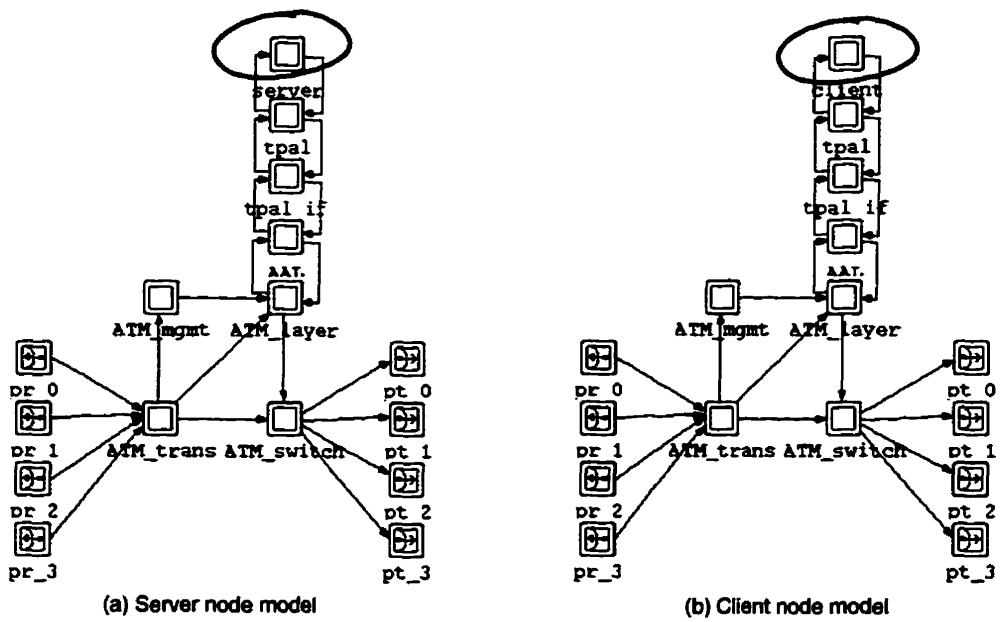

Figure 36. Structure of server and client models.

Client: Figure 36-b represents a client node model. The client attributes, such as *Email*, *Ftp*, *Remote Login*, *X Windows*, *Video Conferencing* and *start time*, allow for the specifi-

cation of application traffic generation in the node. Similarly, the *transport address* attribute allows for the specification of the node address. The difference between these two nodes is the process model implemented on the application layer. The client node is implemented with *net_app_mgr*. The restriction of these two nodes is that the client and server for the same application must not be running in the same node.

5.1.2 Traffic arrivals of applications

Pre-defined distributions: In the OPNET standard client-server model, traffic arrivals are defined by a number of variables in the process model, *net_app_mgr*. These variables represent the interarrival and duration time for the generating sessions and lull/burst periods, as well as the packet size of applications, etc. Table 10 lists these variables for different applications.

Variables	E-mail (sending)	FTP	Remote Login	X Windows	Video Conferencing
sess_ia_dist_ptr	0	exponential	exponential	exponential	exponential
sess_dur_dist_ptr	infinite	1	normal	normal	normal
sess_info.lull_dur_dptr	infinite	infinite	exponential	0.1	infinite
sess_info.lull_pk_ia_dptr	exponential	infinite	exponential	exponential	1/frame_size
sess_info.burst_dur_dptr	0	0	exponential	infinite	0
sess_info.burst_pk_ia_dptr	infinite	infinite	exponential	exponential	infinite
sess_info.pk_size_dptr	normal	512	normal	normal	frame_size
sess_info.resp_size_dptr	16	normal	normal	0	frame_size
sess_info.resp_count_dptr	1	1	poisson	0	1

Table 10. Variables used to define traffic arrivals in standard model.

In Table 10, the term *exponential* (or *normal* or *poisson*) means that the variables in the associated rows are of the exponential distribution with one (or two) argument(s). Note that, in Table 10, the arguments in terms of the same distribution are defined with different values in most cases, depending on the applications. *op_dist_load()*, one of the KPs, is used to load the desired pre-defined distribution [24]. The syntax for calling this KP is

given by:

op_dist_load(dist_name, dist_arg0, dist_arg1)

For example, to obtain an exponential variable with mean value *1/a*, the Proto-C code should be written as:

op_dist_load("exponential", 1/a)

In this line, the first argument, *dist_arg0*, represents the mean outcome *1/a*, and the second argument is ignored for the exponential PDF. In OPNET, the exponential PDF is defined as given in Eq-24, where $a>0$, $E(x)=a^{-1}$ and $\sigma^2_x=a^{-2}$.

$$f_x(x_0) = \begin{cases} ae^{-a \cdot x_0} & x_0 > 0 \\ 0 & otherwise \end{cases} \quad \text{(Eq-24)}$$

As mentioned earlier, traffic between clients and servers is organized into sessions. Some applications have only one or two sessions, others have a random number of sessions. During a session, the client alternates between periods of low traffic (lull periods) and periods of high traffic (burst periods). The similarity between the lull and burst period is that in both periods, two variables, the packet interarrival time (time) and the packet size (size) are defined in the same way. The difference between them is that the packet interarrival time in burst period is defined smaller and the packet size in burst period is larger. In other words, if the random distributions and the arguments for above two variables (time and size) in the lull and burst periods are defined the same, then there is no difference between lull and burst period.

Email: The email application starts one session for generating email and another for requesting email. Both sessions are open for the entire simulation. The length of time dur-

ing which email continues to arrive is infinite, since it is not expected that email stops permanently at any time. Due to the fact that email transfers are discrete, there are no periods of lulls and bursts. In the standard model, email is defined in the lull period throughout the simulation and never in a burst state. The *send rate* parameter is used to decide the number of emails being sent per hour, that is, the interval separating email generation. The email rate parameter is defined as a constant in the process model which may be specified by the user. Traffic arrivals of generated email messages are illustrated in Figure 37. The interval time for the email generation (T) is an exponential distribution with a mean value of *send rate*, and the size of each email is a normal distribution. The arrival time for email (Tl1, Tl2...) could be any time after the email is generated.

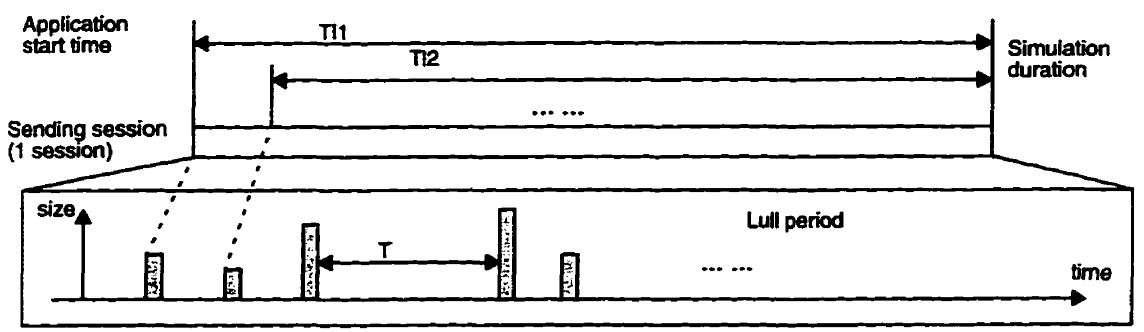

Figure 37. Traffic pattern of generated email.

Ftp: Figure 38 depicts traffic arrivals of FTP applications. FTP has a variable number of sessions depending on the specified *session rate*. Sessions are generated with the exponential distribution based on the *session rate* which is obtained from the attribute specification. For example, Tf in Figure 38 is an exponential distribution with a mean value of *session rate*. The duration of the sessions is defined as 1 second since it only needs to send out one request. Similar to email, FTP will remain in the lull period throughout the simulation duration. The FTP sending interarrival time will be infinite because only one

request is desired. FTP applications use the CLOSE command to request a file (other applications do not). Only one file is returned and the size of the file is determined by the normal distribution with a mean value of *average file size*.

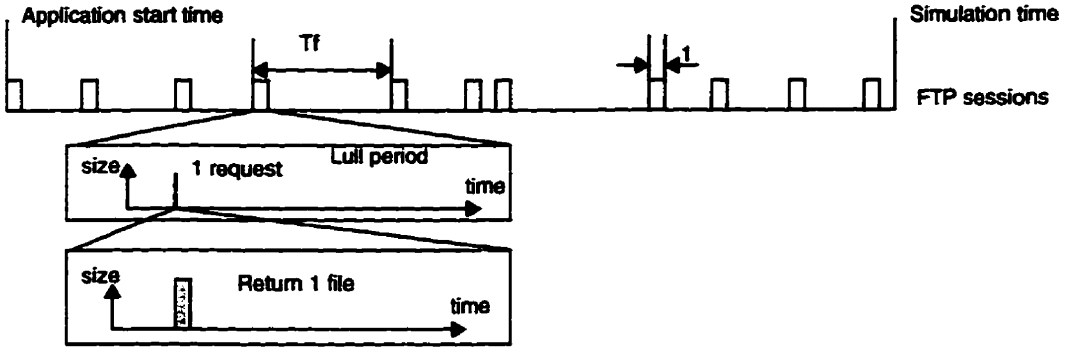

Figure 38. Ftp traffic.

X Windows: There are periods of lulls and bursts for X Windows. At the start-up period of an X Window session, many requests are all made at once. After the start-up period, there is a steady state period, in which requests are made periodically. Thus, the start-up period is modeled as the lull period and the steady state as the burst period. The duration of the lull period is defined as 0.1 second (constant) and the burst period as infinite.

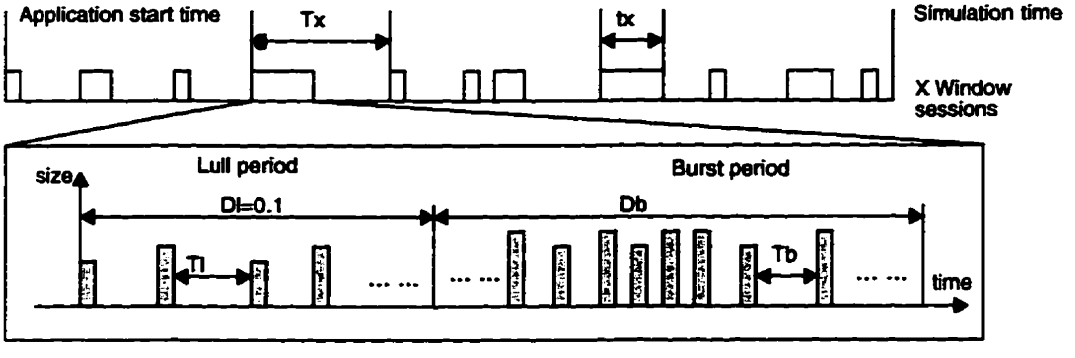

Figure 39. X Window traffic illustration.

Remote Login: There are periods of lulls and bursts for Remote Login. The burst to lull ratio is 1 to 6 on average. The burst to lull packet interarrival ratio is 10 to 1. The number

of sessions of Remote Login (Tr) follows an exponential distribution and the duration of the sessions (tr) follows a normal distribution. Figure 40 illustrates the traffic arrivals of the Remote Login application according to the definition in Table 10. Db and Dl represent the duration of the burst and lull period with Db:Dl=1:6. Tb and Tl represent the packet interarrival time of burst and lull packets and the relationship between them is Tb:Tl=1:10. Db, Dl, Tb and Tl are all determined by the exponential distributions, but with different arguments.

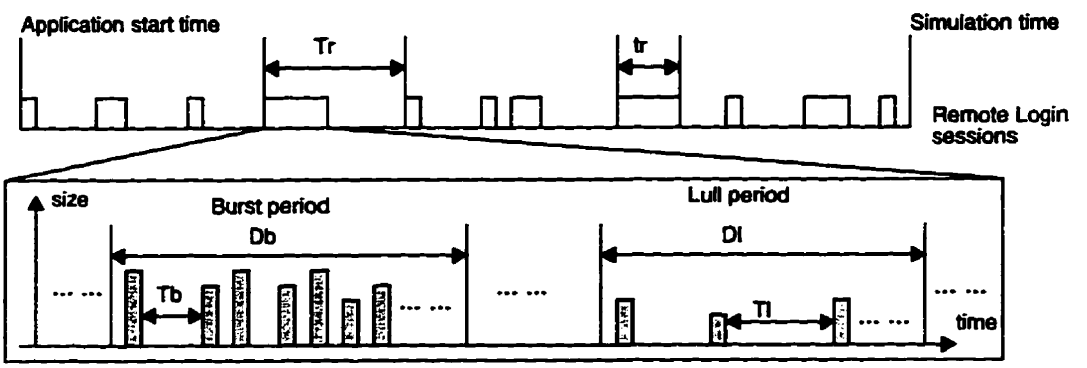

Figure 40. Remote Login traffic.

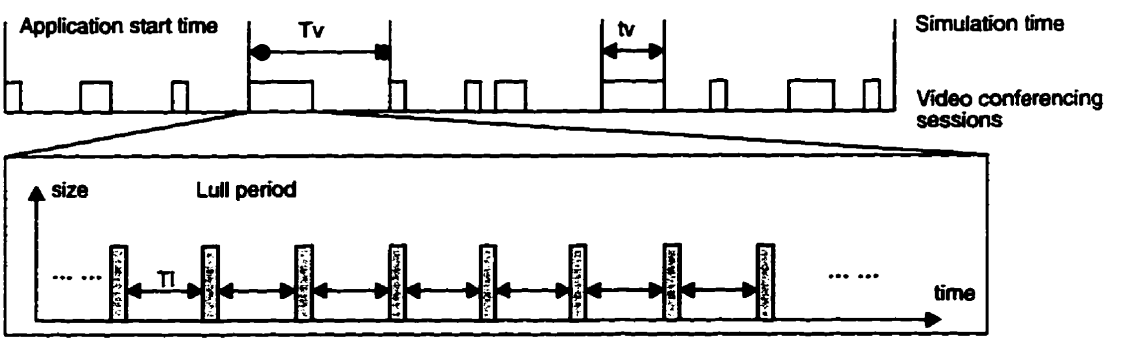

Figure 41. Traffic for Video conferencing.

Video conferencing: The sessions are generated with the exponential distribution (Tv) and the duration of the sessions (tv) with a normal distribution. The burst period never happens and it always acts in the lull period. The interarrival time (Tl) and the frame size

(size) are constants.

5.1.3 Parameters and statistics

The parameters and statistics with respect to the traffic descriptor and QoS are: PCR, CDVT, CLR, CTD, and CDV. In OPNET standard models, PCR is defined in *ams_atm_interfaces.h*. The PCR for the call signalling is defined as 1.2×10^4 cell/sec which is approximately equal to 5Mbps. The PCR for routing cells is defined as 2.2×10^3 cell/sec (≅1Mbps). CDVTs for the four QoS classes (that is, A, B, C and D) are specified in the simulation set before running the simulation. The values are 0.00025, 0.0005, 1.0 and 1.0 for the QoS class A, B, C and D, respectively. By specifying a probe model, CLR, CTD and CDV can be obtained. The global CLR represents the cell loss ratio for all QoS class ATM cells in the network. To obtain this statistics, the number of discarded cells is normalized by the total number of cells sent. Four functions associated with obtaining CLR are declared in *ams_atm_interfaces.h* and defined in the external file, *ams_atm_support.ex.c*. Eq-25 is used to compute CLR.

$$CLR = total\_cells\_discarded / (total\_cells\_sent + total\_cells\_discarded) \quad \text{(Eq-25)}$$

CTD may refer to the AAL layer, or the ATM layer, or to all applications. As introduced in previous chapter, the AAL CTD is computed by Eq-23. To obtain CDV, the attribute, *compound cell*, must be specified as disabled ("0") which allows the model to invoke the Usage Parameter Control (UPC) function. In OPNET standard model, CDV is computed by Eq-26, where S_i is the delay sample and n is the number of samples.

$$CDV = \left(\sum_{i=1}^{n} S_i^2 \right) / n - \left(\left(\sum_{i=1}^{n} S_i \right) / n \right)^2 \quad \text{(Eq-26)}$$

5.2 Simulation and performance analysis

5.2.1 Model performance

Small model: The experiment is organized based on two models: a small model and a large model. The small model is shown in Figure 42. Some attributes are specified in Table 11 (or otherwise use the default setting).

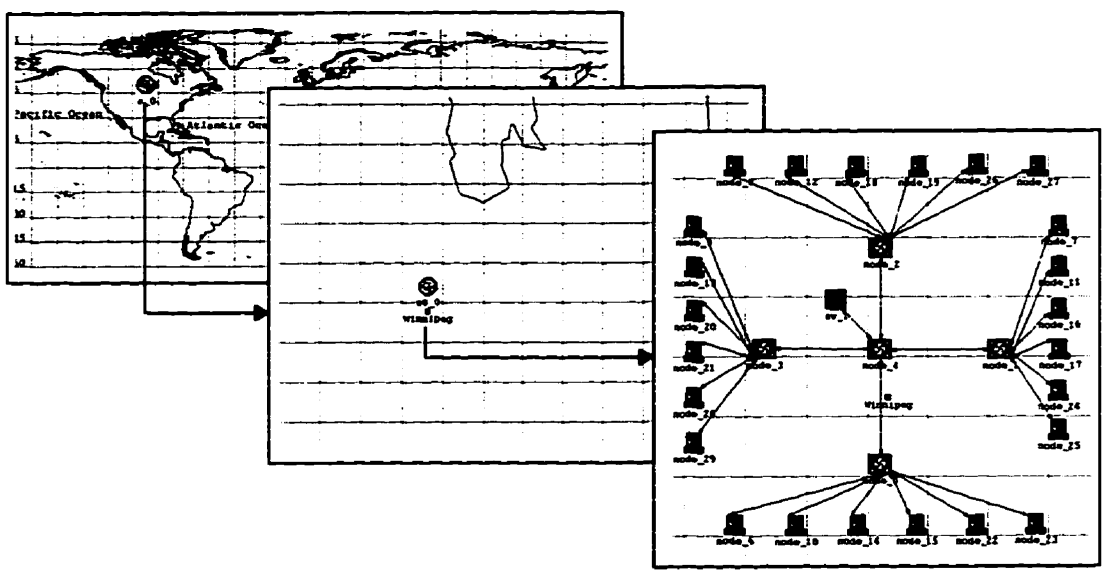

Figure 42. Model with 30 nodes (sec_521_1.nt.m).

Attributes	Switch	Server	Client	Simulation set
ATM QoS D Buffer Capacity	1000 cells	-	-	-
Transport Address	-	Auto Assigned	Auto Assigned	-
Server Configuration Table (name and weight)	-	Email (15), Ftp (10) Remote login (25) X windows (20) Video conferencing (5)	-	-
Start time	-	-	1.0	-
Application server	-	-	Random	-
Application load (L: low, M: medium, H: high)	-	-	Email (L), Ftp (M) Remote login (H) X windows (H) Video conferencing (M)	-
Simulation duration	-	-	-	20

Table 11. Attribute specification.

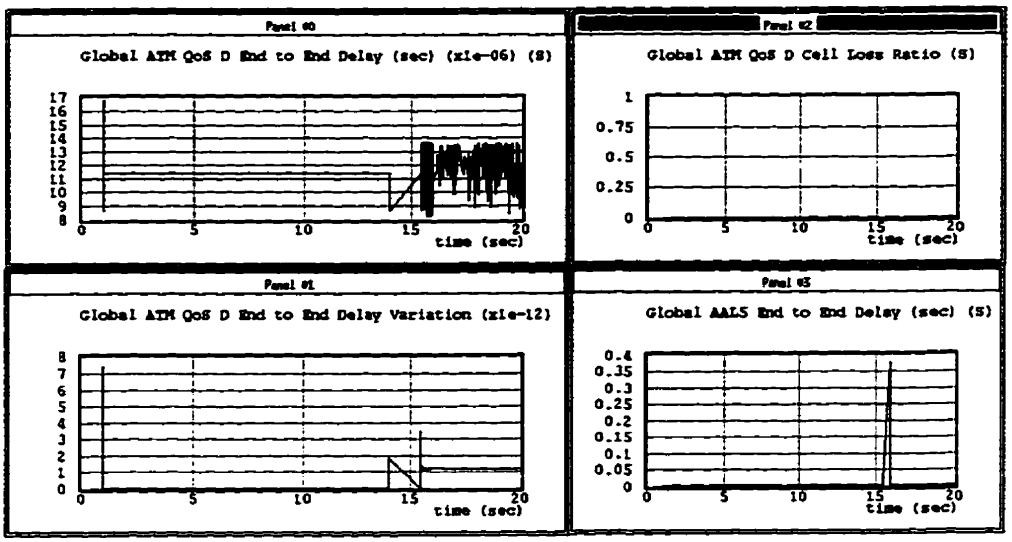

Figure 43. Statistics obtained from model in Figure 42.

Large model: The large model is shown in Figure 44. It is an extension of model *sec_521_1.nt.m*. It consists of 186 nodes, and 35 of them are switch nodes. Traffic sources and destinations are randomly specified.

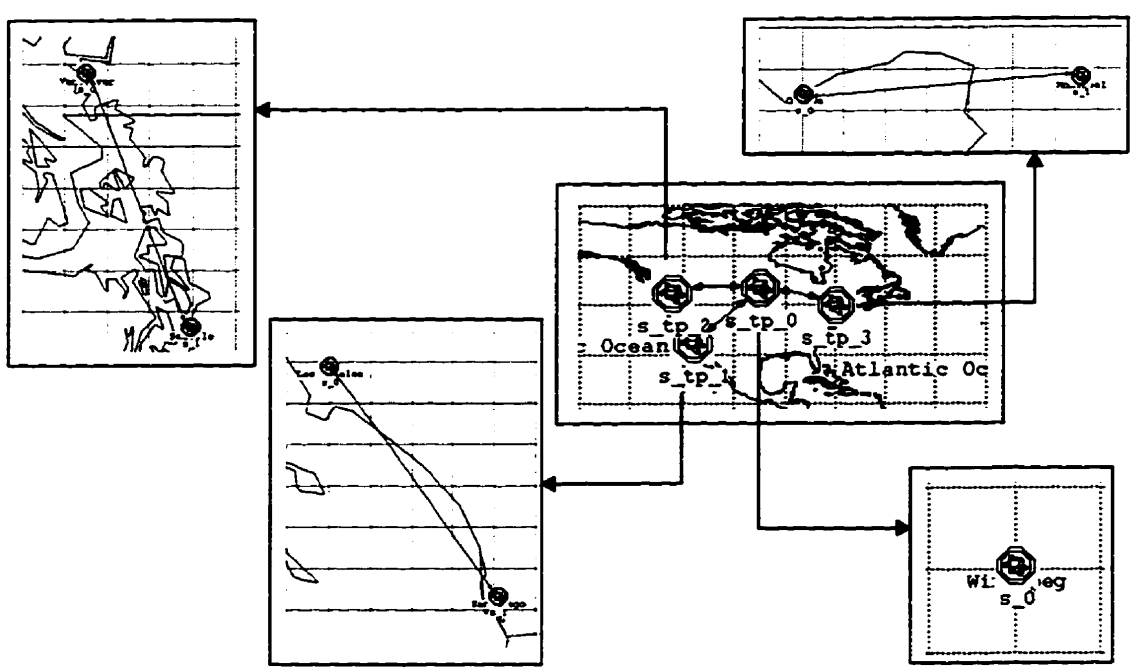

Figure 44. Model with 186 nodes (sec_521_2.nt.m).

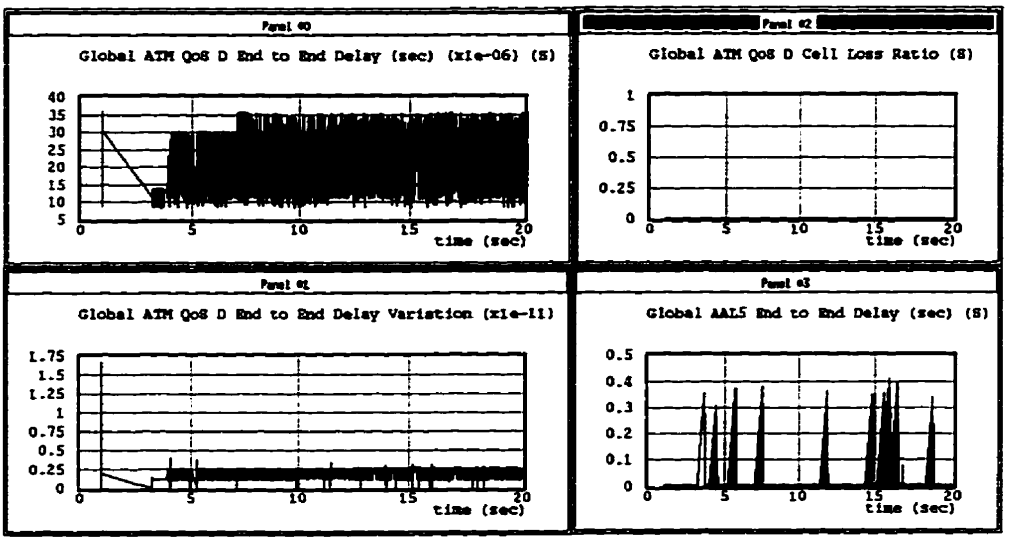

Figure 45. Statistics obtained from model in Figure 44.

Results: Figure 43 and Figure 45 show global statistics obtained from simulating the two models. Results show that: (1) as we expected, the AAL end to end delay is much greater than the ATM end to end delay in both cases; (2) the large model produces larger delays on both AAL and ATM layers; (3) CLR obtained from both simulations are zero, indicating that no cells were discarded. These results bring us some questions, and they will be answered in following sections: Section 5.2.2 and Section 5.2.3.

1. How large a model can be constructed which OPNET can handle?
2. Is OPNET suitable for modeling and simulation of a network with an abnormal situation, for example, congestion?

5.2.2 Simulations with respect to the limitations of OPNET

In this section, models are developed for studying the limitations of OPNET. When evaluating a CAD tool, usually two issues are concerned: memory requirements and processing time. The memory requirements usually refer to both disk space and RAM. Theoretically, an OPNET model can include infinite levels and infinite nodes on each level, memory is allocated as needed and de-allocated when no longer needed [21]. However, this ability is

limited by the computer system (the finite RAM) used for modeling and simulation.

Experiments have shown that (1) at the model specification stage, the number of active sessions affect the memory space and processing time, (2) at the simulation stage, two major factors: the model size and the simulation time affect those two issues. This section examines OPNET limitations at the simulation stage.

Simulations in terms of the model size: The experiments are organized as follows: by using *top* and *ls* commands to obtain information about the process size and the model size, etc. from modeling and simulating a series of models which include the different number of nodes. Table 12 shows how to define the model size, what parameters to be collected and the results. Figure 46 presents one of models (md_10) listed in Table 12. When the number of nodes is 216, the simulation can not be completed, because of the lack of the swap space. According to a large number of experiments (most of them are not shown here), sometimes the process size can reach up to 800 Mbytes, usually in this kind of situation, the free RAM space and free swap space are just several thousand kilobytes.

Models	Number of nodes	Size of process (Mbytes)	Processing time (Sec.)	Model size (Bytes)	Size of output vector (Bytes)
md_1	10	10	20.1	15,649	450,526
md_2	20	11	66.4	30,003	1,432,025
md_3	26	12	66.9	59,312	1,652,366
md_4	36	12	104.1	73,666	2,039,353
md_5	46	13	137.3	88,474	2,384,937
md_6	52	14	205.3	117,329	3,914,537
md_7	72	15	321.2	146,491	5,059,945
md_8	108	19	617.9	219,316	7,745,296
md_9	144	24	1048.6	292,141	12,305,392
md_10	180	30	1653.2	364,966	14,592,816
md_11	216	Abort			

Table 12. Simulation with different model size.

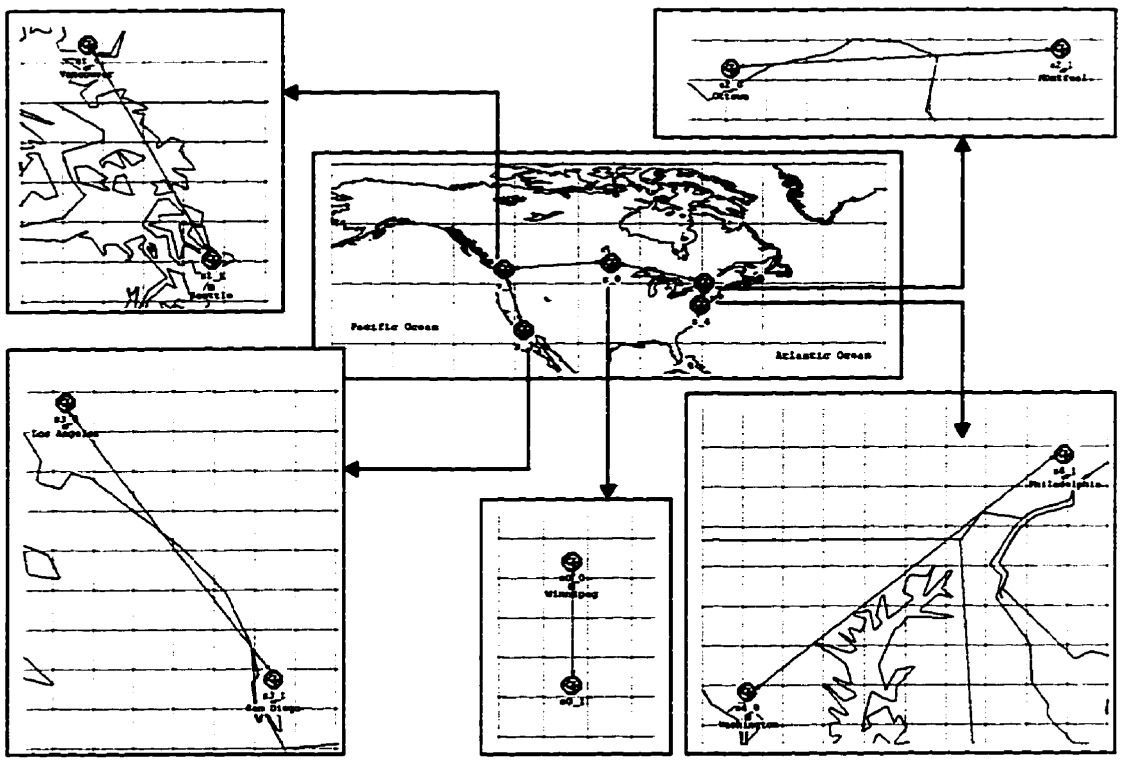

Figure 46. Model with 180 nodes (md_10).

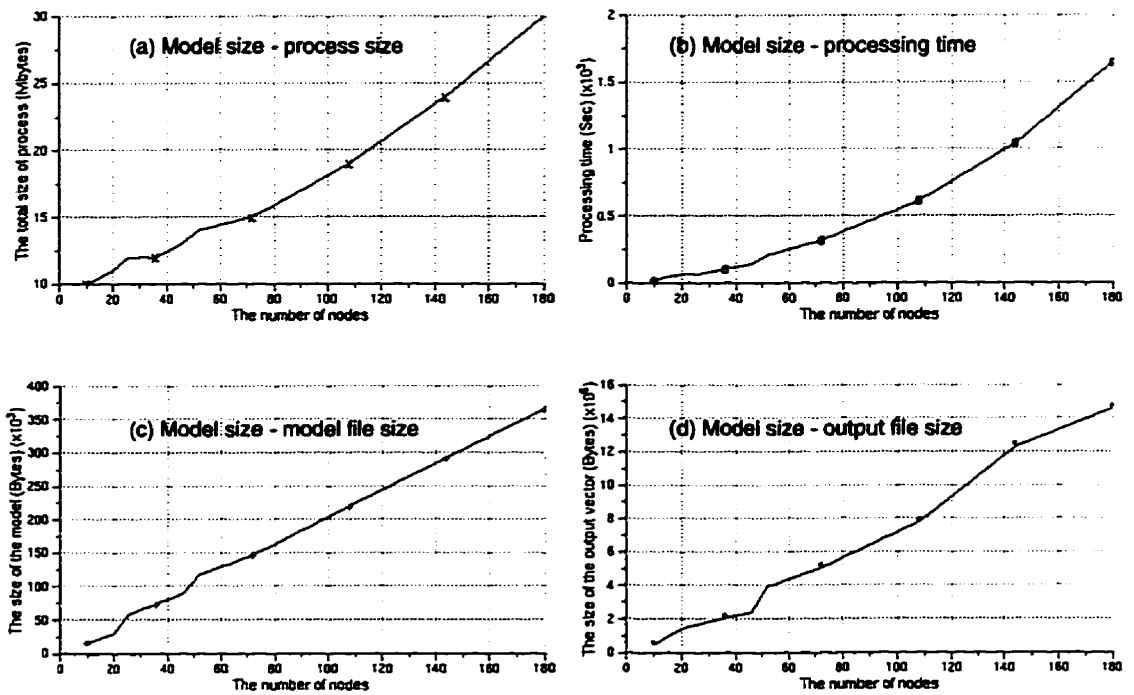

Figure 47. Results about model size.

Figure 47 presents the experiment results with respect to the model size. Basically, an increase in the model size causes an increase in the process size, processing time, the disk space of the model file and the statistic output file.

Simulations in terms of the simulation duration: When using OPNET for modeling, simulation and performance analysis, the disk space is not more crucial than RAM. But when the output vector file reaches a certain size, there is a difficulty to open OPNET windows to view statistics, because running OPNET needs to have sufficient RAM to allocate a data file into the *Analysis Tool*. Table 13 illustrates the simulation performance with different simulation duration.

Simulation duration	Processing time (Sec.)	Size of output vector (Bytes)
300	63.4	1,405,626
600	249.5	5,684,234
900	510.2	11,449,258
1200	936.1	21,559,066
1500	1565.0	34,702,458
1800	2356.0	50,453,114
2100	3120.9	75,010,502
2400	Abort	

Table 13. Simulation with different simulation time.

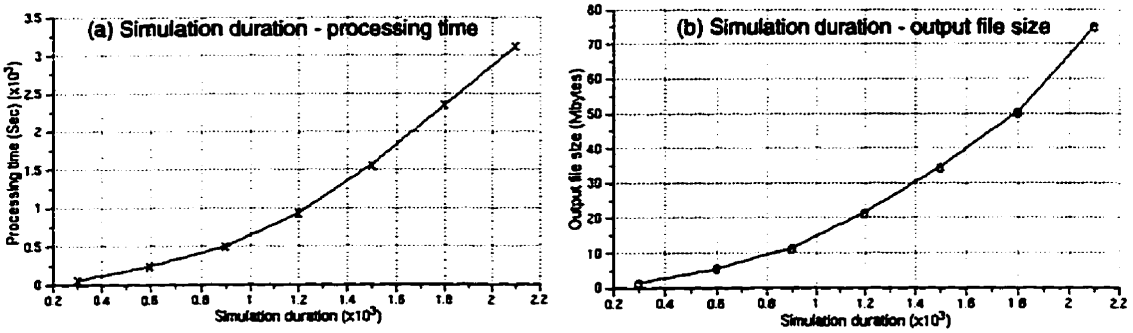

Figure 48. Results about simulation time.

Figure 48 presents the experiment results with respect to the simulation duration. In this

experiment, *md_1* in the previous section is used. The results show that although the process size remains the same, the processing time and the size of the output vector file increase dramatically when increasing the simulation time. Statistics related to the simulation duration are shown in Figure 49 and Figure 50.

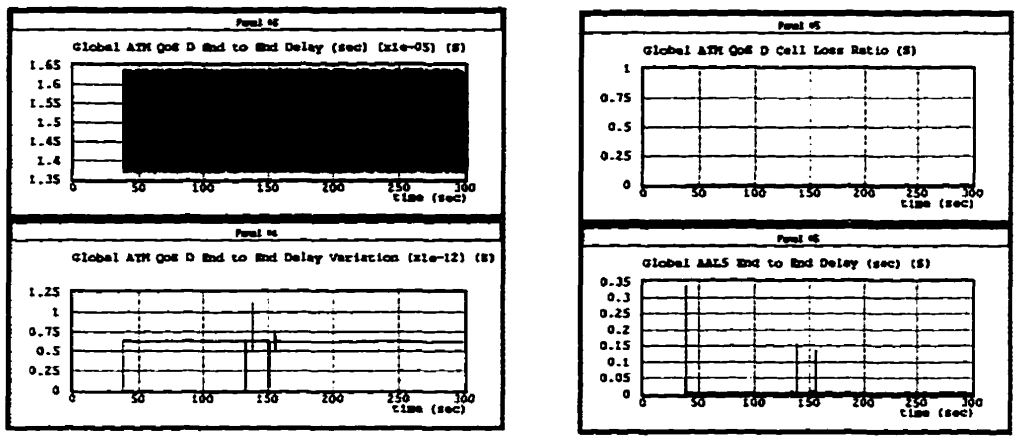

Figure 49. Simulation with 300 seconds (s_300, md_s_1.nt.m)

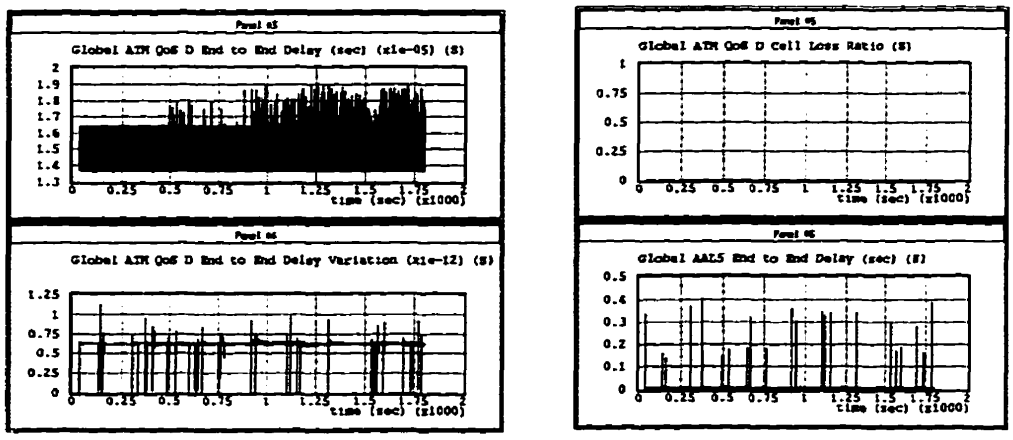

Figure 50. Simulation with 1800 seconds (s_1800, md_s_1.nt.m).

Results: There exist two issues which are in conflict in the course of modeling networks for traffic management purpose. On the one hand, the model should be built to fit the computer ability; on the other hand, the purpose of the model is for traffic management. For

example, obtaining CLR and investigating congestion require a huge number of traffic sources. One solution to ease this problem is to run simulation by the script language. By closing the OPNET windows at least 13Mbytes RAM can be released.

5.2.3 Congestion related simulations

Congestion is a dynamic problem and the imbalance of the network traffic is the main reason for this problem [11][18]. Since traffic of all clients were defined with randomly chosen destinations, there was no congestion in the previous simulations. Figure 43 and Figure 45 show that CLR is zero and throughput is the same as the actual traffic load even for the large model (Figure 44), that is, every packet is delivered successfully. If an imbalance situation can be created, there is a possibility that congestion can occur.

These section investigates congestion under three problems: (1) making an imbalance situation by specifying a solo subnet to communicate with all other subnets; (2) decreasing the buffer size of the switch; and (3) making traffic more bursty.

Making an imbalance: The following simulations use the same model (*sec_521_2.nt.m*, Figure 44) to examine the congestion problem. The difference is that in this experiment, all clients are specified to send the packets to a single destination subnet.

The statistics obtained from this experiment are shown in Figure 51. Statistics show that the ATM end to end delay (10^{-3}) and CDV (10^{-8}) are greatly increased with the imbalance situation as compared with Figure 45 (10^{-6} and 10^{-11}, respectively). CLR, the delay and the delay variation on AAL still remain similar. The throughput and the actual load are almost the same.

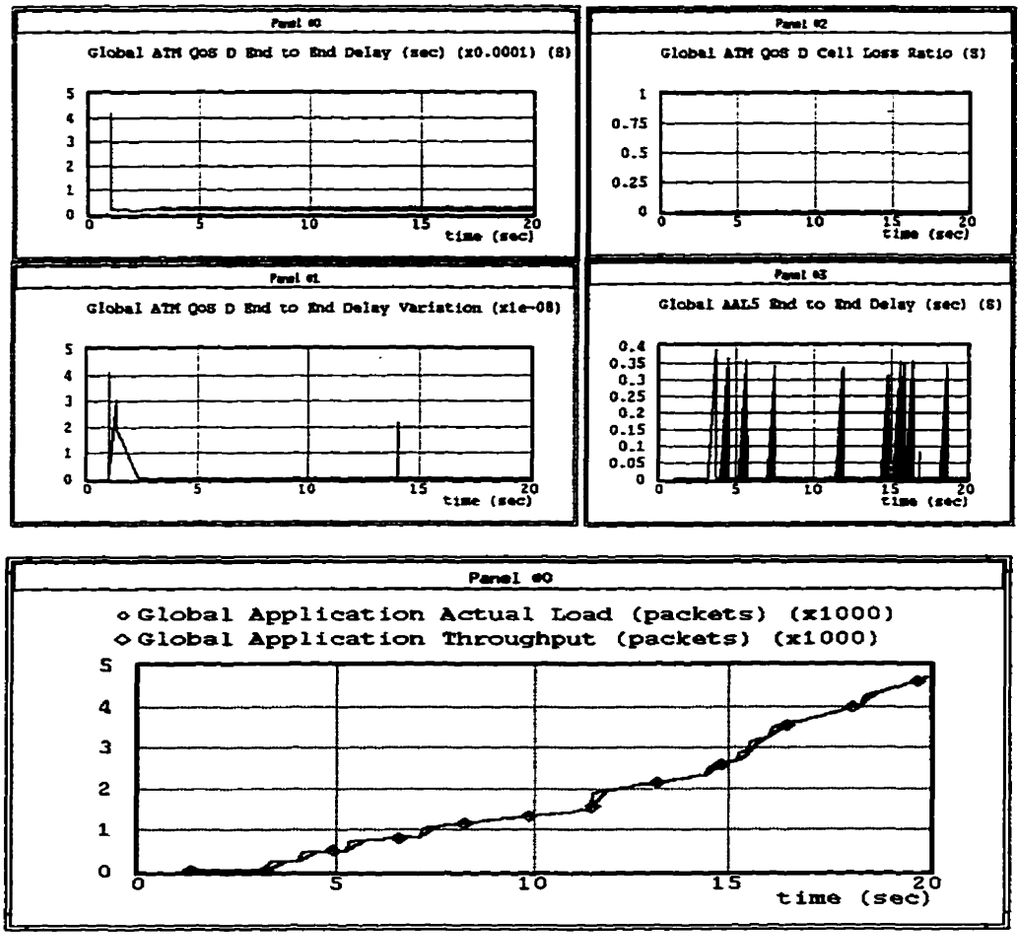

Figure 51. Statistics regarding an imbalance situation.

Decreasing buffer size: In this experiment, the buffer size of switches was decreased from 1000 (the default value) to 100 cells. Two cases are considered.

Case 1: Only decrease the buffer size. Figure 52 represent the simulation results of this case. The simulation results are similar to the results in previous section.

Case 2: In addition to the decrease of the buffer size, an imbalance situation is also created. Figure 53 and Figure 54 represent the simulation results of this case. Statistics in Figure 53 show that CLR is no longer zero in addition to the increase of the ATM delay and CDV (10^{-3} and 10^{-8}).

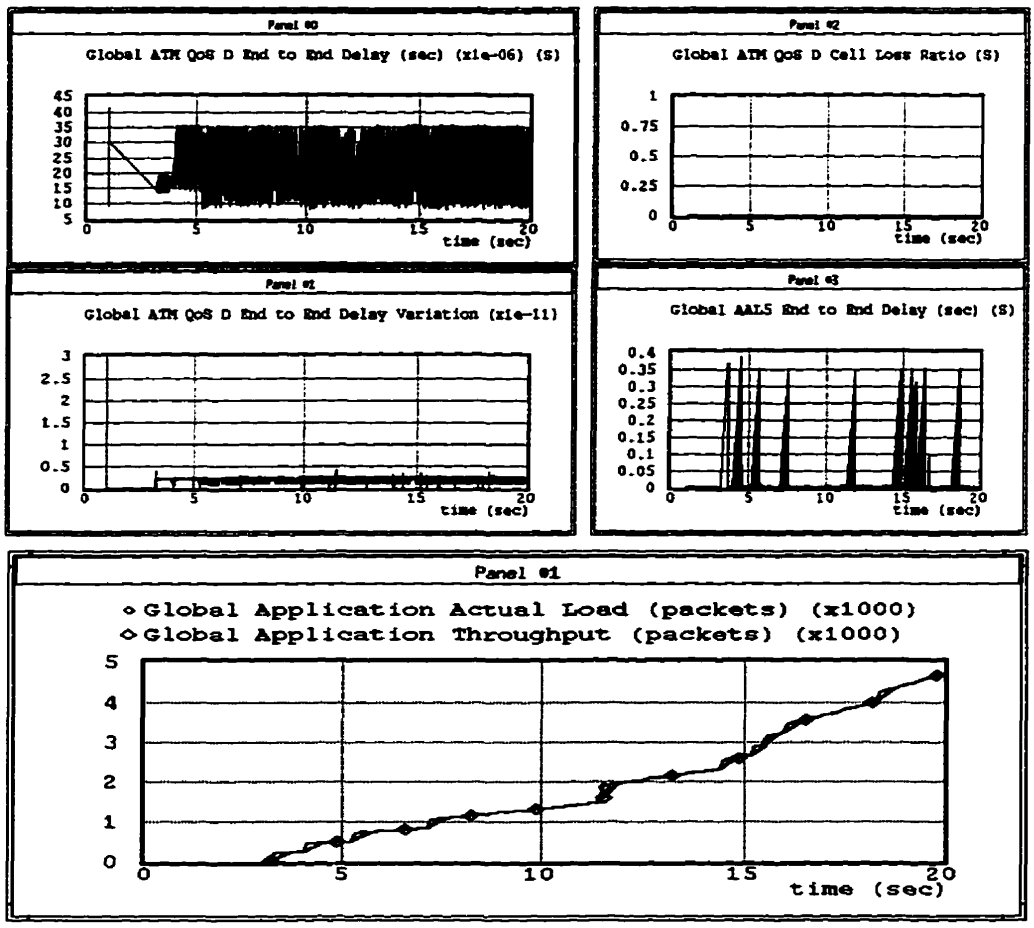

Figure 52. Statistics obtained when decreasing buffer size.

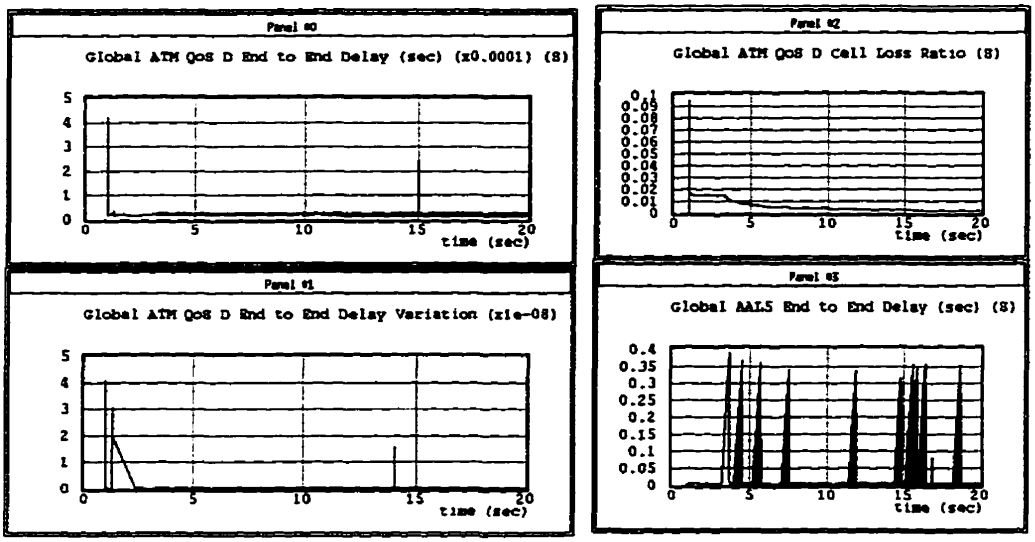

Figure 53. Statistics obtained from reduced buffer size plus the imbalance case.

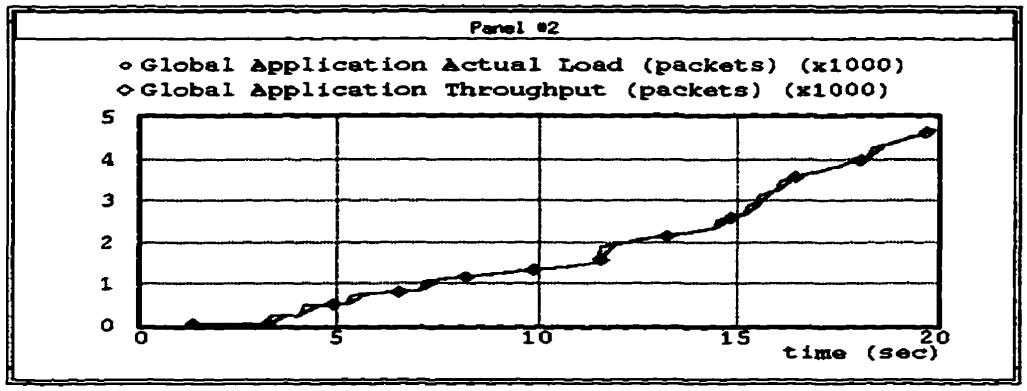

Figure 54. Throughput for less buffer size plus imbalance case.

Specifying traffic more bursty: According to Table 10, Figure 39 and Figure 40, traffic arrivals of X Windows and Remote Login include the burst periods. This experiment increases the application traffic loads of X Windows and Remote Login two times. Similarly, two cases are considered.

Case 1: Only double the application traffic loads of X Windows and Remote Login. Figure 55 and Figure 56 represent the simulation results of this case. Results show that the AAL and ATM delay increase, CLR and ATM CDV remain on the same level.

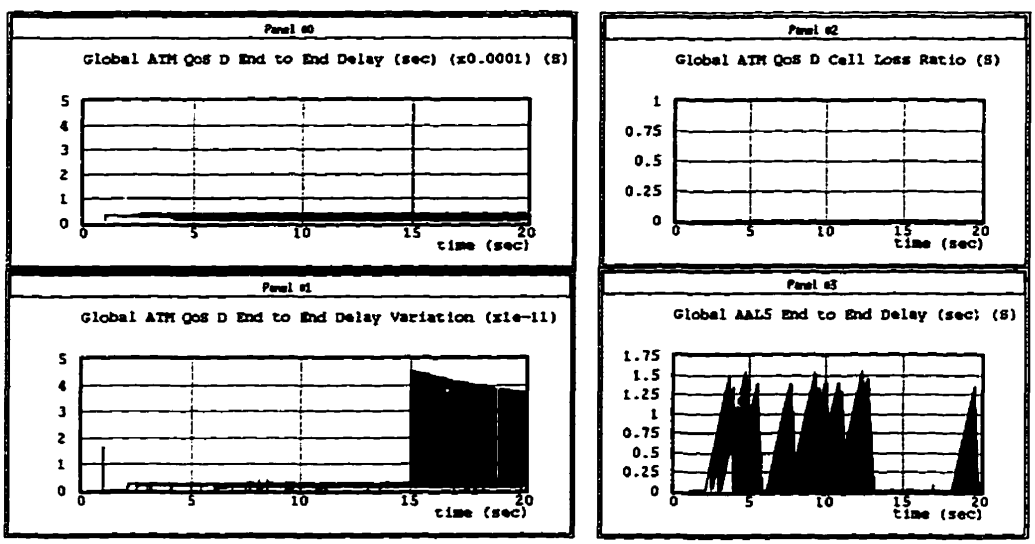

Figure 55. Statistics regarding to doubling bursty traffic.

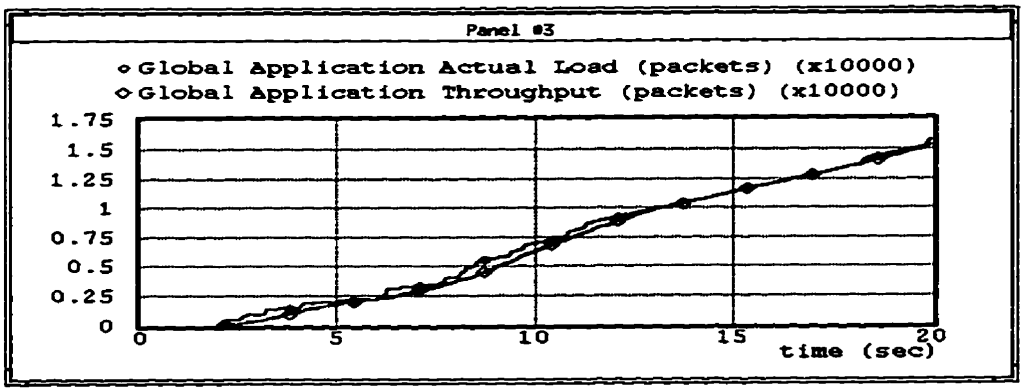

Figure 56. Throughput when doubling bursty traffic.

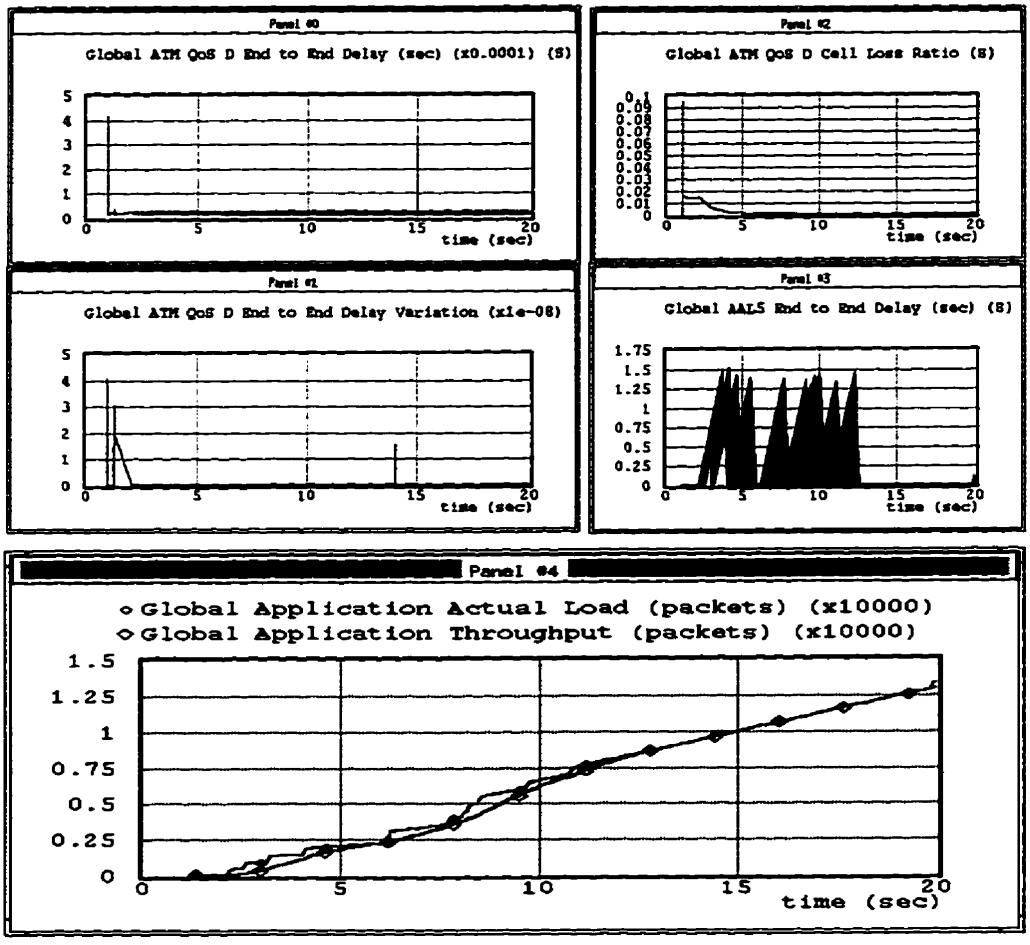

Figure 57. Simulation results of the worst case: 3 problems happen together.

Case 2: In addition to doubling the application traffic loads of X Windows and Remote Login, the buffer size is also decreased and an imbalance situation is also created.

Figure 57 represents the simulation results of this case.

Results: Simulation results show that congestion takes place. Firstly, CLR is no longer zero (Figure 53 and Figure 57); secondly, the throughput is smaller than the traffic actual load (Figure 56); and thirdly, the delay on both AAL and ATM layers increases dramatically. Also, Figure 57 represents the results of the worst case.

5.3 Conclusions

As mentioned earlier in the chapter, the congestion problem is a global issue and a dynamic problem which is difficult to solve. The purpose of this chapter is to use OPNET to design and simulate ATM network models to investigate the congestion problem since OPNET has an ability to simulate communication protocols and networks. According to a series of simulations done in this chapter, some results illustrating congestion are presented.

1. Standard client-server models are suitable for this kind of modeling purpose, because applications provided by these models can generate CBR, VBR and bursty traffic.
2. Theoretically, OPNET is unlimited. However, its ability is limited by the computer used. Comparing to the simulation duration, the model size is more crucial, because increasing the model size causes both disk space and RAM to increase, while increasing the simulation time only requires more disk space.
3. Generally speaking, OPNET is a suitable tool for modeling and simulation of ATM networks under both normal and abnormal conditions since problems, such as congestion, can be created by specifying model attributes.

CHAPTER 6 CONCLUSIONS

6.1 Summary

In this thesis, a number of ATM network models for traffic management have been developed. Several groups of simulations were designed to investigate network performance and OPNET limitations. Because the focus of the thesis is developing models for traffic management and congestion which is always a potential problem in high-speed networks, a simulation set exclusively for the congestion problem was designed and simulated.

Results from a series of simulations provided useful information for investigating what causes congestion, which parameters influence it most significantly, as well as if OPNET is suitable for this kind of purpose modeling. Generally speaking, OPNET is a suitable tool for modeling and simulation of ATM networks in both normal and abnormal conditions since the problem conditions, such as congestion, can be created by specifying model attributes.

OPNET was found to be a useful tool in the modeling, simulation and analysis of ATM networks because OPNET provides a wide variety of standard models, a well developed Graphic User Interface (GUI) and convenient mechanisms for data collecting and performance analysis. However, its ability is limited by the computer used. Results show that the model size (the number of nodes) is the key parameter in the simulation with respect

to the limitations of OPNET.

Also, an ATM network model connecting with Ethernet LANs has been designed and simulated. This model represents a general idea of using OPNET to develop protocols, interfaces and communication networks, as well as basic concerns of using OPNET to evaluate model performance.

In addition, a hardware CA implementation for generating random distributions for ATM traffic sources has been designed, simulated and compared with three other schemes. The simulation results show that the hardware implementation designed in this thesis is efficient and fast.

6.2 Contributions

These are the contributions of this thesis:

1. A model of an ATM network with Ethernet has been developed and simulated. Since this model is built based on a real network, it provides an evidence that OPNET is suitable tool for a simulation study of ATM networks.
2. Models of client-server networks have been developed and simulated. These models are used to examine OPNET limitations and investigate congestion in ATM networks. Simulation results prove that:
   - OPNET is a suitable tool for modeling and simulation of ATM networks under both normal traffic and abnormal traffic (e.g., congestion) conditions, since problems can be created by specifying model attributes and modifying parameters.
   - Three factors can cause congestion: imbalance traffic load, insufficient buffer size and bursty traffic.

3. A hardware implementation with 1-D LHCA to generate random distributions for ATM traffic has been designed and simulated. This novel CA architecture is used in ATM traffic modeling research, and the design proposed here is efficient according to the simulation results.

6.3 Future work

The following issues can be further studied in the future:

- Modifying associated process models to implement congestion control functions other than those implemented in standard models, examining the model performance.
- Modifying associated process models to generate the user designed traffic sources, or making the data of traffic arrivals obtained from real networks available in the OPNET models.
- Constructing a true Internet, evaluating its performance.
- Changing structures or parameters of some components of the network, investigating the network dependability.

REFERENCES

[1] W. Stallings, Data and Computer Communications, Second Edition, 1988 by Macmillan Publishing Company.

[2] Andrew S. Tanenbaum, Computer Networks, Third Edition, 1996 by Prentice Hall PTR.

[3] Uyless Black, ATM foundation for broadband networks, by Prentice Hall PTR, 1995.

[4] C. Partridge, Gigabit Networking, 1994 by Addison-Wesley Publishing Company.

[5] L. Staalhagen, "A Comparison Between the OSI Reference Model and the B-ISDN Protocol Reference Model", IEEE Network, Vol. 10, No. 1, January/February 1996, pp.24-33.

[6] C. Bisdikian, etc., "Approaching B-ISDN: An Overview of ATM and DQDB", Asynchronous Transfer Mode Networks, Edited by Y. Viniotis and R.O. Onvural, Plenum Press, New York, 1993, pp.55-73.

[7] Charles Spurgeon, Quick reference guide to 10Mbps Ethernet, Web site: http://www.ots.utexas.edu/ethernet/ethernet-home.html.

[8] A. M. Law and M. G. McComas, "Simulation Software for Communications Networks: The State of the Art", IEEE Communications Magazine, March 1994, pp.44-50.

[9] L. Gun and G.A. Marin, "An overview of the ATM Forum and the traffic management activities", Asynchronous Transfer Mode Networks, Edited by Y. Viniotis and R.O. Onvural, Plenum Press, New York, 1993, pp.21-29.

[10] R. Jain, "Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey", Department of Computer and Information Science, The Ohio State University, October, 1995.

[11] J. J. Bae and T. Suda, "Survey of Traffic Control Schemes and Protocols in ATM Networks", Proceedings of The IEEE. Vol. 79, No. 2, February 1991, pp.170-189.

[12] C. Lefelhocz, etc., "Congestion Control for Best-Effort Service: Why We Need a New Paradigm", IEEE Network, Vol. 10, No. 1, January/February 1996, pp.10-19.

[13] T. M. Chen, etc., "The Available Bit Rate Service for Data in ATM Networks",

IEEE Communication Magazine, Vol. 34, No. 5, May 1996, pp.56-71.

[14] H. T. Kung and R. Morris, "Credit-Based Flow Control for ATM Networks", IEEE Network, March/April 1995, pp.40-48.

[15] F. Bonome and K. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", IEEE Network, March/April 1995, pp.25-39.

[16] K. K. Ramakrishnan and P. Newman, "Integration of Rate and Credit Schemes for ATM Flow Control", IEEE Network, March/April 1995, pp.49-56.

[17] T. M. Chen and S. S. Liu, "Management and Control functions in ATM Switching Systems", IEEE Network, July/August 1994, pp.27-40.

[18] C. Baransel, etc., "Routing in Multihop Packet Switching Networks: Gb/s Challenge", IEEE Network, May/June 1995, pp38-61.

[19] ATM Model Description, OPNET Example Models Manual, MIL 3, Inc. Release 2.5.B and 3.0.A.

[20] Ethernet Model Description, OPNET Example Models Manual, MIL 3, Inc. Release 2.5.B.

[21] Process Domain Definitions, OPNET Modeling Manual, Volume 1, MIL 3, Inc. OPNET Doc Viewer, Release 2.5.B and 3.0.A.

[22] Modeling Framework, OPNET Modeling Manual, Volume 1, MIL 3, Inc. OPNET Doc Viewer, Release 2.5.B and 3.0.A.

[23] Modeling Overview, OPNET Modeling Manual, Volume 1, MIL 3, Inc. OPNET Doc Viewer, Release 2.5.B and 3.0.A.

[24] Distribution Package, OPNET Simulation Kernel Manual, MIL 3, Inc. OPNET Doc Viewer, Release 2.5.B and 3.0.A.

[25] Z. Yao and D.C. Blight, "Generating Traffic Distributions For ATM Networks: A Cellular Automaton Implementation", will appear in WESCANEX'97 Proceedings, May 1997.

[26] P. P. Chu, "ATM Burst Traffic Generator," Proceedings of Fifth Great Lakes Symposium on VLSI, March 1995, pp.262-265.

[27] W.E. Leland, etc., "On the Self-Similar Nature of Ethenet Traffic (Extended Version)", IEEE/ACM Trans. on Networking, Vol. 2, No. 1, February 1994, pp.1-15.

[28] P. Pruthi and A. Erramilli, "Heavy-Tailed ON/OFF Source Behavior and Self-Sim-

ilar Traffic", Proceedings of ICC'95, June 18-22, 1995.

[29] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", Proceeding of the ACM Sigcomm'94, London, UK, pp.257-268, 1994.

[30] S. Wolfram, "Random Sequence Generation by Cellular Automata", Advances in applied mathematics 7, 123-169 (1986).

[31] W. Pries, A. Thanailakis and H. C. Card, "Group Properties of Cellular Automata and VLSI Applications", IEEE Trans. on Computers. Vol. C-35, No. 12, Dec. 1986.

[32] P. D. Hortensius, R. D. McLeod and H. C. Card, "Parallel Random Number Generations for VLSI Systems Using Cellular Automata", IEEE Trans. on Computers. Vol.38, No. 10, Oct. 1989.

[33] P. H. Bardell, "Analysis of Cellular Automata Used as Pseudorandom Pattern Generators", Proceedings of International Test Conference, 1990, pp.762-768.

[34] K. Cattell and S. Zhang, "Minimal Cost One-Dimensional Linear Hybrid Cellular Automata of Degree through 500", Journal of Electronic Testing: Theory and Applications, 6, 1995.

[35] S. Nandi and P. P. Chaudhuri, "Analysis of Periodic and Intermediate Boundary 90/150 Cellular Automata", Submitted IEEE Trans. on Computers, 1994.

[36] D. R. Chowdhury, I. Sengupta, and P. P. Chaudhuri, "A Class of Two-dimensional Cellular Automata and Their Applications in Random Pattern Testing", Journal of Electronic Testing: Theory and Applications, 5, 67-82 1994.

[37] B. Maglaris, etc., "Performance Models of Statistical Multiplexing in Packet Video Communications", IEEE Trans. on Communications, Vol. 36, No. 7, July 1988, pp.834-843.

[38] Sheldon. M. Ross, "Introduction to Probability Models", Fourth Edition, 1989 by Academic Press, Inc.

[39] Z. Yao and D.C. Blight, "Modeling And Simulation Of ATM Networks", will appear in WESCANEX'97 Proceedings, May 1997.

APPENDIX I: EXPANSION OF DETERMINANTS OF THE CERTAIN FORM

The determinant which has the form: $M_0 = \begin{bmatrix} a_{11} & 1 & 0 & \cdots & \cdots \\ 1 & a_{22} & 1 & \cdots & \cdots \\ 0 & 1 & a_{33} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & a_{nn} \end{bmatrix}$

can be found recursively as follows:

$$M_0 = \left(a_{11} \begin{bmatrix} a_{22} & 1 & 0 & \cdots \\ 1 & a_{33} & 1 & \cdots \\ 0 & 1 & a_{44} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \right) + \begin{bmatrix} 1 & 1 & 0 & \cdots \\ 0 & a_{33} & 1 & \cdots \\ 0 & 1 & a_{44} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

$$= \left(a_{11} \begin{bmatrix} a_{22} & 1 & 0 & \cdots \\ 1 & a_{33} & 1 & \cdots \\ 0 & 1 & a_{44} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \right) + \begin{bmatrix} a_{33} & 1 & 0 & \cdots \\ 0 & a_{44} & 1 & \cdots \\ 0 & 1 & a_{55} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

$$M_1 = \begin{bmatrix} a_{22} & 1 & 0 & \cdots \\ 1 & a_{33} & 1 & \cdots \\ 0 & 1 & a_{44} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \quad M_2 = \begin{bmatrix} a_{33} & 1 & 0 & \cdots \\ 1 & a_{44} & 1 & \cdots \\ 0 & 1 & a_{55} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Thus, we have:

$M_0 = a_{11}M_1 + M_2$ *and*

$M_i = a_{i+1,i+1}M_{i+1} + M_{i+2}$.

Since, $M_{n-1} = a_{nn}M_n + M_{n+1} = a_{nn}$,

$M_n = 1$, $M_{n+1} = 0$.

Example (Figure 11):

$$det(A-\lambda I) = 0, \quad |A-\lambda I| = \begin{vmatrix} 1-\lambda & 1 & 0 & 0 & 0 & 0 \\ 1 & (-\lambda) & 1 & 0 & 0 & 0 \\ 0 & 1 & (-\lambda) & 1 & 0 & 0 \\ 0 & 0 & 1 & (-\lambda) & 1 & 0 \\ 0 & 0 & 0 & 1 & (-\lambda) & 1 \\ 0 & 0 & 0 & 0 & 1 & (-\lambda) \end{vmatrix}$$

$M_6 = 1$, $M_7 = 0$

$M_5 = a_{66}M_6 + M_7 = \lambda$

$M_4 = a_{55}M_5 + M_6 = \lambda\lambda + 1 = \lambda^2 + 1$

$M_3 = a_{44}M_4 + M_5 = \lambda(\lambda^2 + 1) + \lambda = \lambda^3 + \lambda + \lambda = \lambda^3$

$M_2 = a_{33}M_3 + M_4 = \lambda\lambda^3 + \lambda^2 + 1 = \lambda^4 + \lambda^2 + 1$

$M_1 = a_{22}M_2 + M_3 = \lambda(\lambda^4 + \lambda^2 + 1) + \lambda^3 = \lambda^5 + \lambda$

$M_0 = a_{11}M_1 + M_2 = (\lambda + 1)(\lambda^5 + \lambda) + \lambda^4 + \lambda^2 + 1$

$= \lambda^6 + \lambda^5 + \lambda^4 + \lambda + 1$

Because the addition above is the modulo-2 sum, $\lambda + \lambda = 0$, $\lambda^2 + \lambda^2 = 0$, etc.

APPENDIX II: GENERAL DATA FILES

OPNET uses general data files to represent routing tables, address mapping tables, etc., and uses environment files to define parameters for model simulation and performance analysis which have to be done ahead.

VP configuration table: A VP configuration table contains static VP link definitions. These definitions set the characteristics of each VPC on each of the VP links for the duration of the simulation. These characteristics include the data rate and supported QoS class of VPCs. Table 14 is a VP configuration table designed for the model in Figure 27.

User ID	Data rate (Mb/s)	QoS class (A-D)
0	45.0	A, B, C, D
1	155.52	A, B, C, D

Table 14. VP configuration table (my_vp_config.gdf).

Environment files: Most parameters are attributes which can either be specified at the modeling stage or specified as *promoted* at the modeling stage and then defined later in the environment files before simulation. There are four environment files defined for simulating the model in Figure 27. These environment files defined parameters referring to the packet size and application start/end time. Table 15 lists the defined parameters briefly.

Environment files	Packet size Args	Start time				End time			
		N50,...,N59	N60	N100,...,N106	N110	N50,...,N59	N60	N100,...,N106	N110
att1_pkt.ef	1	-	-	-	-	-	-	-	-
att2_pkt.ef	1000	-	-	-	-	-	-	-	-
att1_time.ef	-	0,...,9	>100	11,...,17	>160	1,...,10	>100	12,...,18	>160
att2_time.ef	-	0,...,9	0	11,...,17	11	1,...,10	1	12,...,18	12

Table 15. Environment files for defining parameters.