ATM Traffic Generation and Call Admission Control

By

YuFei Shi

A Thesis
Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Mechanical and Industrial Engineering
University of Manitoba
Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

ATM TRAFFIC GENERATION AND CALL ADMISSION CONTROL

BY

YUFEI SHI

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

MASTER OF SCIENCE

To my parents and my wife, for their encouragement and support.

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.

ABSTRACT

Call Admission Control (CAC) for Asynchronous Transfer Mode (ATM) networks under various traffic types is studied in this thesis. In order to evaluate the performance of different CAC schemes, a generalized traditional traffic generator and a self-similar traffic generator are implemented as process models in OPtimized Network Engineering Tool (OPNET). The generalized traditional traffic generator includes the deterministic ON/OFF, the two-state Markov Modulated Fluid Process (MMFP), the Interrupted Poisson Process (IPP), and the two-state Markov Modulated Poisson Process (MMPP) traffic models as special cases. The self-similar traffic generator is based on multiplexing a number of ON/OFF sources with heavy-tailed ON/OFF periods. Both traffic generators are capable of producing realistic ATM traffic at a very high speed of 25 millions cells per hour. Calls can be setup and released dynamically by those traffic generators.

A comprehensive literature review on existing CAC schemes has been carried out. The advantages and disadvantages of each of these schemes are discussed. A new CAC scheme based on estimating the available bandwidth of an output link is developed. It has no restrictions on the type of traffic sources and is able to operate in real-time. The effect of statistical multiplexing is fully exploited by the new CAC scheme. A mandatory traffic parameter, Peak Cell Rate (PCR) and an optional traffic parameter, Sustained Cell Rate (SCR), are required for each call. The new CAC scheme and two other CAC schemes, peak bandwidth reservation and effective bandwidth reservation, are also implemented as process models in OPNET.

Simulations are performed in OPNET to evaluate the performance of the new CAC scheme. Comparisons are also conducted on the above three CAC schemes, under various traffic arrival processes. Simulation results indicate that the new CAC scheme outperforms the other two CAC schemes under all studied scenarios, with no cell loss and significant improvements of the output link utilization by 100-200 percent in most cases.

ACKNOWLEDGMENTS

During the author's two-year study at the Department of Mechanical and Industrial Engineering, he has enjoyed the pleasure to work with his advisor, Dr. A. S. Alfa. This thesis would not have been possible without the support and guidance from him.

The author would like to thank Dr. R. D. Mcleod from the Department of Electrical and Computer Engineering and Dr. C. M. Laucht from the Department of Computer Science, for taking the time to be on the thesis examining committee and providing helpful comments and suggestions on this thesis.

The author would also like to thank the following friends, graduate students and staff at Telecommunications Research Laboratories (TR*Labs*): Jose Rueda, Jeff Diamond, Dan Erickson, Jeff Giesbrecht, ZhiJian Sun, ZhongHui Yao, Peter Czezowski, George F. Ortega, and Len Dacombe, for their help, kindness, and friendship.

Finally, the author would like to acknowledge TR*Labs* for providing financial support and research facilities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABR	Available Bit Rate
AAL	ATM Adaptation Layer
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband-Integrated Services Digital Network
CAC	Call Admission Control
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CLR	Cell Loss Ratio
CTD	Cell Transfer Delay
IPP	Interrupted Poisson Process
KP	Kernel Procedure
MCR	Minimum Cell Rate
MMFP	Markov Modulated Fluid Process
MMPP	Markov Modulated Poisson Process
NRT-VBR	Non-Real-Time Variable Bit Rate
OAM	Operation, Administration and Maintenance
OPNET	OPtimized Network Engineering Tool
PCR	Peak Cell Rate
pdf	probability density function
QoS	Quality of Service

RT-VBR	Real-Time Variable Bit Rate
SCR	Sustained Cell Rate
SPP	Switched Poisson Process
STD	State Transition Diagram
UBR	Unspecified Bit Rate
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

In the information age, telecommunications are playing a more important role than ever before. The demand for new telecommunication services is growing and becoming increasingly diverse. The traditional response to such a demand has been to offer a new dedicated network to each new service. This, however, requires a large initial investment, even when the long-term demand for each new service cannot be predicted accurately. In addition, the increasing number of specialized networks results in increased maintenance costs along with inefficient use of resources. The need for deployment of a broadband integrated services digital network (B-ISDN) is therefore widely recognized [MILL96]. The underlying technology that makes B-ISDN possible is called Asynchronous Transfer Mode (ATM). ATM is a high-speed, cell switching, and connection-oriented technology

designed to support a wide variety of telecommunication services such as voice, data and video, each with different traffic characteristics and Quality of Service (QoS) requirements. Information is sent in short fixed-length blocks called cells. The flexibility needed to support variable transmission rates is provided by transmitting the necessary number of cells per unit of time [SAIT94].

ATM utilizes the bursty nature of the offered traffic to effectively allocate bandwidths to all accepted calls via statistical multiplexing. The number of cells sent by all users to an output link may exceed the link's capacity temporarily. An output buffer is used to store cells which cannot be transmitted immediately in such a situation. However, if this excess continues, the cell waiting times (cell transfer delay) at the output buffer will increase, and some cells may even be discarded (cell loss) when the buffer is full. Such cell transfer delay and cell loss may seriously interfere with users' communication services [HIRA95]. ATM traffic control is therefore needed to achieve the required QoS of each accepted call in terms of cell loss ratio (CLR), maximum cell transfer delay (maximum CTD), and peak-to-peak cell delay variation (peak-to-peak CDV).

Traffic control methods are often divided into two categories: preventive control and reactive control. In low-speed networks, it is usually adequate to wait for congestion to occur and then react to it by telling the traffic source to slow down. In high-speed networks, this approach often works poorly, because during the interval between sending a congestion notification and the notification's arriving at the source, thousands of additional packets may arrive. Furthermore, ATM networks have real-time traffic sources that produce traffic at an intrinsic rate. Telling such a source to slow down may not work.

Consequently, ATM networks emphasize preventing congestion from occurring in the first place [TANE96].

Call Admission Control (CAC) is a very important component of the preventive traffic control method. CAC represents the set of actions taken by the network at the call setup stage in order to accept or reject a new call based on the availability of network resources, the new call's anticipated traffic characteristics, and the QoS requirements of both already accepted calls and the new call. A new call is accepted only when sufficient resources are available to carry it through the whole network at its required QoS while maintaining the agreed QoS of already accepted calls in the network. Otherwise, the new call is rejected [PRYC95].

Realistic ATM traffic traces from various types of services and scenarios is needed in order to evaluate the performance of different traffic control schemes. Traffic traces can be collected using traffic capture cards or network analyzers from existing ATM networks. However, large scale ATM networks are still being deployed and new services are being introduced. It can be anticipated that traffic traces collected from today's ATM networks will look different from that collected from future ATM networks. Furthermore, collecting traffic traces from existing ATM networks requires a large amount of time, effort, and expensive equipment. Therefore, ATM traffic generators based on advanced traffic models are needed to produce synthetic traffic traces which can resemble the traffic from existing and foreseeable ATM services.

OPtimized Network Engineering Tool (OPNET) provides a convenient environment for modeling and simulating ATM networks. Both the behavior and the performance of modeled ATM networks can be analyzed by performing discrete event simulations.

OPNET consists of a number of tools, each one focusing on a particular stage of a modeling task: specification, simulation, data collection and analysis. OPNET supports model specification with four tools, called Network Editor, Node Editor, Process Editor, Parameter Editor. OPNET models developed with those editors are structured hierarchically, in a manner that parallels real networks [OPNET1]. ATM traffic generators with arbitrary traffic characteristics and call setup/release capabilities can be created by user-defined OPNET process models. CAC schemes for ATM networks can also be implemented as user-defined OPNET process models.

1.2 OBJECTIVES

This thesis focuses on implementing ATM traffic generators in OPNET and developing a new CAC scheme. Specifically, the objectives of this thesis are:

1. Implement ATM traffic generators in OPNET that can produce realistic traffic of various types of services and that have call setup and release capabilities.
2. Develop a new, real-time CAC scheme which can provide the required QoS and improve the output link utilization significantly over existing CAC schemes.
3. Conduct simulations in OPNET to evaluate the performance of the new CAC scheme.
4. Compare the new CAC scheme to other CAC schemes under various traffic arrival processes to show its improvement on output link utilization.

1.3 OUTLINE OF THE REST OF THE THESIS

This rest of the thesis is structured as follows.

Chapter 2 gives an introduction to ATM traffic control, OPNET process models, and the Pareto distribution.

Chapter 3 addresses the issue of generating ATM traffic. Traffic models from the literature are reviewed. The implementations of a generalized traditional traffic generator and a self-similar traffic generator as process models in OPNET are described.

Chapter 4 presents a comprehensive literature review of existing CAC schemes. The advantages and disadvantages of each of them are discussed.

Chapter 5 describes a new CAC scheme based on estimating the available bandwidth of an output link. The new CAC scheme and two other CAC schemes are also implemented as process models in OPNET.

Chapter 6 provides the simulation results of the performance evaluations of the new CAC scheme. Simulation results of the comparisons between the new CAC schemes and two other CAC schemes under various types of traffic arrival processes are also presented.

Chapter 7 concludes this thesis and provides recommendations for future work.

CHAPTER TWO

ATM TRAFFIC CONTROL AND OPNET PROCESS MODELS

This chapter provides the necessary background for the rest of the thesis. Section 2.1 reviews several basic ATM concepts such as the ATM cell and the ATM protocol reference model. Section 2.2 describes the phenomenon of statistical multiplexing. Traffic and QoS parameters, service categories, and traffic control functions in an ATM network are covered in section 2.3. Section 2.4 serves as an introduction to OPNET process models. Section 2.5 gives the definition and properties of the Pareto distribution. A definition of heavy-tailed distributions is also provided in section 2.5.

2.1 ASYNCHRONOUS TRANSFER MODE

The basic idea behind ATM is to transmit all information in short, fix-length blocks called cells [TANE96]. Each cell consists of a 5-byte header and a 48-byte payload. The fixed size of cells allows for fast switching. Transport functions and capabilities in an ATM network are structured hierarchically [SAIT94]. The basic transport element between two end points of an ATM network is the virtual channel. Virtual channels with same end points and same route are often grouped together to form a virtual path. Virtual paths are multiplexed along a transmission path from a given source to a given destination. This makes it easier for network management. There are two fields in the header of a cell, called Virtual Channel Identifier (VCI) and Virtual Path Identifier (VPI). They are used to route cells from one switch to another. ATM networks are connection-oriented [TANE96]. For switched virtual channels, making a call requires first sending a message to set up the connection. After that, subsequent cells of the connection all follow the same path to the destination. Cells sent along a virtual channel never arrive out of order. This allows high-speed communication after a call setup since no additional routing decision need to be made for each cell of the same connection. ATM is mainly designed for use on highly reliable fiber optic networks. Flow control and error recovery are performed on an end-to-end basis. An ATM protocol reference model consists of a user plane, a control plane, and a management plane [SAIT94]. The user plane provides data transport, flow control, error correction, and other user functions. The control plane performs signaling functions such as call setup and release. The management plane's activities include resource management, interlayer coordination, and Operation, Administration, and Maintenance (OAM) functions. There are three layers in the user and

control planes. The lowest layer, the physical layer defines the electrical characteristics and network interfaces. The ATM layer deals with cell header generation and extraction. It also deals with establishment and release of virtual channels. Congestion control is also located here. The highest layer, the ATM adaptation layer (AAL) provides service-independent functions (such as cell segmentation and re-assembly) as well as service-dependent functions (such as message framing and error-detection) to the layers above it.

2.2 STATISTICAL MULTIPLEXING

The cell generation rate of an ATM traffic source usually changes from time to time according to the variation of the amount of information it is asked to transfer per unit of time. As a result, the bandwidth needed to support such a traffic source also varies from time to time. When a number of traffic sources are multiplexed, they do not generate cells at their peak cell rates simultaneously most of the time. Therefore, the total amount of bandwidth needed to support those traffic sources is typically much less than the sum of the peak cell rate of each individual traffic source. This phenomenon, called statistical multiplexing, is one of the most important features of ATM networks.

2.3 TRAFFIC CONTROL IN ATM NETWORKS

The primary role of traffic control in ATM networks is to protect the network from congestion in order to achieve predefined QoS objectives. Congestion can be caused by the cell arrival rate of the aggregate traffic exceeding the network capacity temporarily or

by fault conditions within the ATM network. An additional role of traffic control is to use network resources such as output links efficiently.

2.3.1 Traffic Parameters and QoS Parameters

Before a call is established, both the user and the ATM carrier must agree on a contract which defines the service [TANE96]. The contract between the user and the carrier contains three parts:

1. A traffic descriptor specifying the traffic characteristics of the call.
2. The Quality of Service of the call which both the user and carrier agree upon.
3. A definition of how strictly the first two parts of the traffic contract will be enforced.

Traffic Parameters: A traffic descriptor is the set of traffic parameters specifying the intrinsic traffic characteristics of a call at the time a call is set up. Peak Cell Rate (PCR) is one mandatory traffic parameter in a traffic descriptor. It is the maximum number of cells a traffic source can send per second. Sustained Cell Rate (SCR) is another important traffic parameter which is the expected number of cells a traffic source would send per second averaged over a long time interval. Minimum Cell Rate (MCR) is the minimum number of cells a traffic source would send per second. ATM layer functions (e.g., cell multiplexing) may alter the traffic characteristics of connections by introducing Cell Delay Variation. When cells from two or more connections are multiplexed, cells of a given connection may be delayed while cells of another connection are being inserted at the output of the multiplexer. Consequently with reference to the peak emission interval T (i.e., the inverse of the contracted PCR), some randomness may affect the inter-arrival

time between consecutive cells of a connection (i.e., the inverse of the contracted PCR). The upper bound on the "clumping" measure with reference to the peak emission interval T is the Cell Delay Variation Tolerance (CDVT) associated with the PCR [ATMForumTM]. Similarly, with reference to the sustained emission interval T_S (i.e., the inverse of the contracted SCR), some randomness may affect the inter-arrival time between consecutive cells of a connection (i.e., the inverse of the contracted SCR). The upper bound on the "clumping" measure with reference to the sustained emission interval T_S is the CDVT associated with the SCR [ATMForumTM].

QoS Parameters: Cell Loss Ratio (CLR) measures the fraction of the transmitted cells that are not delivered at all or are delivered too late to be useful (e.g. for real-time traffic). Cell Transfer Delay (CTD) is defined as the elapsed time between a cell exit event at the source and the corresponding cell entry event at the destination for a particular connection. The CTD between a source and a destination for a particular connection includes transmission delay, queuing delay, propagation delay, packetization delay, reassembly delay, switching delay, coding delay, and other delays. Maximum Cell Transfer Delay (maximum CTD) is defined as the $(1-\alpha)$ quantile of the CTD for a particular connection. The requested CLR is used to place an upper bound on α. Peak-to-peak cell delay variation (Peak-to-peak CDV) is defined as the maximum CTD minus the fixed CTD that could be experienced by any delivered cell on a connection during the entire connection holding time. Figure 2-1 below illustrates the probability density function of the CTD in CBR and real-time VBR services, and relates it to the maximum CTD and peak-to-peak CDV parameters [ATMForumTM].

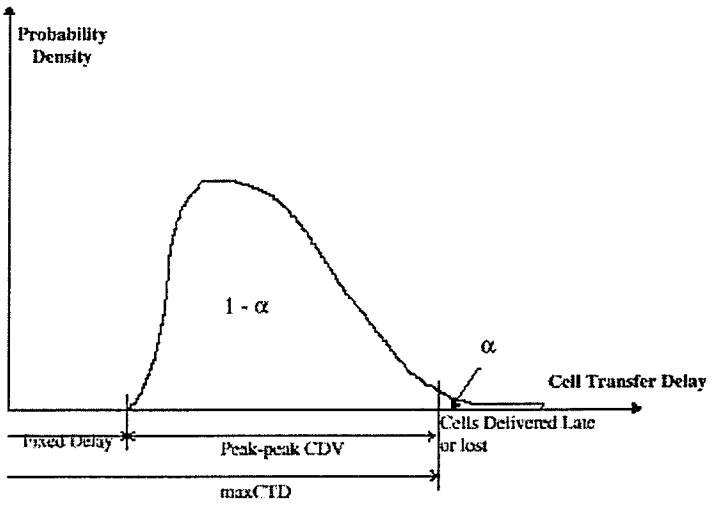

Figure 2-1. CTD probability density function, maximum CTD, and peak-to-peak CDV

2.3.2 Service Categories

The ATM service architecture specifies five different service categories: Constant Bit Rate (CBR), Real-Time Variable Bit Rate (RT-VBR), Non-Real-Time Variable Bit Rate (NRT-VBR), Available Bit Rate (ABR), Unspecified Bit Rate (UBR) [TANE96][MILL96]. Each of these five categories has its own traffic parameters and QoS parameters as shown in Table 2-1[MILL96].

	CBR	RT-VBR	NRT-VBR	UBR	ABR
Traffic Parameters:					
PCR and CDVT	Specified				
SCR and CDVT	N/A	Specified		N/A	
MCR	N/A				Specified
QoS Parameters:					
Peak to peak CDV	Specified		Unspecified		
Maximum CTD	Specified		Unspecified		
CLR	Specified			Unspecified	Low
Other Attribute:					
Rate Feedback	Unspecified				Specified

Table 2-1. ATM service categories

CBR: Connections of this service category are provided a fixed bandwidth for the duration of a connection. In addition, a timing relationship is maintained between the source and destination. This service category is intended to be used by real-time applications that require tight constraints on the maximum CTD and peak-to-peak CDV such as voice, television, or circuit emulation service.

VBR: The VBR service category is divided into two subclasses, real time and non-real time, respectively. RT-VBR is intended for services that have variable bit rates combined with stringent real-time requirements, such as interactive compressed video (e.g., video-conferencing). Both maximum CTD and peak-to-peak CDV must be tightly controlled. Occasional cell loss is tolerable and is just ignored. NRT-VBR is intended for applications that have bursty traffic, but do not require such stringent controls on cell delay such as transaction processing.

ABR: The ABR service category is designed to support applications that are able to increase or decrease their information throughput if network circumstances dictate. ABR is the only service category in which the network provides rate feedback, also called flow control, to the sender, asking it to slow down when congestion occurs. Cell loss for ABR is expected to be low if the sender complies with such requests. An example of an ABR application is web browsing.

UBR: Connections of the UBR service categories are given no promises of QoS and feedback about congestion. All cells from UBR connections are accepted, and if there is capacity left over, they will also be delivered. If congestion occurs, cells from UBR

connections will be discarded, with no feedback to the sender and no expectation that the sender slows down. As such, it is referred to as a best-effort service. Potential candidates for UBR service are email and USENET news.

2.3.3 Call Admission Control

CAC represents the set of actions taken by the network at call set up or re-negotiation phase in order to accept or reject a new call. A call setup request is accepted only when sufficient resources are available to carry the new call through the whole network at its requested QoS while maintaining the agreed QoS of already established connections in the network. CAC schemes are currently not standardized and are at the discretion of network operators. A comprehensive literature review on CAC schemes is provided in Chapter Four.

2.3.4 Usage Parameter Control

Usage Parameter Control (UPC) represents a set of actions taken by the network to monitor and control the traffic on an ATM connection. The main purpose of UPC is to force every ATM connection to comply with its traffic contract. UPC is performed for each traffic parameter in a source traffic descriptor. If UPC detects a violation, it can either discard non-conforming cells or tag them for discarding when the network is congested. Several UPC mechanisms have been proposed in the literature [PATH91]. They are briefly reviewed in the following sections.

2.3.4.1 The Leaky Bucket Mechanism

The leaky bucket mechanism consists of a pseudo-buffer with a counter. The counter is increased by one when a cell from the source is received by the pseudo-buffer. The counter is decreased by one at a constant service rate *R* as long as the counter value is positive. This constant service rate *R* of the pseudo-buffer is usually set to the PCR of the source and the limit of the counter *K* is usually set to the CDVT of the source to assure that the UPC mechanism tolerates for the traffic fluctuations caused by CDV and burstiness. This mechanism is illustrated in Figure 2-2. If the pseudo-buffer overflows, the UPC assumes a violation has occurred and suitable penalty imposing actions (such as discarding or tagging cells) are taken on all subsequently arriving cells from that source until the counter has fallen below its limit again.

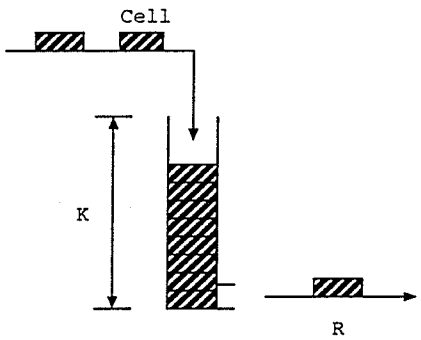

Figure 2-2. The leaky bucket mechanism

2.3.4.2 The Jumping Window Mechanism

The jumping window mechanism defines windows that do not overlap but stick to each other as shown in Figure 2-3. The maximum number of cells which is allowed to be accepted within a window is set a threshold value according to the PCR specified by the

user. A counter counts the number of arriving cells in each window and is reset at the beginning of the following window. A violation occurs when the count exceeds the threshold value.

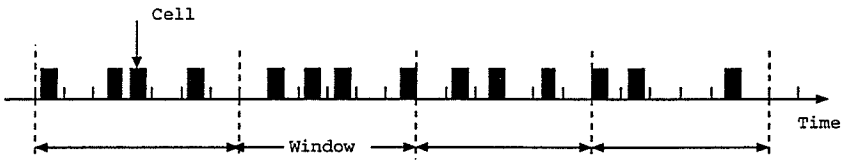

Figure 2-3. The jumping window mechanism

2.3.4.3 The Triggered Jumping Window Mechanism

The windows in the jumping window mechanism are not synchronized with the cell arrivals. The triggered jumping window mechanism has been proposed where windows may not stick to each other but are triggered by the first arriving cell in a batch as depicted in Figure 2-4.

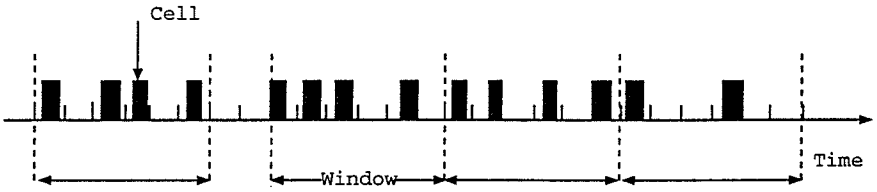

Figure 2-4. The triggered jumping window mechanism

2.3.4.4 The Exponentially Weighted Moving Average Mechanism

The exponentially weighted moving average mechanism defines windows the same way as the jumping window mechanism, but the maximum number of cells allowed to be

accepted within a window is a function of the exponentially weighted sum of the number of cells accepted in the preceding windows.

2.3.4.5 The Moving Window Mechanism

The moving window mechanism defines a window moving along the time axis as illustrated in Figure 2-5. Similar to the jumping windows mechanism, the maximum number of arriving cells in every fixed window is set to a threshold value according to the PCR specified by the user. This mechanism is very stringent and difficult to implement because the arrival time of each cell needs to be remembered for exactly one window width.

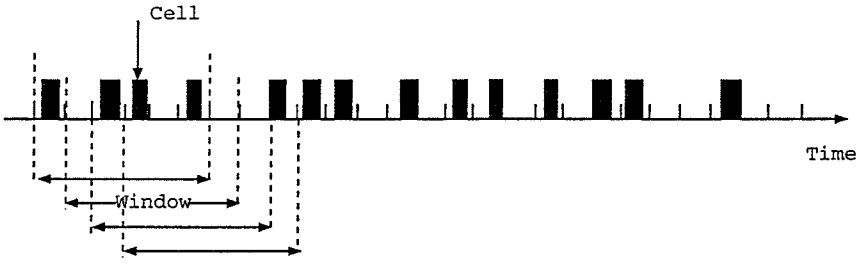

Figure 2-5. The moving window mechanism

2.3.5 Other Traffic Control Functions

In addition to CAC and UPC, several other traffic control functions are performed by the network to monitor and regulate traffic. They are network resource management, traffic shaping, priority control, and feedback control [PRYC95]. Network resource management describes the provisions used to allocate network resources in order to separate traffic flows according to their characteristics. Traffic shaping is defined as a

mechanism to alter the traffic characteristics of a connection in order to reduce the PCR, burst length, and CDV of that connection. Priority control allows users to generate different priority traffic flows. At the time of congestion, low priority cells are discarded to protect cells with higher priority. Feedback control is defined as a set of actions taken by both the network and the users to adjust the traffic according to the congestion status of the network and the availability of network resources.

2.4 OPNET PROCESS MODELS

Process models are the only objects in OPNET which can be programmed by users to implement the desired behaviors and functions of an entity. OPNET expresses process models based on a combination of state transition diagram (STD), a library of high level functions called Kernel Procedures (KP) and the C programming language. A process model's STD defines a set of top-level modes or states that a process can enter as well as a set of conditions which control the process moving from one state to another. States are mutually exclusive and complementary, which means that a process is always in one and only one state. Actions may be associated with each state and they are called executives. The executives of a state are split into two sections, called enter executives and exit executives. A state's enter executives are executed when a process enters the state, and its exit executives are executed when the process leaves the state for another state. There are two types of states, called forced states and unforced states. Unforced states allow a pause between their enter executives and exit executives and thus can be used to model true

states of a process. The symbols used to represent forced and unforced states in an STD are illustrated in Figure 2-6.

Figure 2-6. Symbols used to represent forced and unforced states in an STD

Processes are driven by events. When an event is actually delivered to a process, it is termed as an interrupt and the process is said to be invoked. Interrupts are distinguished by the types and codes associated with them. A process that is in a rest period waiting to be invoked is said to be blocked. A process cycles between invoked and blocked periods. Once a process has completed the enter executives of an unforced state, it blocks and returns control to the simulation kernel or the process that invoked it so that events of other processes can be executed. At this point, the process remains blocked until a new invocation causes it to resume execution of the exit executives of its current state, to move to another state, and to perform actions there. Once these actions are completed, the process must be blocked again. Each process should have at least one unforced state. Processes are not allowed to rest in forced states. Therefore, forced states cannot used to represent modes of a process that persist for any duration. In other words, the exit executives of a forced state are executed by a process immediately upon completion of the enter executives. One special state, called initial state, must be designated in each process model. Graphically, an initial state is identified by a large arrow located at its left side as shown in Figure 2-7. The initial state is the place where the first invocation of the

process begins. This state usually contains executive statements performing initializations that should occur only once.

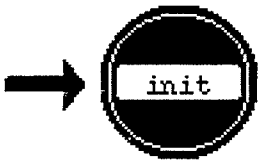

Figure 2-7. Symbol used to represent an initial state in an STD

A transition specification consists of four components: a source state, a destination state, a condition expression and an executive expression. Each state may have any number of incoming and outgoing transitions which are depicted as directed arcs with an arrow pointing toward to the destination state. Transition conditions determine which transition should take place. The condition and executive expressions appear in a combined label next to the arc. A forward slash ('/') is used to separate them. Transitions that have non-empty condition expressions are depicted as dashed arcs and those without conditions are depicted as solid arcs. A special condition called "default" is provided to represent the complement of all other conditions associated with transitions leaving a state. A default condition succeeds if and only if all other transition conditions are false. Figure 2-8 gives an example STD showing transition conditions and transition executives.

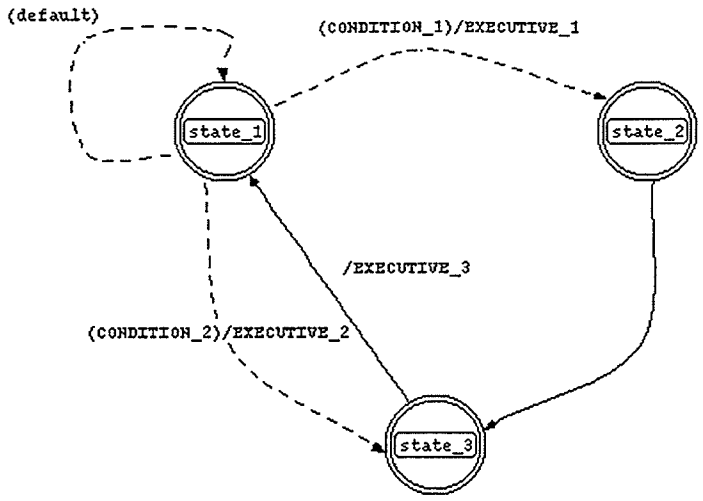

Figure 2-8. An example STD showing transition conditions and transition executives

Transition executives are useful in order to associate an action with a state when it is entered or exited for a particular reason. Incorporating an action into the state's enter or exit executives would cause the action to be executed regardless of the reason for which the state was entered or exited, which may be undesirable in some cases. Transition conditions and executives are expressed as "English-like" text called macros. This technique explains transition conditions and executives in a simple, natural, and user-friendly way and hides the details of their implementations from users.

Process models can be parameterized using model attributes. This mechanism gives users the power to control the behavior, capabilities, and functions of such a model without changing its internals. Model attributes generalize a model, making it more flexible and reusable. An example of process model attributes is given in Figure 2-9. Typically, each process model that uses model attributes loads the values of attributes into a set of state variables in its initial state.

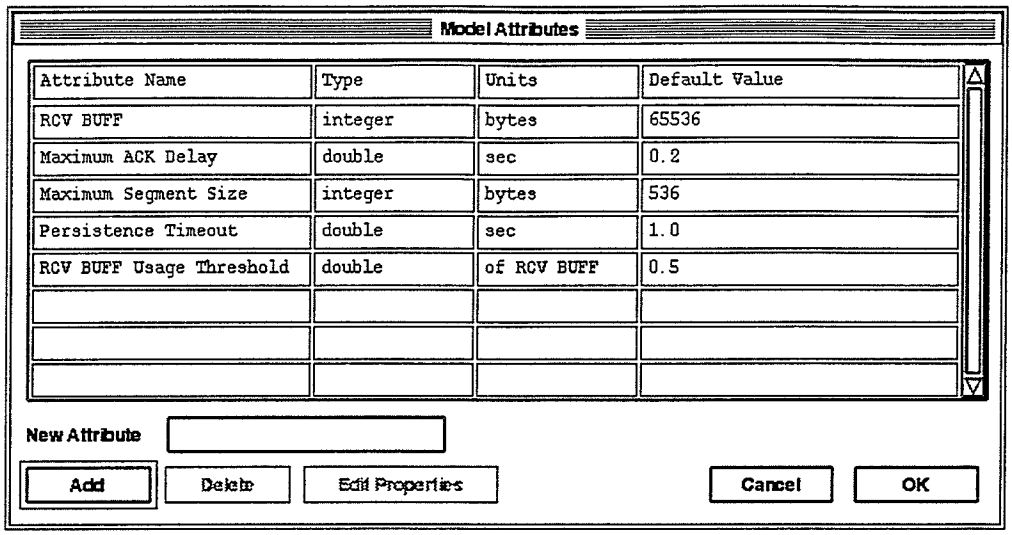

Figure 2-9. An example of process model attributes

In OPNET, a process is capable of creating and destroying other processes. This is useful to manage tasks that are generated on a dynamic basis, i.e., their exact timing or number is not known in advance. This technique is used in this thesis to multiplex the traffic from a number of heavy-tailed ON/OFF traffic sources to produce self-similar traffic. When a self-similar traffic source wants to establish a connection, the process of each heavy-tailed ON/OFF traffic source is created by the process of the self-similar traffic source. When a self-similar traffic source wants to release its connection, the process of each heavy-tailed ON/OFF traffic source is destroyed by the process of the self-similar traffic source. The process of a self-similar traffic source is never destroyed. It is called the parent process of the process of each heavy-tailed ON/OFF traffic source. The process of each heavy-tailed ON/OFF traffic source is called a child process of the process of the self-similar traffic source.

2.5 THE PARETO DISTRIBUTION

Heavy-tailed distributions, especially the Pareto distribution, play a very important role in the construction of self-similar processes. A Pareto distribution with a shape parameter β and a location parameter α has the following cumulative distribution function:

$$F(x) = P[X \le x] = 1 - (\alpha / x)^{\beta}, \quad \alpha, \beta > 0, \quad x \ge \alpha,$$

with the corresponding probability density function:

$$f(x) = \beta \alpha^{\beta} x^{-\beta-1}.$$

Clearly, the closer the shape parameter β of a Pareto distribution gets to 1, the heavier the tail probability $P[X > x]$ of the Pareto distribution, or in other words, the greater the variability of the outcomes of the Pareto distribution. If $\beta \le 2$, the distribution has infinite variance, and if $\beta \le 1$, then it has infinite mean. For $\beta > 1$, the mean of the Pareto distribution is $\beta\alpha / (\beta - 1)$ [PAXS95a].

The Pareto distribution (also referred to as power-law distribution, hyperbolic distribution) has been used to model distributions of random variables exceeding a minimum value α. The Pareto distribution is scale-invariant, which means that the probability that the outcome is at least $2x$ is a fixed fraction of the probability that the outcome is at least x, for any $x \ge \alpha$. The Pareto distribution is also the only distribution that is "invariant under truncation from below". That is, for $y \ge x_0$,

$$P[X > y | X > x_0] = P[(\frac{x_0}{\alpha}) X > y].$$

Hence, the conditional distribution is also a Pareto distribution, with the same shape parameter β and a new location parameter $\alpha' = x_0$.

A distribution is defined as heavy-tailed if its cumulative distribution function is of the form:

$$P[X \geq x] \sim L(x)x^{-\beta}, \quad \text{as } x \to \infty,$$

where $0 < \beta < 2$ and $L(x)$ is a slowly varying function, i.e., $\lim_{x\to\infty} L(tx)/L(x) = 1$, for all $t > 0$ [LELA94]. (Asymptotically constant and logarithm are two examples of slowly varying functions.) This definition includes the Pareto distribution when the shape parameter $0 < \beta < 2$.

A more general definition of heavy-tailed is given in [PAXS95a] where the distribution of a random variable X is said to be heavy-tailed if its conditional mean excess CME_X is an increasing function of x, where

$$CME_x = E[X - x | X \geq x].$$

The conditional mean excess of a medium-tailed distribution is a constant. The conditional mean excess of a light-tailed distribution is a decreasing function of x.

For waiting times with a light-tailed distribution such as the uniform distribution, the longer you have waited, the sooner you are likely to be done. For waiting times with a medium-tailed distribution such as the (memoryless) exponential distribution, the expected future waiting time is independent of the waiting time so far. In contrast, for waiting time with a heavy-tailed distribution, the longer you have waited, the longer is your expected future waiting time.

CHAPTER THREE

GENERATING ATM TRAFFIC

This chapter addresses the issue of generating ATM traffic. Realistic ATM traffic is in great need and of extremely importance for both engineering designs and performance evaluations of ATM traffic control schemes. A generalized traditional traffic generator and a self-similar traffic generator are implemented as process models in OPNET to meet this need. This chapter is structured as follows. In section 3.1, the motivation for implementing ATM traffic generators in OPNET is presented. In section 3.2, several traditional ATM traffic models, including the ON/OFF process, the Markov Modulated Fluid Process, the Interrupted Poisson Process, and the Markov Modulated Poisson Process, are reviewed and their implementation as one generalized traditional traffic generator is described. In section 3.3, the concept, definition, and measurement of self-similarity are surveyed and the implementation of a high-speed self-similar traffic

generator in OPNET is presented. The self-similar traffic generator is based on multiplexing a number of ON/OFF sources with heavy-tailed ON/OFF periods. The relations between the model attributes of the self-similar traffic generator and the Hurst parameter of the synthetic self-similar traffic traces are also studied in section 3.3. Section 3.4 summarizes this chapter.

3.1 MOTIVATION

Traffic control schemes are vital to the successful operation of an ATM network. Realistic ATM traffic is needed in order to evaluate the performance of various traffic control schemes. Over the last decade, researchers around the world have gone through several paradigm shifts regarding the modeling of an ATM traffic source [PERR96]. Early traffic models based on the Poisson process cannot capture the burstiness present in ATM traffic. Therefore, there was a major shift towards using ON/OFF type traffic models, such as the ON/OFF process and the Interrupted Poisson Process (IPP). Later, more complex traffic models such as the Markov Modulated Fluid Process (MMFP) and the Markov Modulated Poisson Process (MMPP) were introduced to capture the short-term correlation present in ATM traffic. More recently, self-similar traffic models based have been proposed to capture the long-term correlation present in ATM traffic.

OPNET is becoming one of the leading software packages for simulating ATM networks. Both the behavior and the performance of modeled ATM networks can be analyzed by performing discrete event simulations. However, OPNET's built-in traffic generators are based on using basic random distributions such as exponential, normal, and

uniform to model cell interarrival times and therefore can not capture the burstiness, short term, and long term correlation found in ATM traffic. Furthermore, OPNET's built-in traffic generators do not have the ability to set up and release calls dynamically, to signal the traffic control managers, and to specify the traffic parameters of each new call. Therefore, OPNET's built-in traffic generators are not suitable to be used directly in simulations for the performance evaluations of ATM traffic control schemes. More sophisticated ATM traffic generators are needed to be created by user-defined OPNET process models. In this chapter, we present the implementations of a generalized traditional traffic generator and a self-similar traffic generator with the following features:

1. Ability to generate realistic traffic of various types of ATM services.
2. Ability to set up and release calls dynamically.
3. Ability to signal the traffic control managers when setting up or releasing a call.
4. Ability to specify the traffic parameters of each new call.

Both traffic generators are mainly implemented for generating ATM traffic. However, they can be configured to produce packets of various formats as well. Since a cell is just a packet with a fixed size and a specific format, the term packet is used to stand for both cell and packet in the rest of this chapter except for those concepts exclusively used for ATM cells such as peak cell rate and average cell rate.

3.2 TRADITIONAL TRAFFIC MODELS

3.2.1 ON/OFF Process

An ON/OFF process switches between two states, ON and OFF alternately as shown in Figure 3-1. At the ON state, packets are generated at a constant rate, r packets/second, and at the OFF state, no packets are generated. Both ON and OFF periods are exponentially distributed with mean $1/\lambda$ and $1/\mu$ second(s), respectively.

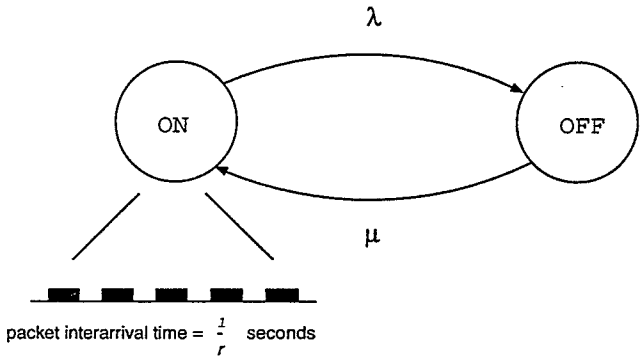

Figure 3-1. An ON/OFF process

3.2.2 Markov Modulated Fluid Process

An m-state Markov Modulated Fluid Process (MMFP) can find itself in m different states [ELWA93]. At state i, $i=1,\ldots,m$, packets are generated at a state dependent constant rate, λ_i packets/second. The transitions among those m states are controlled by a continuous-time Markov chain with an $m\times m$ infinitesimal generator matrix Q, where $Q_{ii}<0$ and $Q_{ij}\geq 0$ for $i\neq j$. The sojourn times at state i, $i=1,\ldots,m$, are exponentially distributed with mean $-1/Q_{ii}$ second(s). An MMFP can be fully characterized by two $m\times m$ matrices: the infinitesimal generator matrix Q and the packet generation rate

matrix Λ, where $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$. A two-state MMFP is illustrated in Figure 3-2. The ON/OFF process is a special case of the two-state MMFP where one of the packet arrival rates λ_1 or λ_2 is 0.

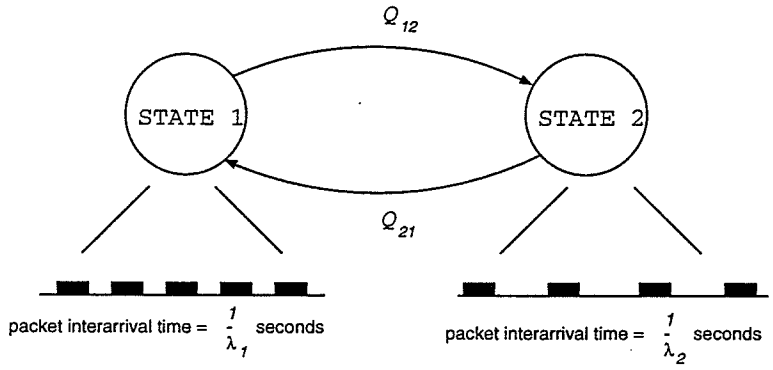

Figure 3-2. A two-state Markov modulated fluid process

3.2.3 Interrupted Poisson Process

An Interrupted Poisson Process (IPP) switches between two states, active and idle alternately as delineated in Figure 3-3 [KUCZ73]. During an active period, packets are generated in a Poisson fashion, i.e., packet interarrival times are of exponential distribution with mean $1/r$ second(s). During an idle period, no packets are generated. Both active and idle periods are exponentially distributed with mean $1/\lambda$ and $1/\mu$ second(s), respectively. The only difference between the IPP and the ON/OFF process is that during an active (or ON) period, packet interarrival times are of exponential distribution in the IPP and are constant in the ON/OFF process.

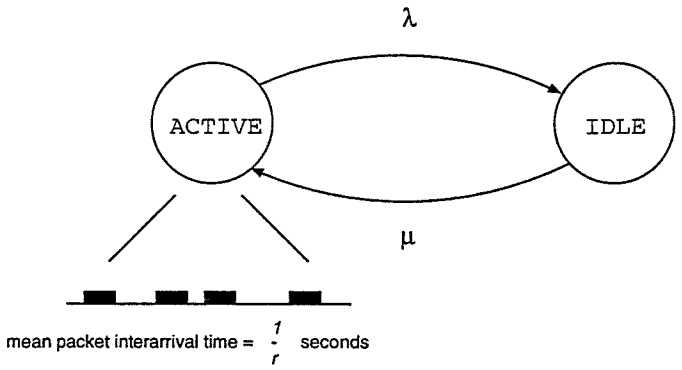

Figure 3-3. An interrupted Poisson process

3.2.4 Markov Modulated Poisson Process

An m-state Markov Modulated Poisson Process (MMPP) can find itself in m different states [HEFF86]. At state i, $i = 1,\ldots,m$, packets are generated in a Poisson fashion at a state dependent rate, λ_i packets/second. The transitions among those m states are controlled by a continuous-time Markov chain with an $m \times m$ infinitesimal generator matrix Q, where $Q_{ii} < 0$ and $Q_{ij} \geq 0$ for $i \neq j$. The sojourn times at state i, $i = 1,\ldots,m$, are exponentially distributed with mean $-1/Q_{ii}$ second(s). An MMPP can be fully characterized by two $m \times m$ matrices: the infinitesimal generator matrix Q and the packet generation intensity matrix Λ, where $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$. A two-state MMPP is sometimes called a Switched Poisson Process (SPP). The IPP is a special case of the SPP where one of the packet arrival intensities λ_1 or λ_2 is 0. A two-state MMPP is depicted in Figure 3-4. The only difference between an m-state MMPP and an m-state MMFP is that at each state, packet interarrival times are of exponential distribution in the MMPP and are constant in the MMFP.

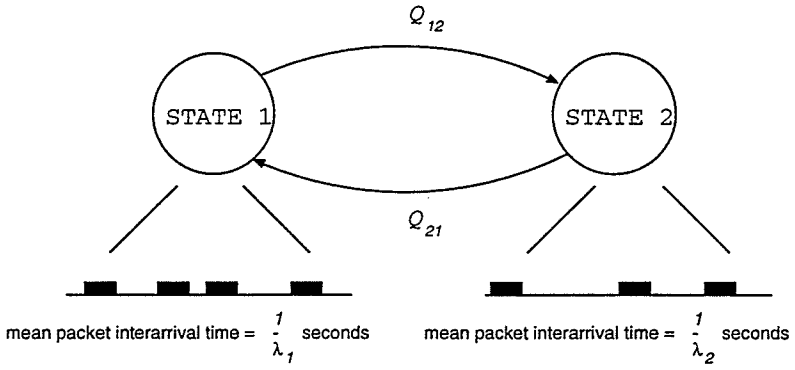

Figure 3-4. A two-state Markov modulated Poisson process

3.2.5 Implementation of a Generalized Traditional Traffic Generator

A generalized traditional traffic generator was implemented as a process model in OPNET. It includes the ON/OFF process, the two-state MMFP, the IPP, and the two-state MMPP as special cases. MMFP and MMPP with more than two states are not considered in this implementation. However, they can be supported by simply adding more states and model attributes. Call setup/release capabilities are also incorporated into the generalized traditional traffic generator. The states of the generalized traditional traffic generator can be decomposed into two levels: a call-level and a burst-level. At the call level, the traffic generator switches between two states, connected and released alternately. At the burst-level, the connected state is further divided into two states, state 1 and state 2. The traffic generator switches between state 1 and state 2 alternately when a call is connected. State 1 can be thought of as the ON state in the ON/OFF process, or as the active state in the IPP, or as the state 1 in either the two-state MMFP or the two-state MMPP. State 2 can be thought of as the OFF state in the ON/OFF process, or as the idle state in the IPP, or as the state 2 in either the two-state MMFP or the two-state MMPP. The packet interarrival

time distributions at both state 1 and state 2, the call holding time and the call interarrival time distributions, and the state 1 and state 2 resident time distributions, can be chosen from: constant, exponential, Pareto and other OPNET built-in distributions, such as uniform, normal, and Erlang. A special "NONE" distribution is also provided for the packet interarrival time distribution at state 2. If this distribution is chosen, no packets are generated at state 2.

The state transition diagram of the process model of the generalized traditional traffic generator is shown in Figure 3-5. The process model has 26 model attributes which are listed in Figure 3-6 (a)-(d). There are seven states in the process model: one forced (init), and six unforced (warm_up, call_setup, state1, state2, released and disabled). They are described in Table 3-1. The process model has 15 transition conditions which are delineated in Table 3-2. Eleven procedures are used in the enter and transition executives of the process model. Their descriptions are given in Table 3-3.

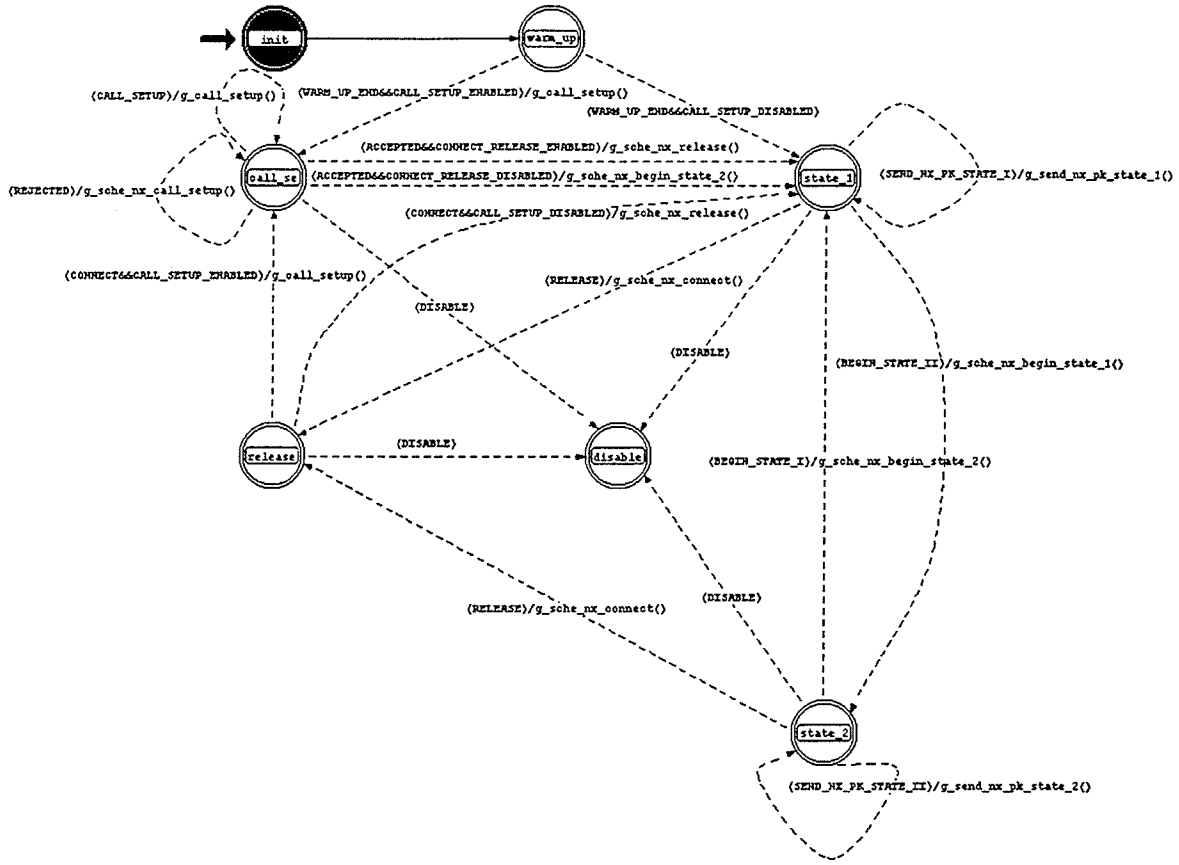

Figure 3-5. State transition diagram of the generalized traditional traffic generator

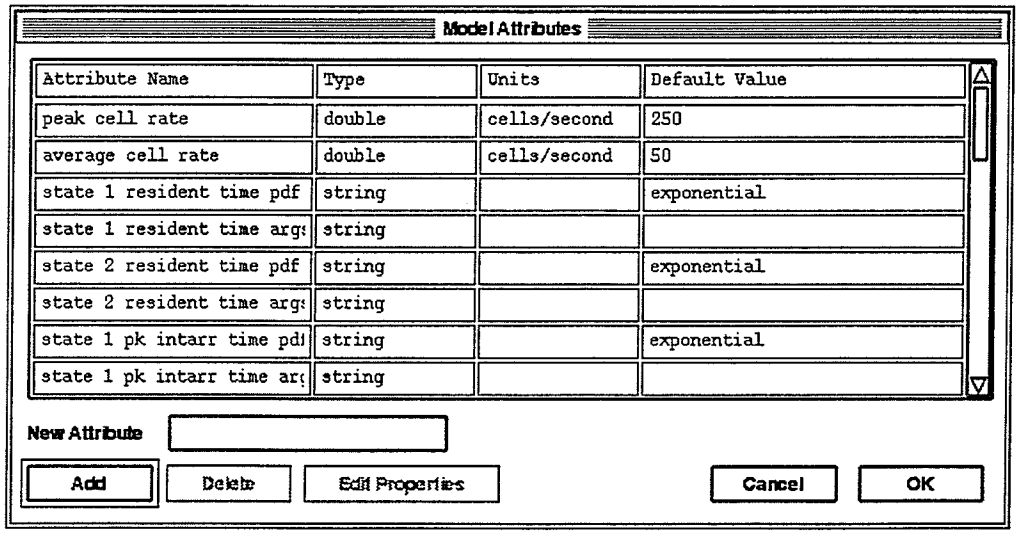

Figure 3-6 (a). Model attributes 1-8 of the generalized traditional traffic generator

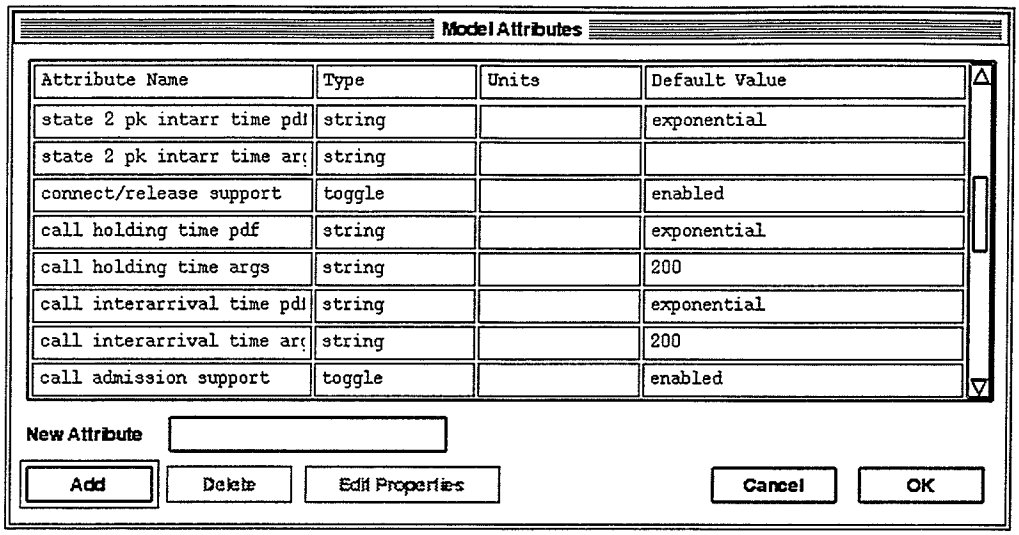

Figure 3-6 (b). Model attributes 9-18 of the generalized traditional traffic generator

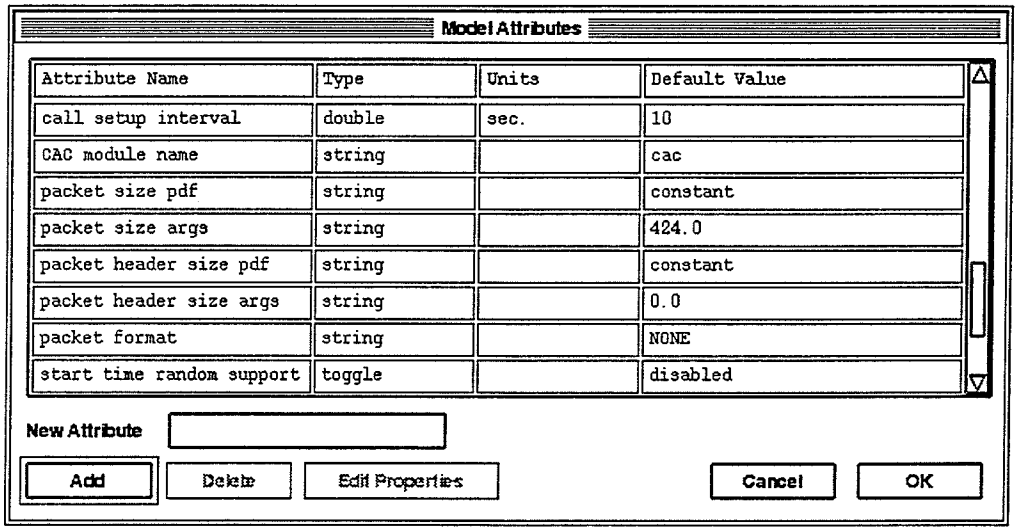

Figure 3-6 (c). Model attributes 19-24 of the generalized traditional traffic generator

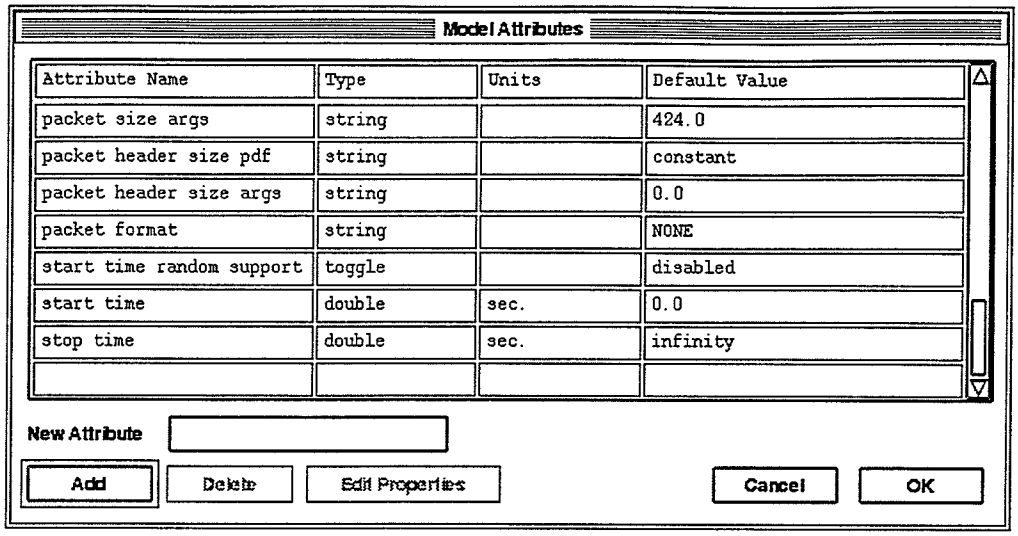

Figure 3-6 (d). Model attributes 20-26 of the generalized traditional traffic generator

State Descriptions

State	Description
init	The process begins at this state where user-supplied model attribute values are read and saved in corresponding state variables. If the "call admission support" model attribute is disabled, a self interrupt is scheduled in order to let the process switch to the state2 state for the first time after it stays at the state1 state for a random period of time. The length of this random period of time is an outcome of the state 1 resident time distribution which can be specified by the "state 1 resident time pdf" and the "state 1 resident time args" model attributes. If the "connect/release support" model attribute is enabled and the "call admission support" model attribute is disabled, another self interrupt is scheduled in order to let the process switch to the released state for the first time after a call has been connected for a random period of time. The length of this random period of time is an outcome of the call holding time distribution which can be specified by the "call holding time pdf" and the "call holding time args" model attributes. A "warm_up_end" self interrupt is also scheduled in a period of time in order to let the process leave the warm_up state and start setting up a call or transmitting packets for the first time. If the "stop time" model attribute is set to be different from the default value "infinity", another self interrupt is scheduled in order to disable the traffic generator permanently at the time specified by the "stop time" model attribute.
warm_up	This state is entered after the initialization is completed at the init state. If the "random start time support" model attribute is enabled, the process stays at this state for a random period of time before setting up a call or

	transmitting any packets. If the "connect/release support" model attribute is enabled, the length of this random period of time is an outcome of the call interarrival time distribution, which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model attributes. If the "connect/release support" model attribute is disabled, the length of this random period of time is an outcome of the state 2 resident time distribution, which can be specified by the "state 2 resident time pdf" and the "state 2 resident time args" model attributes. This feature enables a number of traffic sources to be independent of each other when traffic from them is multiplexed. If the "start time random support" model attribute is disabled, the process stays at this state for a fixed period of time before setting up a call or transmitting any packets. The length of this fixed period of time can be specified by the "start time" model attribute. This attribute can be set to zero, which allows the traffic generator to start setting up a call or transmitting packets right after the simulation starts.
call_setup	This state is entered every time the traffic generator wants to make a call if the "call admission support" model attribute is enabled. Before any packet transmission takes place, the process sends a call setup request with the call's traffic parameters to a neighboring CAC module and waits for the CAC module to send back an accept/reject notification. The PCR of the call can be specified by the "peak cell rate" model attribute. The SCR of the call can be specified by the "average cell rate" model attribute. The CAC module is identified by the "CAC module name" model attribute. If the call setup request is accepted, the process moves to the state1 state and starts transmitting packets at rate 1. If the call setup request is rejected, the process remains at this state and waits for a fixed period of time before resending the last call setup request to the neighboring CAC module. The length of this fixed period of time can be specified by the "call setup interval" model attribute.
state1	This state is entered when a state 2 period is over, or when a call setup request is accepted if the "call admission support" model attribute is enabled, or when a released period is over if the "connect/release support" model attribute is enabled and the "call admission support" model attribute is disabled. At this state, the procedure "g_send_nx_pk_state_1()"is repeatedly called and packets are generated at rate 1. The traffic generator stays at this state for a random period of time before a transition to the state2 state takes place, unless a "release the ongoing call" or "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the state 1 resident time distribution, which can be specified by the "state 1 resident time pdf" and the "state 1 resident time args" model attributes. At this state, packet interarrival times are outcomes of a distribution which can be specified by the "state 1 pk intarr pdf" and the "state 1 pk intarr args" model attributes.

state2	This state is entered when a state1 period is over if the call is still connected. At this state, the procedure "g_send_nx_pk_state_2()"is repeatedly called and packets are generated at rate 2. The process stays at this state for a random period of time before it switches back to the state1 state unless a "release the ongoing call" or "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the state 2 resident time distribution, which can be specified by the "state 2 resident time pdf" and the "state 2 resident time args" model attributes. At this state, packet interarrival times are outcomes of a distribution which can be specified by the "state 2 pk intarr pdf" and the "state 2 pk intarr pdf args" model attributes. If the "state 2 pk intarr pdf" is set to "NONE", no packets are generated at the state2 state.
released	This state is entered when a call holding time period is over if the "connect/release support" model attribute is enabled. The call is then released. If the "call admission support" model attribute is enabled, the process stays at this state for a random period of time before a transition to the call_setup state takes place unless a "disable the traffic generator" self interrupt occurs during this random period of time. If the "call admission support" model attribute is disabled, the process also stays at this state for a random period of time before a transition to the connected state takes place unless a "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the call interarrival time distribution, which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model attributes.
disabled	This state is entered when the simulation reaches the time specified by the "stop time" model attribute if the attribute's value is set to be smaller than the default value "infinity". The process then remains at this state for the rest of the simulation. No packets are generated after this state is entered. The traffic generator is practically disabled.

Table 3-1. State descriptions of the generalized traditional traffic generator

Transition Condition Descriptions

Transition Condition	Description
WARM_UP_END	The warm-up period is over. It is time for the process to move to the call_setup state to start setting up a new call if the "call admission support" model attribute is enabled, or to move to the connected state to start transmitting packets if the "call admission support" model attribute is disabled.

CALL_SETUP_ENABLED	The "call admission support" model attribute is enabled.
CALL_SETUP_DISABLED	The "call admission support" model attribute is disabled.
CALL_SETUP	A call setup interval is over. It is time for the process to resend the last call setup request to a neighboring CAC module.
ACCEPTED	A process interrupt is received from a neighboring CAC module. The last call setup request is accepted.
REJECTED	A process interrupt is received from a neighboring CAC module. The last call setup request is rejected.
BEGIN_STATE_I	A state 2 period is over. It is time for the process to switch back to the state1 state from the state2 state.
BEGIN_STATE_II	A state 1 period is over. It is time for the process to switch back to the state2 state from the state1 state.
SEND_NX_PK_STATE_I	A packet transmission at the state1 state is completed. It is time for the process to start another packet transmission at the state1 state.
SEND_NX_PK_STATE_II	A packet transmission at the state2 state is completed. It is time for the process to start another packet transmission at the state2 state.
CONNECT_RELEASE_ENABLED	The "connect/release support" model attribute is enabled.
CONNECT_RELEASE_DISABLED	The "connect/release support" model attribute is disabled.
CONNECT	A call interarrival time period is over. It is time for the traffic generator to make a new call.
RELEASE	A call holding time period is over. It is time for the traffic generator to release the ongoing call.
DISABLE	The stop time of this traffic generator is reached. It is time for the process to disable the traffic generator.

Table 3-2. Transition condition descriptions of the generalized traditional traffic generator

Procedures Descriptions

Procedure	Description
g_string_to_args(src_string, arg0, arg1)	This procedure decomposes a model attribute of type "string" into one or two arguments of type "double".
g_send_nx_pk_state_1()	This procedure creates a packet of the format specified by the "packet format" model attribute, and of the length specified by the "packet size pdf" and the

	"packet size args" model attributes. The length of the header of a packet can be specified by the "packet header size pdf" and the "packet header size args" model attributes. It then sends the packet to an output stream. It also schedules a "send next packet at state 1" self interrupt in a random period of time unless the traffic generator switches back to the state2 state or releases the call before this interrupt takes place. The length of this random period of time is an outcome of the state 1 packet interarrival time distribution which can be specified by the "state 1 pk intarr pdf" and the "state 1 pk intarr args" model attributes.
g_send_nx_pk_state_20	This procedure creates a formatted or unformatted packet and then sends it to an output stream. It also schedules a "send next packet at state 2" self interrupt in a random period of time unless the traffic generator switches back to the state1 state or releases the call before this interrupt takes place. The length of this random period of time is an outcome of the state 2 packet interarrival time distribution which can be specified by the "state 2 pk intarr pdf" and the "state 2 pk intarr args" model attributes.
g_sche_nx_begin_state_10	This procedure schedules a self interrupt in random period of time in order to let the traffic generator switch back to the state1 state from the state2 state unless a "release the ongoing call" or "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the state 2 resident time distribution which can be specified by the "state 2 resident time pdf" and the "state 2 resident time args" model attributes. The event time of this interrupt is saved in a state variable called "scheduled_go_to_state1_time" which is used in the procedure "g_send_nx_pk_state_2()" to prevent scheduling another "send next packet at state 2" self interrupt if the traffic generator is about to switch back to the state1 state.
g_sche_nx_begin_state_20	This procedure schedules a self interrupt in a random period of time in order to let the traffic generator switch back to the state2 state from the state1 state unless a "release the ongoing call" or "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the state 1 resident

	time distribution which can be specified by the "state 1 resident time pdf" and the "state 1 resident time args" model attributes. The event time of this interrupt is saved in a state variable called "scheduled_go_to_state2_time" which is used in the procedure "g_send_nx_pk_state_1()" to prevent scheduling another "send next packet at state 1" self interrupt if the traffic generator is about to switch back to the state2 state.
g_sche_nx_release()	This procedure schedules a self interrupt in a random period of time in order to let the traffic generator release the ongoing call and switch back to the release mode from the connected mode unless a "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the call holding time distribution which can be specified by the "call holding time pdf" and the "call holding time pdf args" model attributes. The event time of this interrupt is saved in a state variable called "scheduled_release_time" which is used in the procedures "g_send_nx_pk_state_1()" and "g_send_nx_pk_state_2()" to prevent scheduling another "send next packet at state 1" or "send next packet at state 2" self interrupt, and in procedures "g_sche_nx_begin_state_1()" and "g_sche_nx_begin_state_2()" to prevent scheduling another "go to state 1" or "go to state 2" self interrupt if the traffic generator is about to release the ongoing call.
g_sche_nx_connect()	This procedure schedules a self interrupt in a random period time in order to let the traffic generator make a new call and switch back to the connected mode from the released mode unless a "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the call interarrival time distribution which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model attributes.
g_call_setup()	This procedure sends a forced remote interrupt to notify the neighboring CAC module that the traffic generator wants to make a call. It also sends the traffic generator's process identification, the call's PCR, SCR, and other traffic parameters to the CAC module.

	Those traffic parameters are used by the CAC module to decide the acceptance of the call. The traffic generator's process identification is used by the CAC module to identify the call if the call is accepted.
g_sche_nx_call_setup()	This procedure schedules a self interrupt in a fix period of time to let the traffic generator to resend the last call setup request to a neighboring CAC module if the request was rejected. The length of this fixed period of time can be specified by the "call setup interval" model attribute.
g_call_release()	This procedure sends a forced remote interrupt to notify the neighboring CAC module that the traffic generator wants to release the ongoing call. It also sends the traffic generator's process identification to the CAC module so that the CAC module can locate and delete the record of this call from a list of connected calls.
g_pareto_dist(alpha, beta)	This procedure produces a random outcome of a Pareto distribution with two parameters: alpha and beta. The first parameter alpha is the mean of the outcome. The second parameter is the shape parameter of the Pareto distribution.

Table 3-3. Procedure descriptions of the generalized traditional traffic generator

3.3 SELF-SIMILAR TRAFFIC MODELS

Recent analysis of Ethernet traffic measurements has shown that Ethernet traffic is self-similar in nature [LELA94]. Other studies have indicated that self-similarity is a feature of VBR video traffic [BERA95]. Studies of wide area traffic have also revealed the existence of self-similarity [PAXS95a]. Contrary to the common beliefs that aggregate traffic becomes smoother (less bursty) as the number of traffic sources increases, aggregating self-similar traffic typically intensifies burstiness rather than diminishes it. Self-similar phenomena will have a significant impact on the operation, design and control of ATM networks.

3.3.1 A Graphic View of Self-Similarity

Graphically, self-similar traffic seems to look like the same across a wide range of time scales [LELA94]. Figure 3-7 (a)-(e) depicts a sequence of cell count plots (i.e. the number of cells generated per time unit) of a synthetic self-similar traffic for five different choices of time unit.

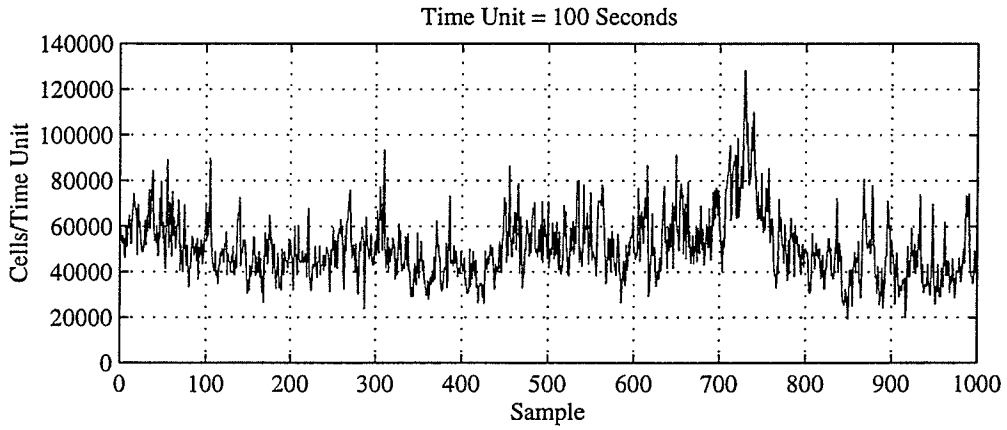

Figure 3-7 (a). Cell count plot of a synthetic self-similar traffic: time unit = 100 seconds

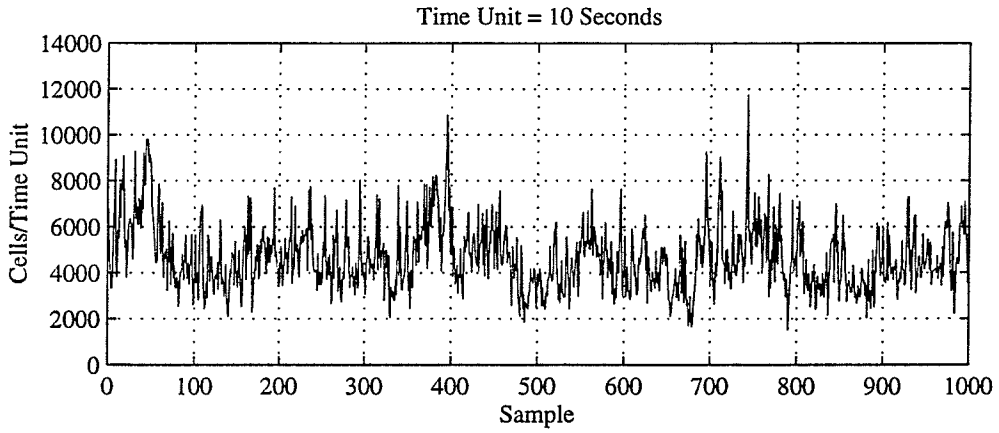

Figure 3-7 (b). Cell count plot of a synthetic self-similar traffic: time unit = 10 seconds

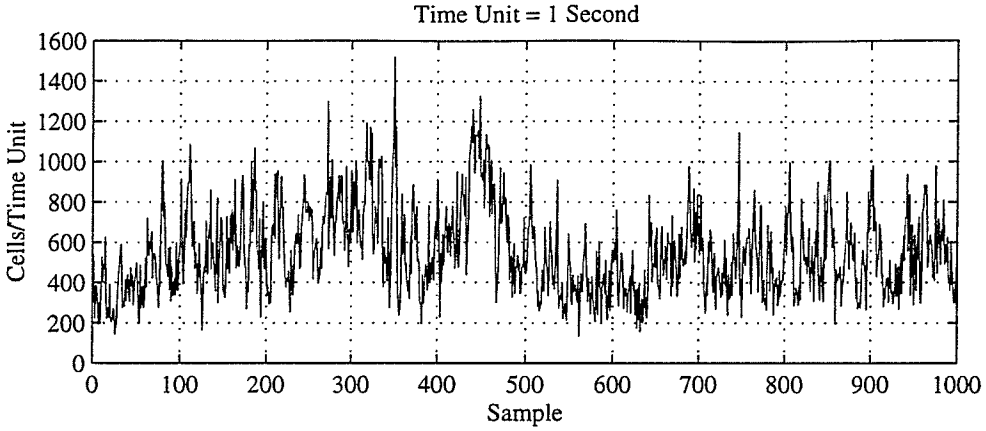

Figure 3-7 (c). Cell count plot of a synthetic self-similar traffic: time unit = 1 second

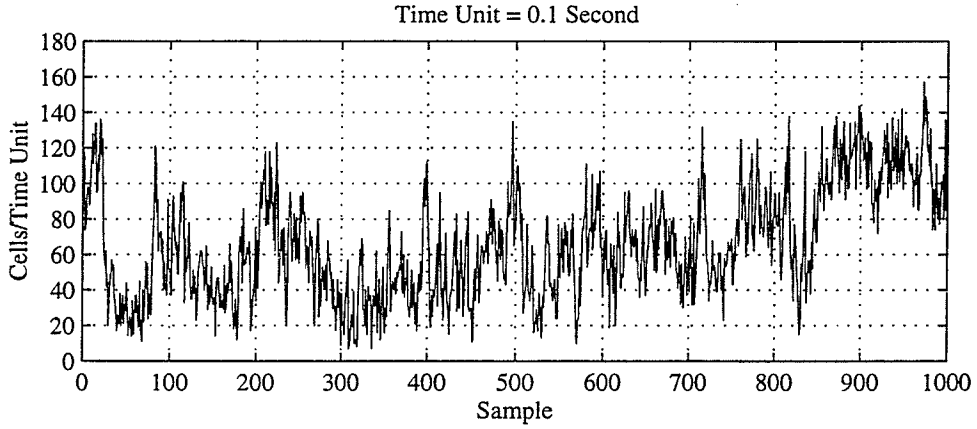

Figure 3-7 (d). Cell count plot of a synthetic self-similar traffic: time unit = 0.1 second

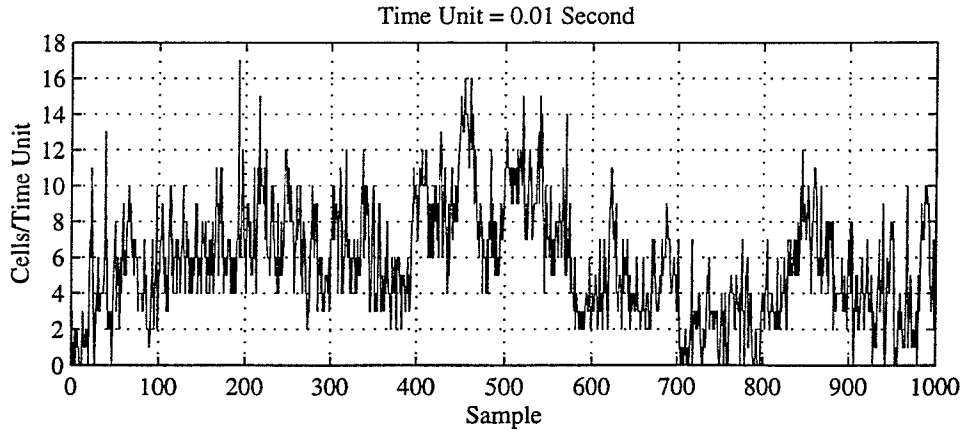

Figure 3-7 (e) Cell count plot of a synthetic self-similar traffic: time unit = 0.01 second

Starting with a time unit of 100 seconds, each subsequent plot is obtained from the previous one by increasing the time resolution by a factor of 10 and by concentrating on a randomly chosen subinterval. Observe that there are bursts ranging from milliseconds to minutes instead of a natural length of burst.

This scale-invariant feature of self-similar traffic is drastically different from traditional traffic models. The latter typically produce cell count plots which are indistinguishable from white noise after aggregating over a few hundred milliseconds. Figure 3-8 (a)-(e) depicts a sequence of cell count plots of a synthetic Poisson process with the same average cell rate for five different choice of time unit.

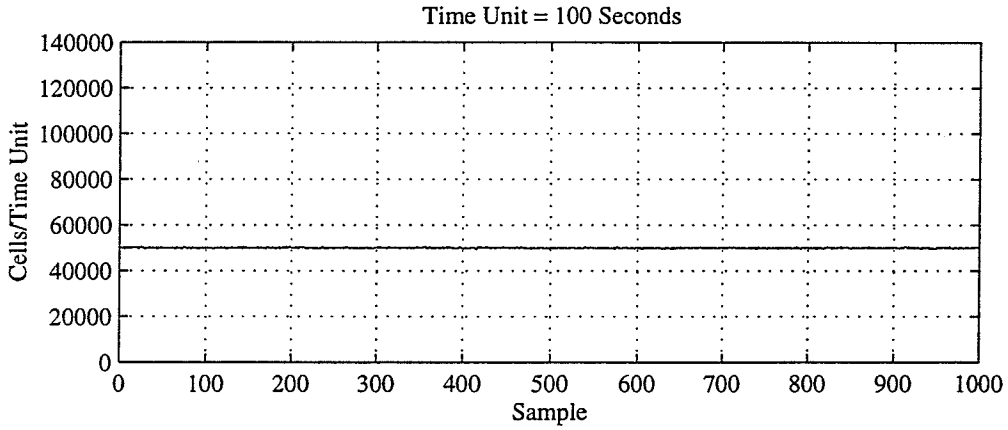

Figure 3-8 (a). Cell count plot of a synthetic Poisson process: time unit = 100 seconds

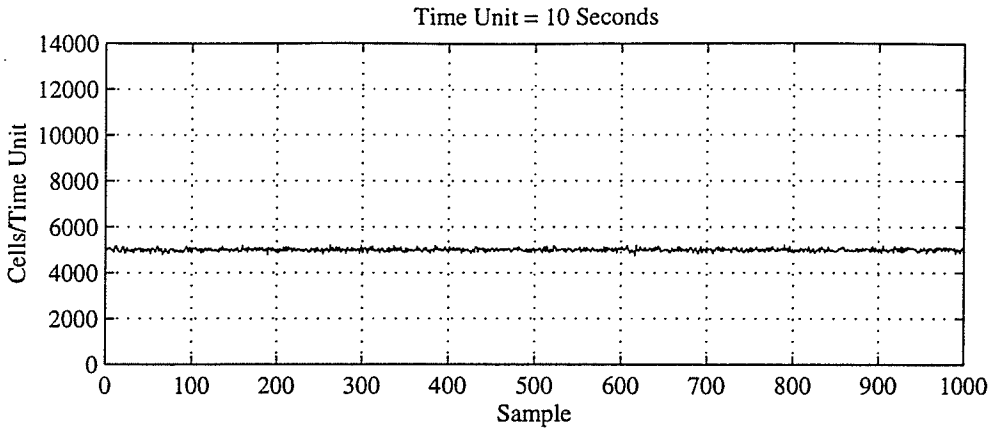

Figure 3-8 (b). Cell count plot of a synthetic Poisson process: time unit = 10 seconds

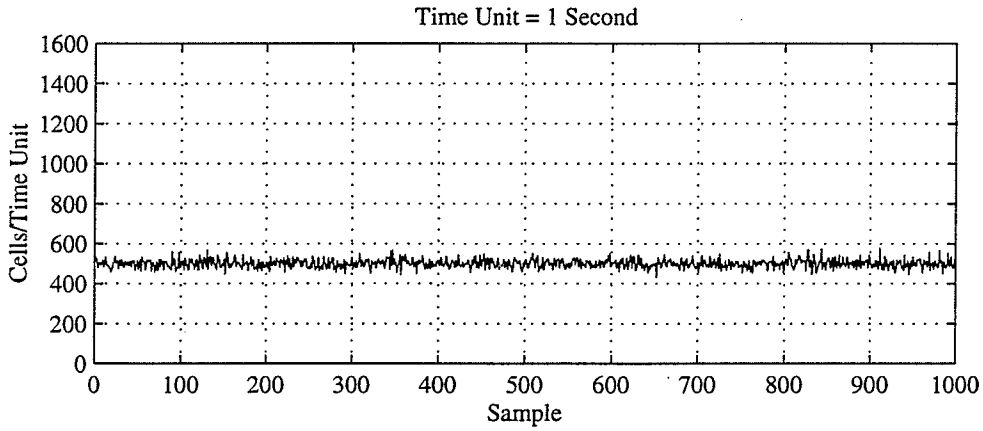

Figure 3-8 (c). Cell count plot of a synthetic Poisson process: time unit = 1 second

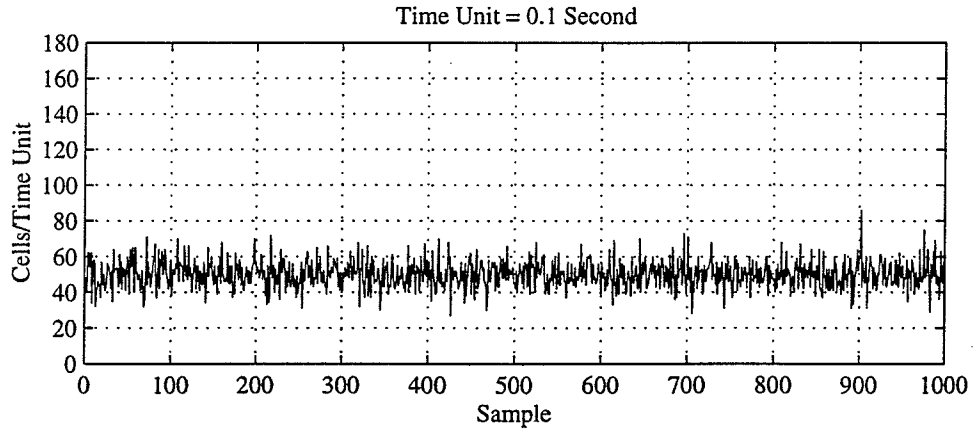

Figure 3-8 (d). Cell count plot of a synthetic Poisson process: time unit = 0.1 second

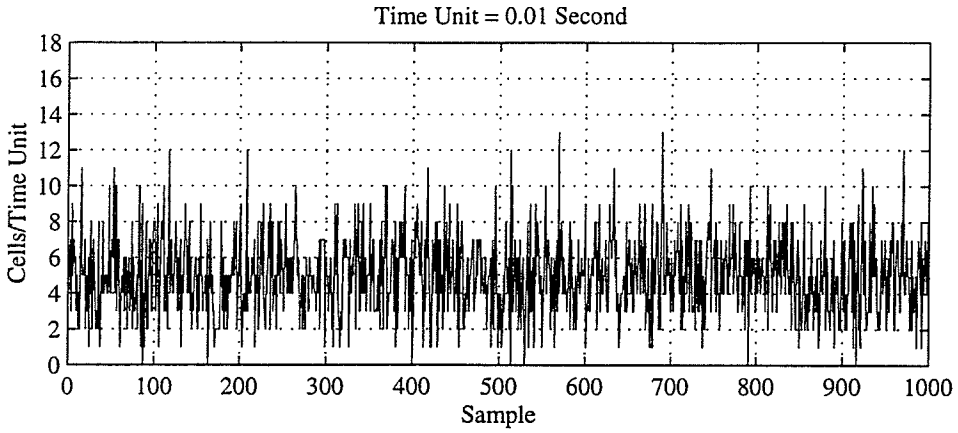

Figure 3-8 (e). Cell count plot of a synthetic Poisson process: time unit = 0.01 second

3.3.2 Definition and Properties of A Discrete-Time Self-Similar Stochastic Process

The definition and properties of a discrete-time self-similar stochastic process can be found in [LELA94]. Let $X=(x_t: t=1,2,3,\ldots)$ be a discrete-time stationary stochastic process with mean μ, variance σ^2 and an autocorrelation function $r(k)$, $k \geq 0$.

Assume that the autocorrelation function is of the form:

$$r(k) = k^{-\beta} L(t), \quad \text{as } k \to \infty,$$

where $0 < \beta < 1$ and $L(t)$ is slowly varying function. Let $X^m = (x_k^m: k=1,2,3,\ldots)$ denote the new stationary time series obtained by averaging the original series X over non-overlapping blocks of size m. That is, for each $m=1,2,3,\cdots$, X^m is given by

$$x_k^m = \frac{1}{m} \sum_{i=km-m+1}^{km} x_i, \quad k \geq 1.$$

Let $r^m(k)$ be the autocorrelation function of X^m. The process X is called exactly second-order self-similar with Hurst parameter (or the degree of self-similarity) $H = 1 - \beta / 2$ if for all $m = 1,2,3,\ldots,$

$$\mathrm{var}(X^m) = \sigma^2 m^{-\beta}$$

and

$$r^m(k) = r(k), \quad k \geq 0.$$

X is called asymptotically second-order self-similar with Hurst parameter $H = 1 - \beta / 2$ if

$$\mathrm{var}(X^m) \sim \sigma^2 m^{-\beta}, \quad \text{as } m \to \infty,$$

and for all k large enough,

$$r^m(k) \to r(k), \quad \text{as } m \to \infty.$$

In other words, X is exactly or asymptotically second-order self-similar if the aggregated process X^m are the same as X or become indistinguishable from X with respect to their autocorrelation functions.

The most striking feature of a discrete-time self-similar stochastic process is that the correlation structure of their aggregated processes X^m does not degenerate as $m \to \infty$. This is in stark contrast to traditional traffic models currently considered in the literature, all of which have the property that their aggregated processes X^m tend to second-order pure noise, i.e., for all $k \geq 1$,

$$r^m(k) \to 0, \quad \text{as } m \to \infty.$$

Self-similar processes provide an elegant explanation of an empirical law that is commonly referred as the Hurst effect. For a given set of observations $x_1, x_2, \ldots, x_n$ with

sample mean $\bar{x}(n)$ and sample variance $S^2(n)$, the rescaled adjusted range statistic (or R/S statistic) is given by

$$R(n)/S(n) = 1/S(n)[\max(0, W_1, W_2, \ldots, W_n) - \min(0, W_1, W_2, \ldots, W_n)],$$

with $W_k = x_1 + x_2 + \cdots + x_k - k\bar{x}(n)$, $1 \le k \le n$. While many naturally occurring time series appear to be well represented by the relation:

$$E[R(n)/S(n)] \sim cn^H, \quad \text{as } n \to \infty,$$

with Hurst parameter "typically" around 0.7, observations from a short range dependent model are known to satisfy :

$$E[R(n)/S(n)] \sim dn^{0.5}, \quad \text{as } n \to \infty,$$

with Hurst parameter "typically" around 0.5.

3.3.3 Hurst Parameter Estimation

There are several methods known to measure the Hurst parameter H of a time series [LERA94]. Two of them, variances time plots and R/S analysis, are described here.

3.3.3.1 Variance-Time Plots

We have seen in section 3.3.2 that for a self-similar process, the relation between the variance of the aggregated processes X^m and the block size m is given as

$$\text{var}(X^m) \sim \sigma^2 m^{-\beta}, \quad \text{as } m \to \infty.$$

Taking logarithm at both sides of the above relation, we have

$$\ln[\text{var}(X^m)] \sim -\beta \ln(m) + \ln(\sigma^2), \quad \text{as } m \to \infty.$$

The so-called variance-time plots are obtained by plotting $\ln[\mathrm{var}(X^m)]$ against $\ln(m)$. The coefficient $-\beta$ is estimated as the asymptotic slope of the variance-time curve by fitting a straight line through the sample points of the curve according to the least squares method, ignoring those points for small m. Values of the asymptotic slope between -1 and 0 suggest self-similarity, and the Hurst parameter is estimated as $\hat{H} = 1 - \hat{\beta}/2$. Figure 3-9 depicts the variance time curve of a synthetic self-similar traffic trace. It has an asymptotic slope that is distinctly different from -1 (dotted line) and the Hurst parameter is estimated as 0.84283. Figure 3-10 depicts the variance time curve of a synthetic Poisson process. The slope of the variance time curve matches the dotted line extremely well and the Hurst parameter is estimated as 0.51857.

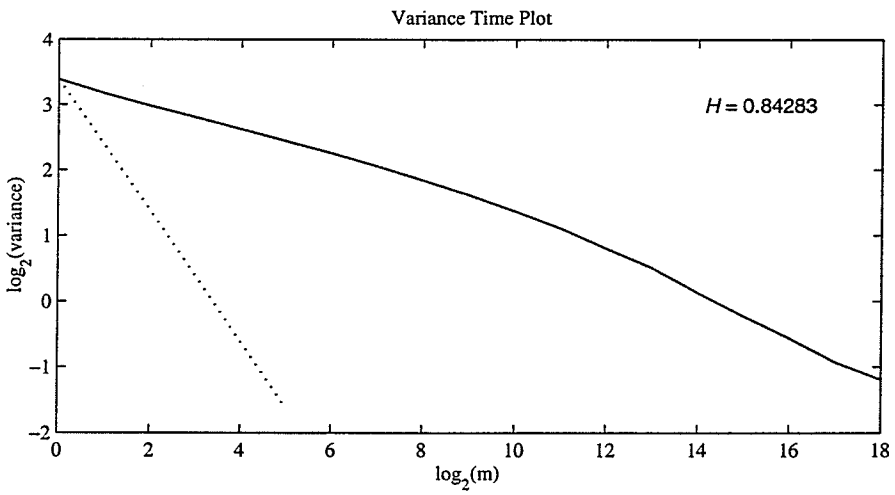

Figure 3-9. Variance time plot of a synthetic self-similar traffic

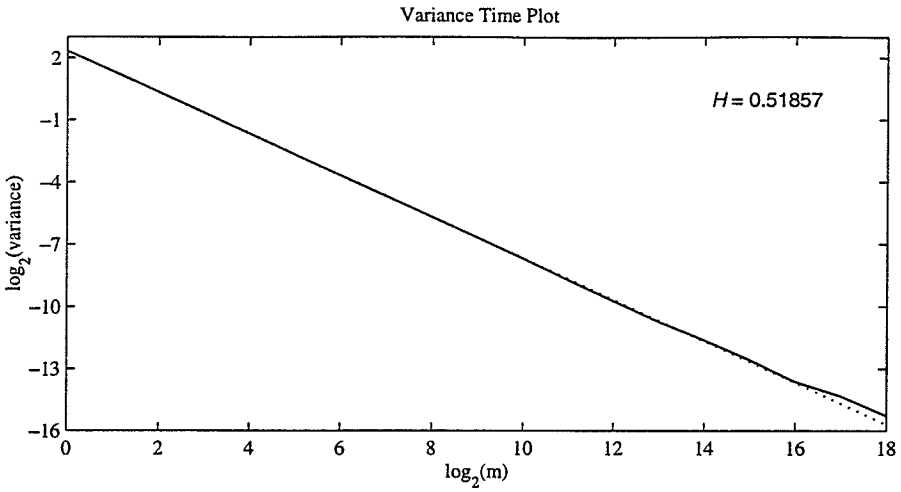

Figure 3-10. Variance time plot of a synthetic Poisson process

3.3.3.2 R/S Analysis

R/S analysis provides another way to estimate the Hurst parameter. We have seen in section 3.3.2 that for a self-similar process, the relation between the R/S statistic and the sample size n is given as

$$E[R(n)/S(n)] \sim cn^{H}, \quad \text{as } n \to \infty,$$

Taking logarithm at both sides of the above relation, we have

$$\ln\left[E[R(n)/S(n)]\right] \sim H\ln(n) + \ln(c), \quad n \to \infty$$

The so-called pox plot is obtained by plotting $\ln\left[E[R(n)/S(n)]\right]$ against $\ln(n)$. The Hurst parameter H is estimated as the asymptotic slope of the R/S curve by fitting a straight line through the sample points of the curve according to the least squares method, ignoring those points for small n.

3.3.4 Generating Self-Similar Traffic

Currently, there are a number of methods for generating self-similar traffic. Here, we briefly describe two of them. For a more detailed presentation and references on self-similar traffic generation, see [LELA94], [PAXS95a], [PAXS95b], [WILL97], [LAU95], and [GARR94].

The most intuitive method for generating self-similar traffic is to multiplex ON/OFF sources with heavy-tailed ON/OFF periods. This method comes directly from the fact that an individual Ethernet host can be modeled as an ON/OFF source and Ethernet LAN traffic is statistically self-similar. Traffic measurements also indicate that the lengths of the ON/OFF periods of an individual Ethernet host are heavy-tailed and can be fitted into Pareto distribution very well.

Self-similar traffic can also be generated by simulating the number of customers in an *M/G/∞* queuing system, where customers arrive according to a Poisson process and service times are drawn from a heavy-tailed distribution with infinite variance. Let X_t be the number of customers in the system at time *t*. The count process $(X_t : t = 1,2,\ldots)$ is asymptotically self-similar.

3.3.5 Implementation of A Self-Similar Traffic Generator

3.3.5.1 Model Structure

A self-similar traffic generator based on multiplexing a number of ON/OFF traffic sources with heavy-tailed ON/OFF periods is implemented as a set of process models in OPNET. The heavy-tailed distribution used in this implementation is the famous Pareto

distribution. The complexity of implementing such a self-similar generator with call setup and release capabilities is decomposed into two process models which are organized hierarchically. The first process model is for creating a heavy-tailed ON/OFF traffic source. The second process model is for multiplexing a number of heavy-tailed ON/OFF traffic sources as well as setting up and releasing calls. An auxiliary process model is designed to collect and analyze the synthetic traffic traces generated by both the self-similar traffic generator and the generalized traditional traffic generator. Packet counts at various time scales are provided as a set of vector statistics which can be viewed as packet count plots in OPNET Analysis Tool. Those packet count plots can be used for visually testing the self-similarity of collected synthetic traffic traces. This process model also estimates the PCR and SCR of a synthetic traffic trace.

3.3.5.2 Process Model for Creating A Heavy-Tailed ON/OFF Traffic Source

This section covers the process model for creating a heavy-tailed ON/OFF traffic source. The state transition diagram of the process model is shown in Figure 3-11. The process model has 14 model attributes as listed in Figure 3-12 (a)-(b). There are four states in the process model: one forced (init), and three unforced (on, off, and idle). They are described in Table 3-4. The process model has 5 transition conditions which are delineated in Table 3-5. Seven procedures are used in the enter and transition executives of the process model. Their descriptions are given in Table 3-6.

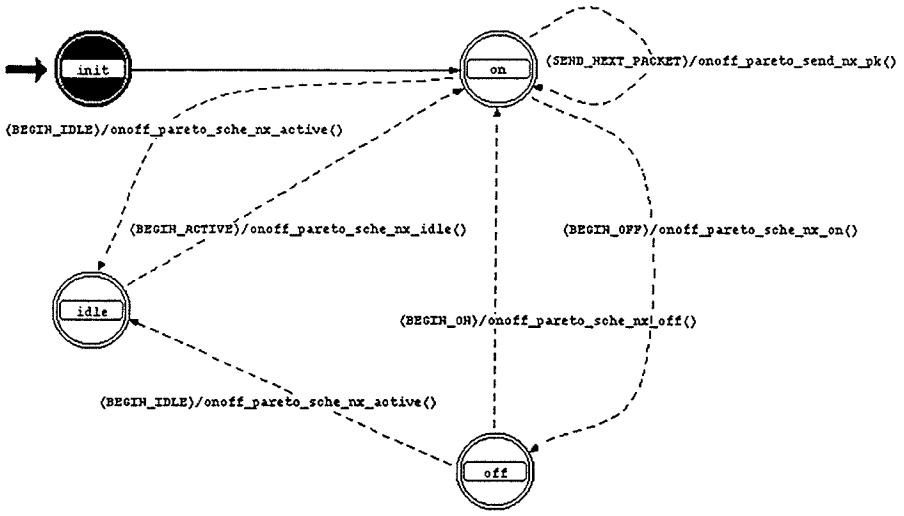

Figure 3-11. State transition diagram of the process model for creating a heavy-tailed ON/OFF traffic source

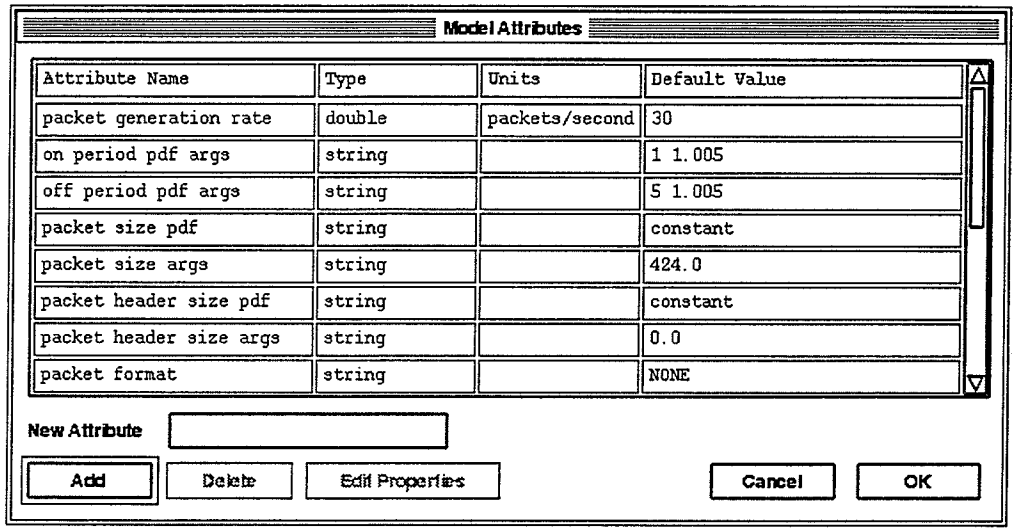

Figure 3-12 (a). Model attributes 1-8 of the process model for creating a heavy-tailed ON/OFF traffic source

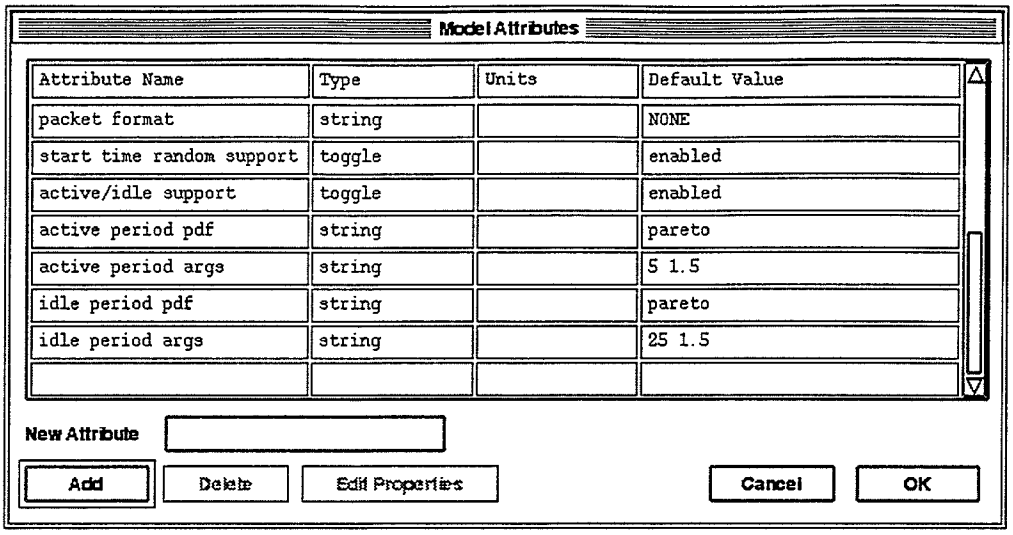

Figure 3-12 (b). Model attributes 8-14 of the process model for creating a heavy-tailed ON/OFF traffic source

State Descriptions

State	Description
init	The process starts at this state where user-supplied model attribute values are read and saved in corresponding state variables. A self interrupt is scheduled in a period of time in order to let the process start transmitting packets at the on state for the first time. If both the "random start time support" and the "active/idle support" model attributes are enabled, the length of this period of time is a random outcome from the distribution of the length of an idle period which can be specified by the "idle period pdf" and the "idle period args" model attributes. If the "random start time support" model attribute is enabled and the "active/idle support" model attribute is disabled, the length of this period of time is a random outcome from a Pareto distribution of the length of an off period with parameters specified in the "off period pdf args" model attribute. This feature is used to enable a number of heavy-tailed ON/OFF traffic sources to be independent to each other when traffic from them is multiplexed to produce self-similar traffic. If the "start time random support" model attribute is disabled, the process starts transmitting packets at the on state as soon as the initialization at the init state is completed. Another self interrupt is scheduled to let the process switch to the off state for the first time after it staying at the on state for a random period of time. The length of this random period of time is an outcome from a Pareto distribution of the length of an on period with parameters given in the "on period pdf args" model attribute. If the "active/idle support" model attribute is enabled, a third self interrupt is scheduled in a random period of time in order to let the traffic generator switch to the idle mode from the active mode for the first time. The length of this random period

	of time is an outcome from the distribution of the length of an active period which can be specified by the "active period pdf" and the "active period args" model attributes.
on	This state is entered when an off period is over, or when an idle period is over if the "active/idle support" model attribute is enabled. At this state, the procedure "onoff_pareto_send_nx_pk()" is repeatedly called and packets are generated at a constant rate which can be specified by the "packet generation rate". The traffic generator stays at this state for a random period of time before a transition to the off state takes place, unless a "begin an idle period" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of a Pareto distribution of the length of an on period with parameters specified in the "on period pdf args" model attribute.
off	This state is entered when an on period is over. The traffic generator stays at this state for a random period of time before a transition to the on state takes place, unless a "begin an idle period" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of a Pareto distribution of the length of an off period with parameters specified in the "off period pdf args" model attribute.
idle	This state is entered when an active period is over if the "active/idle support" model attribute is enabled. The process stays at this state for a random period of time before it starts another active period and switches to the on state. The length of this random period of time is an outcome of the distribution of the length of an idle period which can be specified by the "idle period pdf " and the "idle period args" model attributes.

Table 3-4. State descriptions of the process model for creating a heavy-tailed ON/OFF traffic source

Transition Condition Descriptions

Transition Condition	Description
SEND_NEXT_PACKET	A packet transmission is completed. It is time for the process to start transmitting another packet at the on state.
BEGIN_ON	An off period if over. It is time for the process to begin an on period and to switch to the on state from the off state.
BEGIN_OFF	An on period if over. It is time for the process to begin an off period and to switch to the off state from the on state.
BEGIN_ACTIVE	An idle period is over. It is time for the process to begin an active period and to switch to the on state from the idle state.

BEGIN_IDLE	An active period is over. It is time for the process to begin an idle period and to switch to the idle state from either the on state or the off state.

Table 3-5. Transition condition descriptions of the process model for creating a heavy-tailed ON/OFF traffic source

Procedure Descriptions

Procedure	Description
onoff_pareto_string_to_args(src_string, arg0, arg1)	This procedure decomposes a model attribute of type "string" into one or two arguments of type "double".
onoff_pareto_send_nx_pk()	This procedure creates a packet of the format specified by the "packet format" model attribute, and of the length specified by the "packet size pdf" and the "packet size args" model attributes. The length of the header of a packet can be specified by the "packet header size pdf" and the "packet header size args" model attributes. It then sends the packet to an output stream. It also schedules a "send next packet" self interrupt in a fixed period of time unless the process switches to the off or the idle state before this interrupt takes place. The length of this fix period of time is set as the inverse of the value specified in the "packet generation rate" model attribute.
onoff_pareto_sche_nx_active()	This procedure schedules a self interrupt in a random period of time in order to let the process switch back to the active mode from the idle mode. The length of this random period of time is an outcome of the distribution of the length of an idle period which can be specified by the "idle period pdf" and the "idle period args" model attributes.
onoff_pareto_sche_nx_idle()	This procedure schedules a self interrupt in a random period of time in order to let the process switch back to the idle mode from the active mode. The length of the random period of time is an outcome of the distribution of the length of an active period which can be specified by the "active period pdf" and the "active period args" model attributes. The event time of this interrupt

	is saved in a state variable called "scheduled_idle_time" which is used in the procedure "onoff_pareto_send_nx_pk()" to prevent scheduling another "send next packet" self interrupt at the on state, and in the procedures "onoff_pareto_sche_nx_on()" and "onoff_pareto_sche_nx_off()" to prevent scheduling another "begin an on period" at the off state, or another "begin an off period" self interrupt at the on state, when the process is about to switch to the idle state.
onoff_pareto_sche_nx_on()	This procedure schedules a self interrupt in a random period of time in order to let the process switch back to the on state from the off state, unless a "begin an idle period" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of a Pareto distribution of the length of an off period with parameters specified in the "off period pdf args" model attribute.
onoff_pareto_sche_nx_off()	This procedure schedules a self interrupt in a random period of time in order to let the process switch back to the off state from the on state, unless a "begin an idle period" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of a Pareto distribution of the length of an on period with parameters specified in the "on period pdf args" model attribute. The event time of this interrupt is saved in a state variable called "scheduled_off_time" which is used in the procedure "onoff_pareto_send_nx_pk()" to prevent scheduling another "send next packet" event at the on state when the process is about to switch back to the off state.
onoff_pareto_dist(alpha, beta)	This procedure produces a random outcome of a Pareto distribution with two parameters: alpha and beta. The first parameter alpha is the mean of the outcome. The second parameter is the shape parameter of the Pareto distribution.

Table 3-6. Procedure descriptions of the process model for creating a heavy-tailed ON/OFF traffic source

3.3.5.3 Process Model for Multiplexing A Number of Heavy-Tailed ON/OFF Traffic Sources

This section covers the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources. The state transition diagram of the process model is shown in Figure 3-13. The process model has 17 model attributes as listed in Figure 3-14 (a)-(c). There are six states in the process model: one forced (init), and five unforced (warm_up, call_setup, connected, released, and disabled). They are described in Table 3-7. The process model has 11 transition conditions which are delineated in Table 3-8. Nine procedures are used in the enter and transition executives of the process model. Their descriptions are given in Table 3-9.

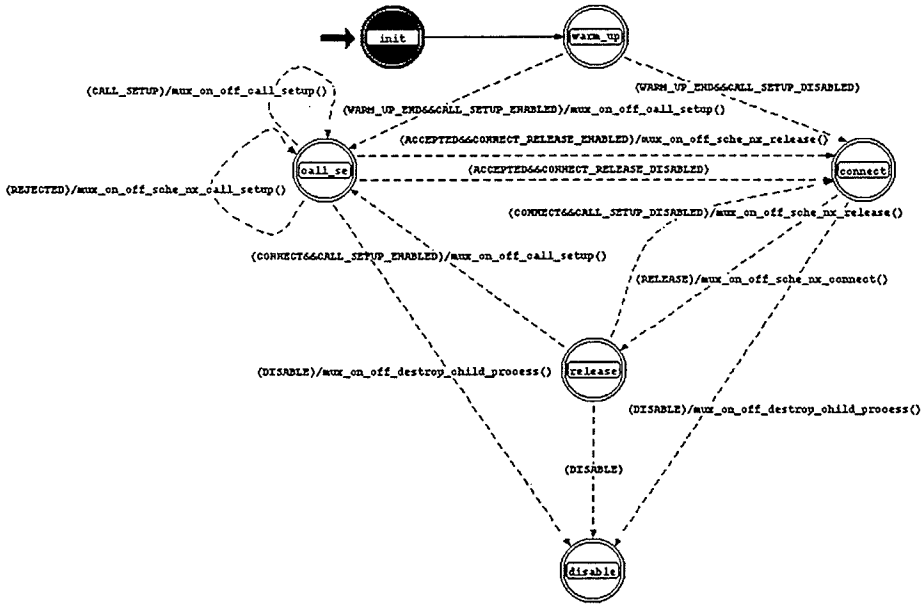

Figure 3-13. State transition diagram of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

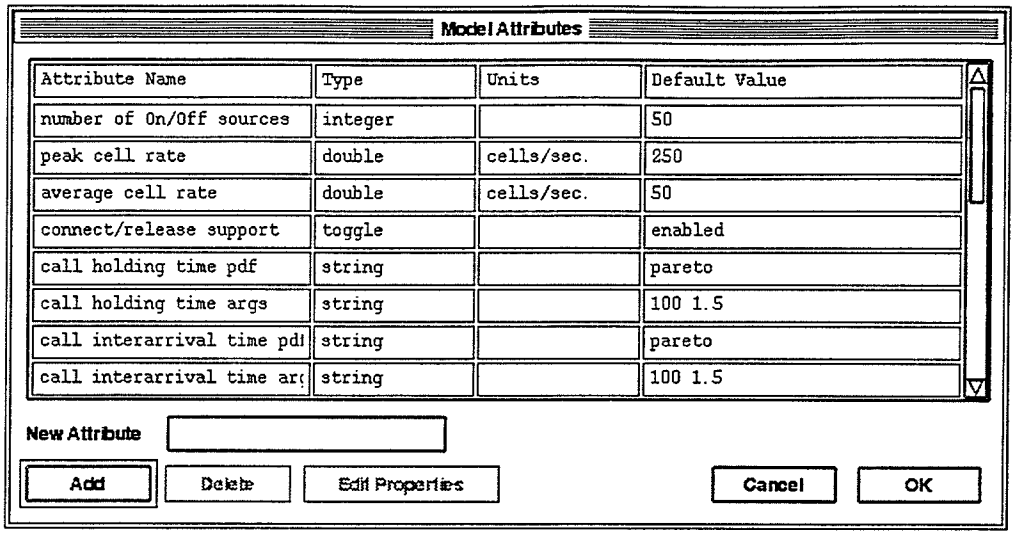

Figure 3-14 (a). Model attributes 1-8 of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

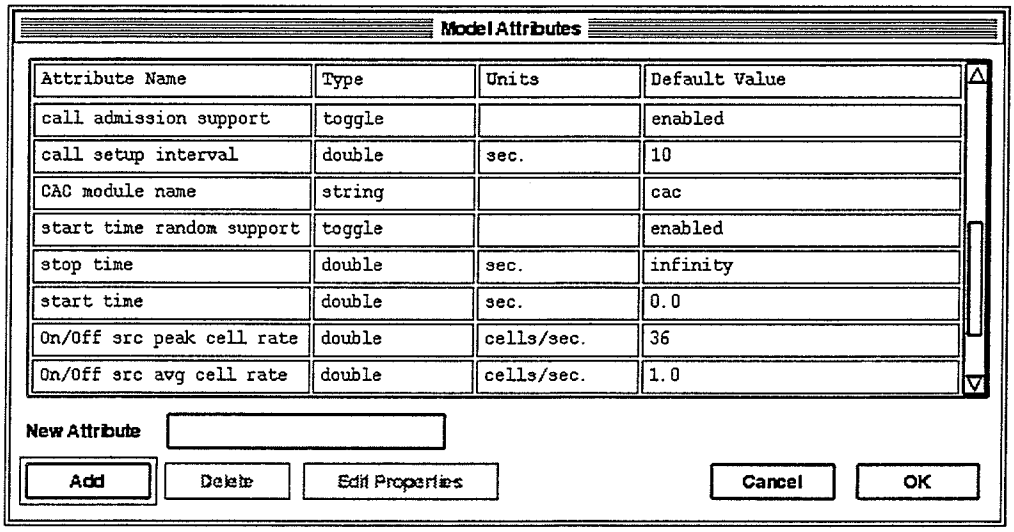

Figure 3-14 (b). Model attributes 9-18 of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

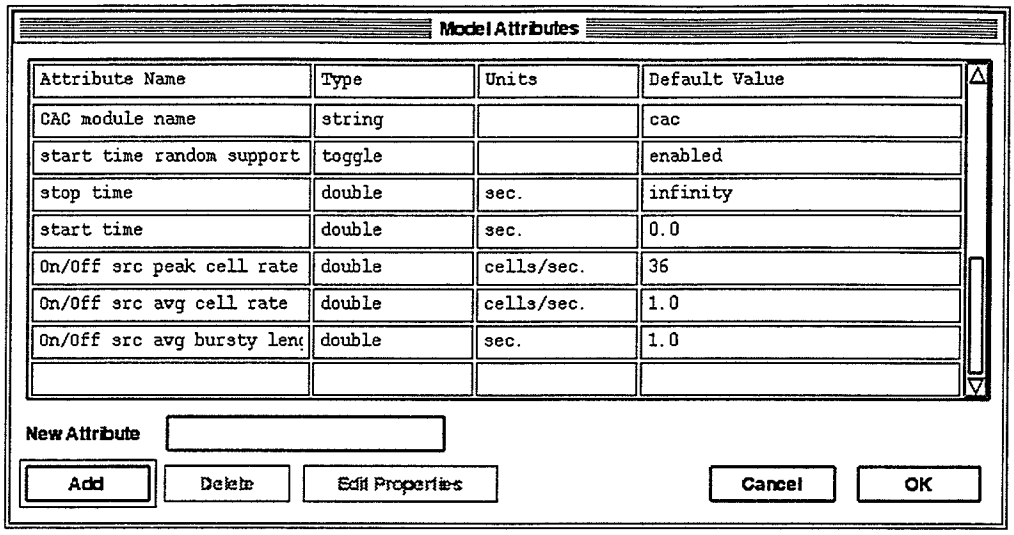

Figure 3-14 (c). Model attributes 11-17 of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

State Descriptions

State	Description
init	The process starts at this state where user-supplied model attribute values are read and saved in state variables. If the "connect/release support" model attribute is enabled and the "call admission support" model attribute is disabled, a future "release the ongoing call" self interrupt is scheduled in order to let the process switch to the released state from the connected state for the first time after a call is being connected for a random period of time. The length of this random period of time is an outcome of the call holding time distribution which can be specified by the "call holding time pdf" and the "call holding time args" model attributes. A "warm_up_end" interrupt is also scheduled in a period time in order to let the process leave the warm_up state and start setting up a call or transmitting packets for the first time. If the "stop time" model attribute is set to be different from the default value "infinity", another self interrupt is scheduled in order to disable the traffic generator permanently at the time specified by the "stop time" model attribute.
warm_up	This state is entered after the initialization is completed at the init state. If both the "random start time support" and the "connect/release support" model attributes are enabled, the process stays at this state for a random period of time before setting up a call or transmitting any packets. The length of this random period of time is an outcome from the call interarrival time distribution which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model

	attributes. This feature is used to enable a number of self-similar traffic sources to be independent to each other when traffic from them is multiplexed. If the "start time random support" model attribute is disabled, the process stays at this state for a fixed period of time before setting up a call or transmitting any packets. The length of this fixed period of time can be specified by the "start time" model attribute. It can be set as zero which allows the traffic generator to start setting up a call or transmitting packets right after the simulation starts.
call_setup	This state is entered every time when the traffic generator wants to make a call if the "call admission support" model attribute is enabled. Before any packet transmission takes place, the process sends a call setup request with the call's traffic parameters to a neighboring CAC module and waits for the CAC module to send back an accept/reject notification. The PCR of the call can be specified by the "peak cell rate" model attribute. The SCR of the call can be specified by the "average cell rate" model attribute. The CAC module is identified by the "CAC module name" attribute. If the call setup request is accepted, the process moves to the connected state and starts transmitting packets. If the call setup request is rejected, the process remains at this state and waits for a fixed period of time before resending the request to the CAC module. The length of this fixed period of time can be specified by the "call setup interval" model attribute.
connected	This state is entered when a call setup request is accepted if the "call admission support" model attribute is enabled, or when an released period is over if the "connect/release support" model attribute is enabled and the "call admission support" model attribute is disabled. The procedure "mux_on_off_invoke_child_processes()" is called to invoke and multiplex a number of heavy tailed ON/OFF traffic sources. If the "connect/release support" model attributes is disabled, the process stays at this state for the rest of the simulation unless a "disable the traffic generator" self interrupt occurs during the rest of the simulation. If the "connect/release support" model attributes is enabled, the process stays at this state for a random period of time before a transition to the released state takes place unless a "disable the traffic generator" self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the call holding time distribution which can be specified by the "call holding time pdf" and the "call holding time args" model attributes.
released	This state is entered when a call holding time period is over if the "connect/release support" model attribute is enabled. The procedure "mux_on_off_destroy_child_processes()" is called to destroy a number of invoked heavy tailed ON/OFF traffic sources and the call is released. The process stays at this state for a random period of time before a transition to the call_setup state takes place if the "call admission support" model attribute is enabled, or a transition to the connected

	state takes place if the "call admission support" model attribute is disabled, unless a "disable the traffic generator' self interrupt occurs during this random period of time. The length of this random period of time is an outcome of the call interarrival time distribution which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model attributes.
disabled	This state is entered when the simulation reaches the time specified by the "stop time" model attribute when it is set to be smaller than the default value "infinity". The process then remains at this state for the rest of the simulation. No packets are generated after this state is entered. The traffic generator is practically disabled.

Table 3-7. State descriptions of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

Transition Condition Descriptions

Transition Condition	Description
WARM_UP_END	The warm-up period is over. It is time for the process to move to the call_setup state to start setting up a call if the "connect/release support" model attribute is enabled, or to move to the connected state to start transmitting packets if the "connect/release support" model attribute is disabled.
CALL_SETUP_ENABLED	The "call admission support" model attribute is enabled.
CALL_SETUP_DISABLED	The "call admission support" model attribute is disabled.
CALL_SETUP	A call setup interval is over. It is time for the process to resend the last call setup request to a neighboring CAC module.
ACCEPTED	A process interrupt is received from a neighboring CAC module. The last call setup request is accepted.
REJECTED	A process interrupt is received from a neighboring CAC module. The last call setup request is rejected.
CONNECT_RELEASE_ENABLED	The "connect/release support" model attribute is enabled.
CONNECT_RELEASE_DISABLED	The "connect/release support" model attribute is disabled.
CONNECT	A call interarrival time period is over. It is time for the process to move to the call_setup state to start setting up a call if the "connect/release support" model attribute is enabled, or to move to the connected

	state to start transmitting packets if the "connect/release support" model attribute is disabled.
RELEASE	A call holding time period is over. It is time for the process to move to the released state and release the ongoing call.
DISABLE	The stop time of this traffic generator is reached. It is time for the process to disable the traffic generator.

Table 3-8. Transition condition descriptions of the process model for multiplexing a number of heavy-tailed ON/OFF sources

Procedures Descriptions

Procedure	Description
mux_onoff_string_to_args(src_string, arg0, arg1)	This procedure decomposes a model attribute of type "string" into one or two arguments of type "double".
mux_onoff_sche_nx_release()	This procedure schedules a self interrupt in a random period of time in order to let the traffic generator release an ongoing call and switch back to the released state from the connected state. The length of this random period of time is an outcome of the call holding time distribution which can be specified by the "call holding time pdf" and the "call holding time pdf args" model attributes.
mux_onoff_sche_nx_connect()	This procedure schedules a self interrupt in a random period time in order to let the traffic generator make a new call and switch back to the connected state from the released state. The length of this random period of time is an outcome of the call interarrival time distribution which can be specified by the "call interarrival time pdf" and the "call interarrival time args" model attributes.
mux_onoff_invoke_child_processes()	This procedure invokes a number of heavy-tailed ON/OFF traffic sources. The number of heavy-tailed ON/OFF traffic sources being invoked by this process can be specified by the "number of on/off sources" model attribute. Each child process is assigned a process handle as an identification which is used in the procedure

	"mux_onoff_destroy_child_processes()" where those child processes are destroyed.
mux_onoff_destroy_child_processes()	This procedure destroys a number of heavy-tailed ON/OFF traffic sources which were invoked by the procedure "mux_onoff_invoke_child_processes()".
mux_onoff_call_setup()	This procedure sends a forced remote interrupt to notify a neighboring CAC module that the traffic generator wants to make a call. It also sends the traffic generator's process identification, the call's PCR, SCR, and other traffic parameters to the CAC module. Those traffic parameters are used by the CAC module to decide the acceptance of this call. The traffic generator's process identification is used by the CAC module to identify the call if the call is accepted.
mux_onoff_sche_nx_call_setup()	This procedure schedules a self interrupt in a fixed period of time in order to let the traffic generator to resend the last call setup request to a neighboring CAC module when the last request was rejected. The length of this fixed period of time can be specified by the "call setup interval" model attribute.
mux_onoff_call_release()	This procedure sends a forced remote interrupt to notify the neighboring CAC module that the traffic generator wants to release its call. It also sends the traffic generator's process identification to the CAC module so that the CAC module can locate and delete the record of this call from a list of the connected calls.
mux_onoff_pareto_dist(alpha, beta)	This procedure produces a random outcome of a Pareto distribution with two parameters: alpha and beta. The first parameter alpha is the mean of the outcome. The second parameter is the shape parameter of the Pareto distribution.

Table 3-9. Procedure descriptions of the process model for multiplexing a number of heavy-tailed ON/OFF traffic sources

3.3.5.4 Process Model for Collecting And Analyzing Synthetic Traffic Traces

This section covers the process model for collecting and analyzing synthetic traffic traces. The state transition diagram of the process model is shown in Figure 3-15. The process model has 12 model attributes as listed in Figure 3-16 (a)-(b). Visual tests of the self-similarity of collected synthetic traffic traces are based on a sequence of packet count plots for five different time scales. The number of sample points in each packet count plot can be specified by the "number of samples per time scale" model attribute. The unit of each time scale can be specified by the "time unit" model attribute with a corresponding Roman number suffix. The sampling start time for the largest time scale, "time unit I", can be specified by the "sampling start time". The sampling start time for each subsequent time scale is a random outcome between the sampling starting time and the sampling stop time of the previous time scales. The length of the traffic trace, which is going to be collected, can be specified by the "traffic trace length" model attribute. Each entry in the traffic trace is the number of packets generated in the smallest time unit, time unit V. The process starts to collect a synthetic traffic trace at the time specified by the "trace collecting start time" model attribute. The name of the traffic trace data file can be specified by the "trace data file name" model attribute. For each time scale, a packet count data file will be created, with a file name having the corresponding Roman number suffix appended to a base file name. The base file name can be specified by the "pkcnt base file name" model attribute. PCR of a synthetic traffic trace is estimated as the maximum count of the number of packets generated per second. The counting process starts at the time specified by the "PCR estimation start time" model attributes. The process model has five states: three forced (init, arrival, and write_stat) and two

unforced (idle and sim_end). They are described in Table 3-10. The process model has 3 transition conditions which are delineated in Table 3-11. No procedures are defined in this process model.

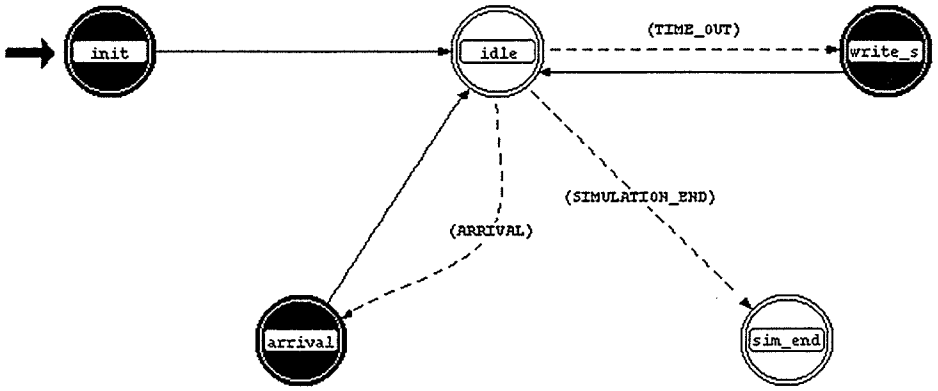

Figure 3-15. State transition diagram of the process model for collecting and analyzing synthetic traffic traces

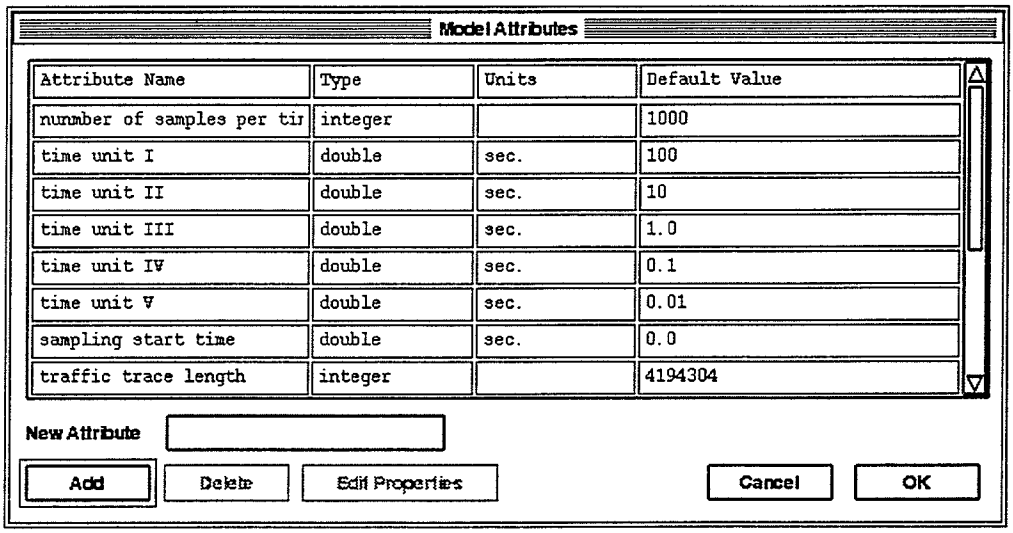

Figure 3-16. (a) Model attributes 1-8 of the process model for collecting and analyzing synthetic traffic traces

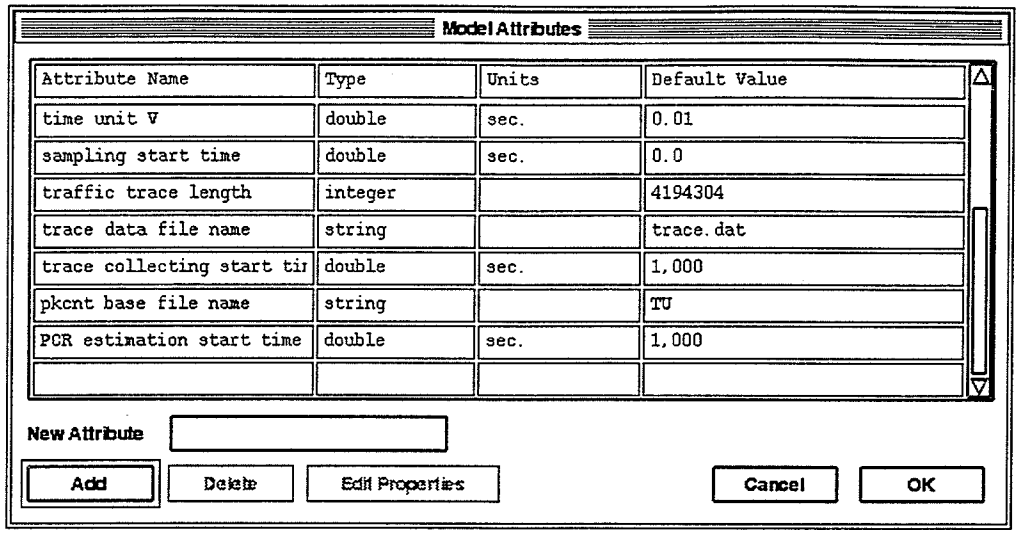

Figure 3-16. (b) Model attributes 6-12 of the process model for collecting and analyzing synthetic traffic traces

State Descriptions

State	Description
init	The process begins at this state where user-supplied model attribute values are read and saved in corresponding state variables. Seven self interrupts are scheduled in this state. Five of them are used to mark the sampling start time of each time scale. The other two are used to mark the start time for collecting a synthetic traffic trace and the start time for counting the number of packets received per second, respectively. Six data files are created and opened. Five of them are the packet count data files for each time scale. The other one is the traffic trace data file.
arrival	This state is entered when a packet is received. The number of packets received by this process is incremented by one. Related packet counters are updated if the simulation time is in their corresponding sampling time intervals.
write_stat	This state is entered when a self-interrupt is received. If the self interrupt marks the sampling start time of a time scale, several counters for this time scale are initialized and a self interrupt is scheduled in the unit of this time scale to mark the end of the first sample time interval for this time scale. If the self interrupt marks the end of a sample time interval of a time scale and the number of packet count samples for this time scale is less than the value specified by the "number of samples per time scale" model attribute, several counters for this time scale are updated, and the packet count is saved into the corresponding packet count data file, and a self interrupt is scheduled in the unit of this time scale to mark the end of another sample time interval for this time scale. If the

	self interrupt marks the start time for collecting a synthetic traffic trace, several counters for the synthetic traffic trace are initialized and a self interrupt is scheduled in the smallest time unit, time unit V, to mark the end of the first traffic trace sample time interval. If the self interrupt marks the end of a traffic trace sample time interval and the length of the traffic trace is less than the value specified by the "traffic trace length" model attribute, several counters for the synthetic traffic trace are updated, and the packet count is saved into the traffic trace data file, and a self interrupt is scheduled in the smallest time unit, time unit V, to mark the end of another traffic trace sample time interval. If the self interrupt marks the start time for counting the number of packets received per second, several counters for estimating the PCR and SCR of the synthetic traffic trace are initialized and a self interrupt is scheduled in 1 second to mark the end of the first "packet count per second" sample time interval. If the self interrupt marks the end of a "packet count per second" sample time interval, several counters for estimating the PCR and SCR of the synthetic traffic trace are updated, and a self interrupt is scheduled in 1 second to mark the end of another "packet count per second" sample time interval.
idle	This state is entered right after the process completes all the actions at one of the forced states. The process rests in this state and waits to be invoked again.
sim_end	This state is entered just before the end of the simulation. The peak cell rate and the average cell rate of the synthetic traffic trace are estimated and printed out to the standard output.

Table 3-10. State descriptions of the process model for collecting and analyzing synthetic traffic traces

Transition Condition Descriptions

Transition Condition	Description
ARRIVAL	A packet is received by this process.
TIME_OUT	A self interrupt is received.
SIMULATION_END	The simulation is either completed or stopped by the user.

Table 3-11. Transition condition descriptions of the process model for collecting and analyzing synthetic traffic traces

3.3.5.5 Effects of Some Model Attributes of the Self-Similar Traffic Generator on the Hurst Parameters of Collected Synthetic Self-Similar Traffic Traces

In this section, we study the effects of some model attributes of the self-similar traffic generator on the Hurst parameters of collected synthetic self-similar traffic traces. Those model attributes include the variability of the ON/OFF periods of each individual heavy-tailed ON/OFF source, the number of heavy-tailed ON/OFF sources being multiplexed, and the utilization of each individual heavy-tailed ON/OFF source. The utilization of each individual heavy-tailed ON/OFF traffic source, ρ, is defined as the fraction of time that the traffic source is at the ON state. If the "active/idle support" model attribute is enabled for each individual traffic source, ρ is calculated as:

$$\rho = \frac{E(active)}{E(active) + E(idle)} \times \frac{E(on)}{E(on) + E(off)},$$

where *E(active)* is the average length of an active period, *E(idle)* is the average length of an idle period, *E(on)* is the average length of an on period, and *E(off)* is the average length of an off period.

If the "active/idle support" model attribute is disabled for each individual traffic source, ρ is calculated as:

$$\rho = \frac{E(on)}{E(on) + E(off)}.$$

The average cell rate of the self-similar traffic, which is the result of multiplexing a number of heavy-tailed ON/OFF sources, can be obtained as:

$$SCR_{sekf} = N_{onoff} \times Rate_{on} \times \rho,$$

where SCR_{self} is the average cell rate of the self-similar traffic, N_{onoff} is the number of heavy-tailed ON/OF sources being multiplexed, $Rate_{on}$ is the cell generation rate at the ON state of each individual heavy-tailed ON/OFF source.

3.3.5.5.1 Effect of the Variability of the ON/OFF Periods of Each Individual Heavy-Tailed ON/OFF Source

As discussed in Chapter two, the variability of the ON (or OFF) periods of each individual heavy-tailed ON/OFF source is controlled by the shape parameter β_1 (or β_2) of the corresponding Pareto distribution. The closer the shape parameter gets to 1, the higher the variability of a finite sample set of those periods. Figure 3-17 is obtained by varying the variability of the ON/OFF periods of each individual traffic source when the number of traffic sources and the utilization of each individual traffic source are fixed. The figure indicates that the Hurst parameter of a synthetic traffic traces produced by the self-similar traffic generator decreases as the variability of both ON/OFF periods decreases.

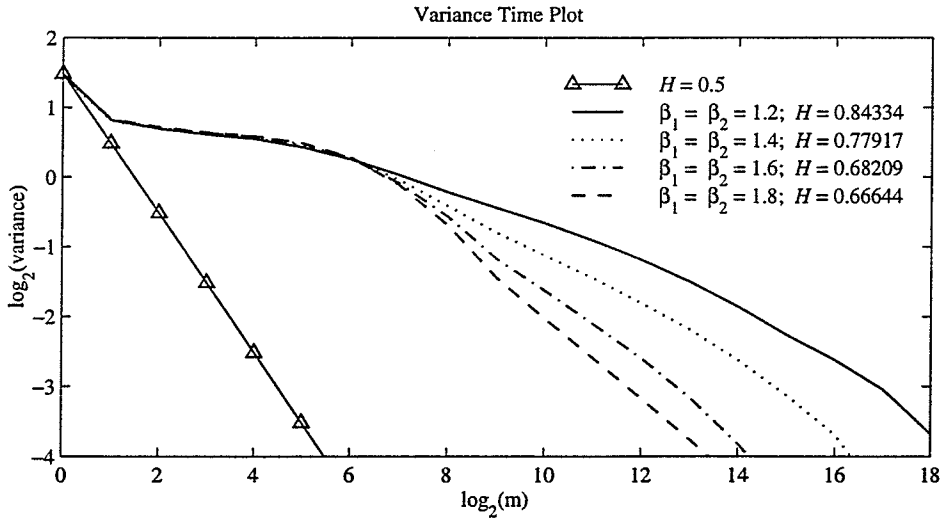

Figure 3-17. Variability of the ON/OFF periods of each individual heavy-tailed ON/OFF source vs. Hurst parameter

3.3.5.5.2 Effect of the Number of Heavy-Tailed ON/OFF Sources Being Multiplexed

Effect of the number of heavy-tailed ON/OFF sources being multiplexed on the Hurst parameters of collected synthetic traffic traces is studied in two cases. In the first case, the cell generation rate at the ON state of each individual heavy-tailed ON/OFF source is fixed. As a result, the average cell rate of the self-similar traffic increases as the number of heavy-tailed ON/OFF sources increases. In the second case, the average cell rate of the self-similar traffic is fixed. As a result, the cell generation rate at the ON state of each individual heavy-tailed ON/OFF source decreases as the number of heavy-tailed traffic sources increases. In both cases, the variability and utilization of each individual heavy-tailed ON/OFF source are fixed. Results from both cases indicate that the number of heavy-tailed ON/OFF sources has no significant effect on the Hurst parameters of collected synthetic traffic traces as shown in Figure 3-18(a) and Figure 3-18(b).

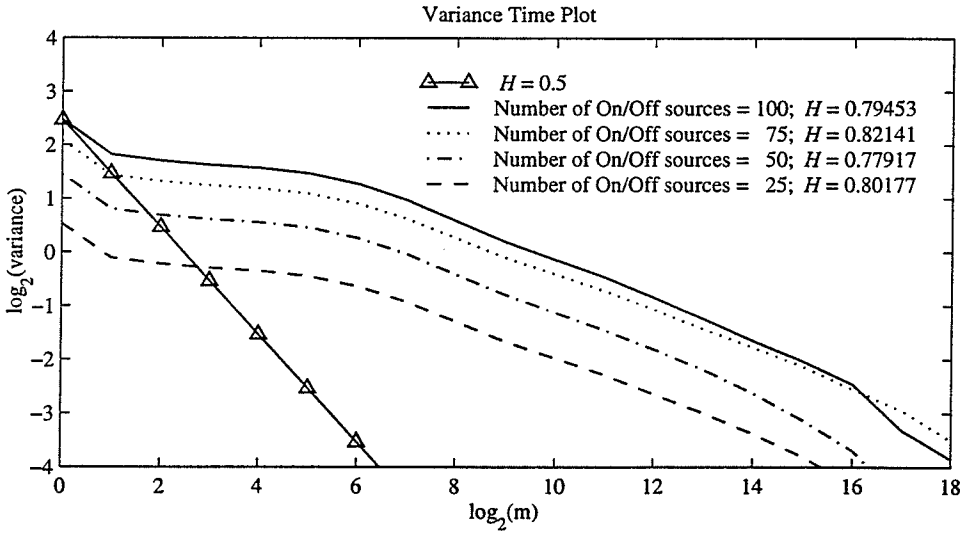

Figure 3-18 (a). Number of heavy-tailed ON/OFF sources vs. Hurst parameter where the cell generation rate at the ON state of each individual source is fixed

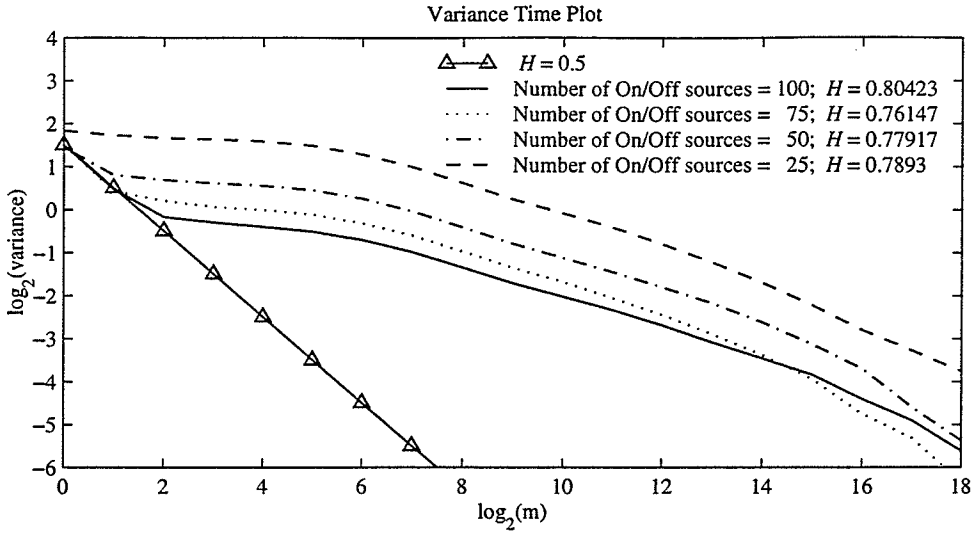

Figure 3-18 (b). Number of heavy-tailed ON/OFF sources vs. Hurst parameter where the average cell rate of the self-similar traffic is fixed

3.3.5.5.3 Effect of the Utilization of Each Individual Heavy-Tailed ON/OFF Source

Results presented here are obtained by disabling the "active/idle support" model attribute for each individual heavy-tailed ON/OFF source. The effect of the utilization of each individual heavy-tailed ON/OFF source on the Hurst parameters of collected synthetic traffic traces is studied in two cases. In the first case, the cell generation rate at the ON state of each individual heavy-tailed traffic source is fixed. As a result, the average cell rate of the self-similar traffic increases as the utilization of each individual heavy-tailed traffic source increases. In the second case, the average cell rate of the self-similar traffic is fixed. As a result, the cell generation rate at the ON state of each individual heavy-tailed traffic source decreases as the utilization of each individual heavy-tailed traffic source increases. In both cases, the number of heavy-tailed ON/OFF sources being multiplexed and the variability of each individual source are fixed. Results from the first case indicate that the utilization of each individual heavy-tailed ON/OFF source has no

significant effect on the Hurst parameters of collected synthetic traffic traces produced as shown in Figure 3-19 (a). Results from the second case indicate that the Hurst parameters of collected synthetic traffic traces generally decreases as the utilization of each individual heavy-tailed ON/OFF source decreases as shown in Figure 3-19 (b).

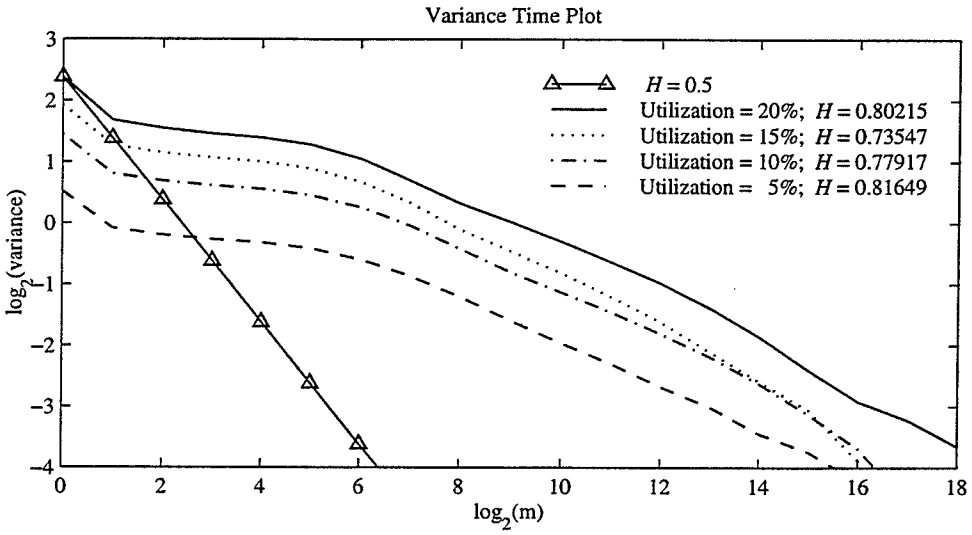

Figure 3-19 (a). Utilization of each individual heavy-tailed ON/OFF source vs. Hurst parameter where the cell generation rate at the ON state of each individual source is fixed

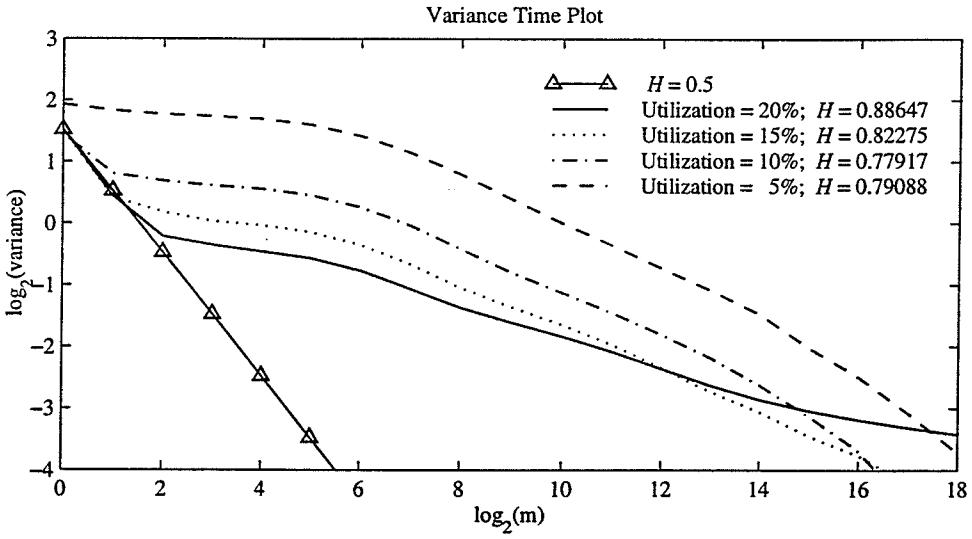

Figure 3-19 (b). Utilization of each individual heavy-tailed ON/OFF source vs. Hurst parameter where the average cell rate of the self-similar traffic is fixed

3.4 SUMMARY

In this chapter, the implementations of a generalized traditional traffic generator and a self-similar traffic generator as process models in OPNET are described. They can be used to generate ATM cells as well as packets of various formats. They also have the call setup and release capabilities and can be used directly in performance evaluations of ATM traffic control schemes. Visual tests are performed on the generated self-similar traces and Hurst parameter are estimated using variance time plots. The impacts of different attributes of the self-similar traffic generator on the Hurst parameter of collected synthetic traffic traces are also investigated. It is found that the Hurst parameter of collected synthetic traffic traces produced by the self-similar traffic generator decreases as the variability of both ON/OFF periods decreases. Results from both cases indicate that the number of heavy-tailed ON/OFF sources has no significant effect on the Hurst parameters of collected synthetic traffic traces. Results from the first case indicate that the utilization of each individual heavy-tailed ON/OFF source has no significant effect on the Hurst parameters of collected synthetic traffic traces produced as shown in Figure 3-19 (a). Results from the second case indicate that the Hurst parameters of collected synthetic traffic traces generally decreases as the utilization of each individual heavy-tailed ON/OFF source decreases.

CHAPTER FOUR

LITERATURE REVIEW ON CAC SCHEMES

CLR, maximum CTD, and peak-to-peak CDV are often adopted as measures of QoS for ATM services. Maximum CTD and peak-to-peak CDV can usually be controlled within a desired bound by engineering the buffer sizes in ATM switches [SAIT94]. Therefore, most CAC schemes found in the literature focus on fulfilling the CLR QoS requirement. Some of the most important features of the ATM technology are to support various types of services, to utilize network resources efficiently via statistical multiplexing, and to minimize the processing time at each switch in order to provide a very high data rate. As an indispensable component of the ATM technology, a CAC scheme should also keep all these valuable features. The objectives of a CAC scheme can be summarized as follows:

1. Robustness: The characteristics of a traffic source usually cannot be fully described by a set of traffic parameters. Traffic sources with the same traffic parameter values may have different

impacts on the ATM networks. However, CAC decisions have to be made under the condition that only traffic parameter values are given. Therefore, CAC schemes should be designed to achieve the predefined QoS requirements under any possible traffic circumstances, as long as each traffic source complies with its declared traffic parameter values.

2. Simplicity: As in the traditional telephone networks, call setup requests may arrive in high density in ATM networks. CAC decisions have to be made quickly. Therefore, CAC schemes should be designed to be simple enough to operate in real-time.
3. Efficiency: Protecting the network from congestion is the basic requirement of a CAC scheme. The force behind intensive research on CAC schemes is to use network resources efficiently. CAC schemes should be designed to accept as many calls as possible without violating the more important objective of robustness.
4. Flexibility: Unlike traditional dedicated networks, ATM networks have calls of various traffic types. CAC schemes should be able to operate on a fundamental set of traffic parameters regardless of the traffic type of each call. This allows CAC schemes to operate on calls of new traffic types, as well as calls of existing traffic types, as long as calls of new traffic types are described by the same set of traffic parameters.

A substantial number of CAC schemes have been proposed in the literature. However, None of them can simultaneously achieve all those somewhat conflicting objectives mentioned above. In this chapter, some of these schemes are reviewed.

4.1 PEAK BANDWIDTH RESERVATION

The simplest and most robust method to limit the CLR is to reserve the peak bandwidth for each connection. A new call is only admitted if the sum of the peak cell rates of all existing connections, and that of the new call, is not greater than the capacity of the output link. This guarantees that congestion and cell loss will never occur. However, for bursty traffic sources, this scheme makes inefficient use of the output link since statistical multiplexing is not exploited. This scheme also relies on users having accurate PCR specifications. Even more bandwidth may be wasted if users over-specify this traffic parameter. The peak bandwidth reservation CAC scheme is considered only as a lower bound for other CAC schemes aiming to achieve higher output link utilization[TRAN92].

4.2 EFFECTIVE BANDWIDTH RESERVATION

The concept of effective bandwidth was first proposed in [GUER91], where expressions for the effective bandwidth of ON/OFF sources were given. Different expressions for the effective bandwidth of ON/OFF sources were documented in [COST94]. Expressions of the effective bandwidth of general Markovian traffic sources, including MMFP and MMPP, can be found in [ELWA93] and [KESI93]. An overview on effective bandwidth using the theory of large deviations was provided in [CHAN95], where a calculus of effective bandwidth related operations was introduced. Effective bandwidth of leaky-bucket regulated VBR traffic sources was studied in [ELWA95]. Expressions of the effective bandwidth of MMFP sources with multiple CLR QoS requirements were obtained in [KULK95]. Estimating the effective bandwidth of VBR traffic sources using neural networks was suggested in [YOUS97]. An

effective bandwidth approach for CAC was reported in [FAN96], where aggregate traffic was approximated by a two state MMPP. The drawback of CAC schemes using effective bandwidth in some situations was discussed in [CHOU96].

4.2.1 Concept of Effective Bandwidth

The concept of effective bandwidth comes naturally from the fact that it is far too conservative to reserve peak bandwidth for each connection, and it is far too optimistic to reserve average bandwidth for each connection. The idea is to assign each connection an effective bandwidth whose value is between its peak and average bandwidth. The CLR QoS requirements of all connections are satisfied as long as the sum of the effective bandwidths of all accepted connections is not greater than the capacity of the output link [GUER91]. The effective bandwidth of each connection depends on the connection's characteristics, the connection's CLR QoS requirement, the capacity of the output link, and size of the output buffer.

4.2.2 Effective Bandwidth of Markovian Traffic Sources

ON/OFF Sources

An expression for the effective bandwidth of an ON/OFF source was given in [GUER91]. Each ON/OFF source is characterized by three parameters: peak cell rate R_{peak}, utilization (the fraction of time that the source is active) ρ, and mean duration of the active period b. Consider that such an ON/OFF source is feeding a finite capacity queue with a constant service rate c ($c < R_{peak}$). Let K be the buffer size. The buffer overflow probability is found to be of the form:

$$\varepsilon = \beta \times \exp\left(-\frac{K(c - \rho R_{peak})}{b(1-\rho)(R_{peak} - c)c} \right),$$

where $\beta = \dfrac{(c - \rho R_{peak}) + \varepsilon\rho(R_{peak} - c)}{(1-\rho)c}$.

The service rate, or the effective bandwidth of an ON/OFF source, needed to achieve a given buffer overflow probability or CLR QoS requirement ε, can be found by inverting the above equation. However, no closed form solution can be obtained. A natural simplification is to approximate β by 1 (β is always less than 1. Since $\varepsilon < 1, \beta < \dfrac{(c - \rho R_{peak}) + \rho(R_{peak} - c)}{(1-\rho)c} = 1$). An upper bound of the effective bandwidth of an ON/OFF source is then given as:

$$\hat{c} = \frac{\alpha b(1-\rho)R_{peak} - K + \sqrt{[\alpha b(1-\rho)R_{peak} - K]^2 + 4K\alpha b\rho(1-p)R_{peak}}}{2\alpha b(1-\rho)}, \qquad (1)$$

where $\alpha = \ln(1/\varepsilon)$.

The above expression is obtained from the assumption that the ON/OFF periods of an ON/OFF source are exponentially distributed. For ON/OFF sources with generally distributed ON or OFF periods, a two moment matching technique is used to map the first and second moments of a generally distributed ON/OFF period into the first moment of an equivalent, exponentially distributed ON/OFF period.

The aggregate effective bandwidth of a number of multiplexed ON/OFF sources is computed from the combination of two different approximations. The first approximation works when the number of sources is small or when the effect of statistical multiplexing is small. In this case, the aggregate effective bandwidth of *N* multiplexed ON/OFF sources is given as :

$$\hat{C}_F = \sum_{i=1}^{N} \hat{c}_i ,$$

where $\hat{c}_i$ is the effective bandwidth of source i, $1 \le i \le N$.

The second approximation is more accurate when the effect of statistical multiplexing is of significance. In this case, the aggregate effective bandwidth of N multiplexed ON/OFF sources is given as:

$$\hat{C}_S = m + \alpha' \sigma,$$

where m is the aggregate mean cell rate ($m = \sum_{i=1}^{N} m_i$), and σ is the standard deviation of the aggregate cell rate ($\sigma^2 = \sum_{i=1}^{N} \sigma_i^2 = \sum_{i=1}^{N} m_i (R_{peak,i} - m_i)$), and $\alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}$.

As both situations are typically exclusive, the two approximations complement each other and can be combined to produce relatively accurate estimation of the aggregate effective bandwidth of a number of multiplexed ON/OFF sources. The aggregate effective bandwidth $\hat{C}$ is taken to be the minimum of $\hat{C}_F$ and $\hat{C}_S$. That is:

$$\hat{C} = \min(\hat{C}_F, \hat{C}_S) \tag{2}$$

MMFP and MMPP Sources

Expressions of the effective bandwidth of MMFP and MMPP sources were derived in [ELWA93] based on the theory of large deviations. Each MMFP or MMPP source is characterized by two matrices, the infinitesimal generator of a controlling Markov chain Q and the cell generation rate (or density) matrix Λ. The effective bandwidth of an MMFP source is found to be the maximum real eigenvalue of the matrix $\Lambda - \frac{1}{\zeta} Q$, where $\zeta = \ln(\varepsilon) / K$. A closed form for the effective bandwidth of a two-state MMFP source is given as:

$$\frac{1}{2\zeta}\left([(\lambda_1+\lambda_2)\zeta-Q_{1,1}-Q_{2,2}]-\sqrt{[(\lambda_1+\lambda_2)\zeta-Q_{1,1}-Q_{2,2}]^2-4(\lambda_1\lambda_2\zeta^2-Q_{2,2}\lambda_1\zeta-Q_{1,1}\lambda_2\zeta)}\right). \quad (3)$$

The effective bandwidth of an MMPP source is found to be the maximum real eigenvalue of the matrix $\frac{1}{e^{\zeta}}\Lambda+\frac{1}{1-e^{\zeta}}Q$. A closed form for the effective bandwidth of a two-state MMPP source is given as:

$$\frac{1}{2}\left(\left[\frac{\lambda_1+\lambda_2}{e^{\zeta}}+\frac{Q_{1,1}+Q_{2,2}}{1-e^{\zeta}}\right]+\sqrt{\left[\frac{\lambda_1+\lambda_2}{e^{\zeta}}+\frac{Q_{1,1}+Q_{2,2}}{1-e^{\zeta}}\right]^2-4\left(\frac{\lambda_1\lambda_2}{e^{2\zeta}}+\frac{Q_{1,1}\lambda_2+Q_{2,2}\lambda_1}{e^{\zeta}(1-e^{\zeta})}\right)}\right). \quad (4)$$

4.2.3 Advantages and Disadvantages

The major advantage of a CAC scheme using effective bandwidth reservation is its computational simplicity resulting from the additive nature of effective bandwidth. This simplicity enables it to operate in real time. It is also quite robust. In most cases, it outperforms other CAC schemes found in the literature.

The major disadvantage of a CAC scheme using effective bandwidth is its poor flexibility. Such a CAC scheme relies on users to specify the traffic model for each call. Different traffic models require different traffic parameters and have different formulae for calculating the effective bandwidth. Furthermore, the additive feature of effective bandwidth corresponds to ignoring the effect of statistical multiplexing and thus, results in low efficiency when statistical multiplexing is significant. In fact, as pointed out in [CHOU96], the aggregate effective bandwidth is overestimated when there are many sources that are more bursty than a Poisson source. In some cases, the performance of this CAC scheme can be worse than that of the CAC

scheme using peak bandwidth reservation. The aggregate effective bandwidth can also be underestimated when there are many sources that are less bursty than a Poisson source.

4.3 BUFFERLESS FLUID FLOW MODEL AND VIRTUAL CELL LOSS PROBABILITY

Decomposition of congestion was first introduced in [HUI88]. A bufferless fluid flow model was proposed in [JACO90]. Virtual cell loss probability was given under the buffer fluid flow model in [MURA91] to approximate the actual CLR. A fast implementation of the CAC scheme based on the bufferless fluid flow model and the virtual cell loss probability can be found in [LEE96], where each traffic source is characterized by two traffic parameters, PCR and SCR.

4.3.1 Decomposition of Congestion and the Bufferless Fluid Flow Model

Congestion can be decomposed into two levels: cell-level and burst-level [HUI][KEY95]. Cell-level congestion is caused by simultaneous cell arrivals from different traffic sources at the same output buffer of an output link. Burst-level congestion is caused by the variations of aggregate cell generation rate of all active traffic sources. Moderate sized cell-scale output buffers are effective in coping with the cell-level congestion. As the cell-scale output buffer size increases linearly, the buffer overflow probability associated with the cell-level congestion decreases exponentially. However, the buffer overflow probability associated with the burst-level congestion decreases much more slowly when the burst-scale buffer size is increased linearly. A great amount of extra buffering is then needed to cope with the burst-level congestion, which may result in large cell delay. Therefore, only cell-level output buffers are used in ATM switches, and CAC schemes are needed to prevent the burst-level congestion. A bufferless fluid

flow model was proposed in [JACO90] to study the burst-level congestion. Under the bufferless fluid flow model, cell loss due to buffer overflow occurs if and only if the sum of the cell arrival rates of all active connections exceeds the output link capacity C.

4.3.2 Virtual Cell Loss Probability

Virtual cell loss probability was introduced in [MURA91] under the bufferless fluid flow model to approximate the actual CLR. Each traffic source is assumed to be an ON/OFF source, and is characterized by two parameters: PCR and SCR. Suppose that there are N existing connections. Let MAX_i and AVG_i denote the PCR and the SCR of connection i, $1 \le i \le N$. $f_i(x)$ represents the probability density function of the traffic generated by connection i, i.e.,

$$f_i(x) = \begin{cases} \dfrac{AVG_i}{MAX_i}, & x = MAX_i \\ 1 - \dfrac{AVG_i}{MAX_i}, & x = 0. \end{cases}$$

All connections are assumed to be independent of each other. The density function of the aggregate traffic generated by the N existing connections, denoted by $q(x)$, is equal to the convolution of $f_1, f_2, \cdots,$ and f_N, i.e., $q(x) = (f_1 * f_2 * \cdots * f_N)(x)$.

Let ET denote the expected excess traffic and φ represent the traffic load. The virtual cell loss probability Pv under the bufferless fluid flow model is defined as:

$$Pv = \frac{ET}{\varphi},$$

where $ET = \sum_x (x - C)^+ q(x)$ and $\varphi = \sum_{i=1}^{N} AVG_i$.

4.3.3 A Fast Implementation

A fast implementation of the CAC scheme based on the bufferless fluid flow model and the virtual cell loss probability can be found in [LEE96]. A basic data rate μ is selected so that the output link capacity *C*, is an integral multiple of μ. Assume that the PCR of each connection is also an integral multiple of μ. The density function, $f_i(x)$, of a traffic source, *i*, whose PCR equals *j* times the basic data rate μ can be expressed as:

$$f_i(k) = \begin{cases} \dfrac{AVG_i}{MAX_i}, & k = j \\ 1 - \dfrac{AVG_i}{MAX_i}, & k = 0. \end{cases}$$

Define *I(m)* as follows:

$$I(m) = \sum_{k=m+1}^{\infty} (k-m)q(k).$$

I(m) represents the expected excess traffic when the output link capacity is *m* times the basic data rate μ under the bufferless fluid flow model, and *I(0)* is equal to the traffic load, φ. Moreover, the virtual cell loss probability is given by *I(C)/I(0)*.

Suppose there are *N* existing connections and a call setup request with PCR MAX_{N+1} and SCR AVG_{N+1} arrives. Let $\hat{I}(m)$ denote the updated value of *I(m)* assuming that the call setup request is accepted. $\hat{I}(m)$ can be obtained as follows:

$$\hat{I}(m) = \sum_k (k-m)^+ q * f_{N+1}(k)$$

$$= (1 - \frac{AVG_{N+1}}{MAX_{N+1}}) \sum_k (k-m)^+ q(k) + \frac{AVG_{N+1}}{MAX_{N+1}} \sum_k (k-m)^+ q(k - \frac{MAX_{N+1}}{\mu})$$

$$= \begin{cases} (1-\frac{AVG_{N+1}}{MAX_{N+1}})I(m)+\frac{AVG_{N+1}}{MAX_{N+1}}[(\frac{MAX_{N+1}}{\mu}-m)+I(0)], & m<\frac{MAX_{N+1}}{\mu} \\ (1-\frac{AVG_{N+1}}{MAX_{N+1}})I(m)+\frac{AVG_{N+1}}{MAX_{N+1}}I(m-\frac{MAX_{N+1}}{\mu}), & m\geq\frac{MAX_{N+1}}{\mu} \end{cases}.$$

Since $I(m)=I(0)-m$ for $m<0$, $\hat{I}(m)$ can be rewritten as:

$$\hat{I}(m)=f_{N+1}(0)I(m)+f_{N+1}(\frac{MAX_{N+1}}{\mu})I(m-\frac{MAX_{N+1}}{\mu}),$$

where $\hat{I}(0)=I(0)+AVG_{N+1}$.

The initial value of *I(m)* when $N=0$ is zero for all *m*. The call setup request is accepted and the values of $I(m)$, $0\leq m\leq C$, are updated if $\hat{I}(C)/\hat{I}(0)\leq\varepsilon$, where ε is a predetermined CLR QoS requirement. The call setup request is denied and the values of $I(m)$, $0\leq m\leq C$, are not updated if $\hat{I}(C)/\hat{I}(0)>\varepsilon$. When a connection i with parameters MAX_i and AVG_i finishes, *I(m)*, $0\leq m\leq C$, can be updated recursively as follows:

$$\hat{I}(m)=\frac{1}{f_i(0)}\left[I(m)-f_i(MAX_i)\hat{I}(m-MAX_i)\right],$$

where $\hat{I}(0)=I(0)-AVG_i$.

4.3.4 Advantages and Disadvantages

The major advantage of CAC based on the bufferless fluid flow model and virtual cell loss probability is its computational simplicity. This CAC scheme is able to operate in real-time as long as the values of *I(m)*, $0\leq m\leq C$, are stored, which requires only a constant memory size. To make an accept/reject decision, the CAC scheme only needs to perform two multiplications to get $\hat{I}(C)$, and one division to get $\hat{I}(C)/\hat{I}(0)$. To update the *I(m)*, $0\leq m\leq C$, the CAC scheme

needs to perform a total of $2(C+1)$ multiplications. The CAC scheme is flexible since only two traffic parameters, PCR and SCR, are required for each call. The CAC scheme is also quite robust.

The major disadvantage of this CAC scheme is its possible low efficiency due to overestimating the actual CLR. Overestimating the CLR results from the assumptions that no buffer is used and that each traffic source is an ON/OFF source. It is necessary to point out that the ON/OFF source is not the "worst-case" traffic that complies with the traffic descriptor in some situations [DOSH93].

4.4 NEURAL NETWORK TRAINED BY THE VIRTUAL OUTPUT BUFFER METHOD

The use of neural networks for CAC in ATM networks was first proposed in [HIRA90]. Several neural network architectures for CAC were given in [TRAN92]. Two training techniques, called relative target method and virtual output buffer method, for neural networks with applications in CAC were introduced in [HIRA94] and [HIRA95]. A CAC scheme based on a neural network was studied in [YOUS97], where the neural network was used to calculate the effective bandwidth of calls of different classes. A good textbook on the subject of neural network is [FREE92].

4.4.1 Neural Network

Neural networks have recently received great attention from many researchers for their potential applications in ATM traffic control. A neural network approach for CAC has several advantages. One of the them is the learning and adaptive capabilities of a neural network which enable it to

support new services and to maximize statistical multiplexing gain transparently. Second, in contrast to the conventional mathematical approaches, neither explicit traffic models nor an accurate modeling of the system are needed. Third, the parallel structure of neural networks can be exploited in hardware implementations to achieve short response times.

Different types of neural networks can be found in the literature. The most commonly used type is called the multi-layer perception neural network. It is generally a multiple-input, multiple-output, non-linear mapping mechanism. It can learn an unknown non-linear input-output relationship from a set of examples. It consists of many neurons connected to each other. Each neuron is a multiple-input, single-output, non-linear circuit. The connection strengths between neurons are called weights. The non-linear, input-output relationship can be approximated by changing the set of weight values in the multi-layer perception neural network. The most widely used algorithm to adjust weight vectors according to the difference between the actual output and the target output is called back-propagation. A typical structure of a multi-layer perception neural network trained by the back-propagation algorithm is depicted in Figure 4-1.

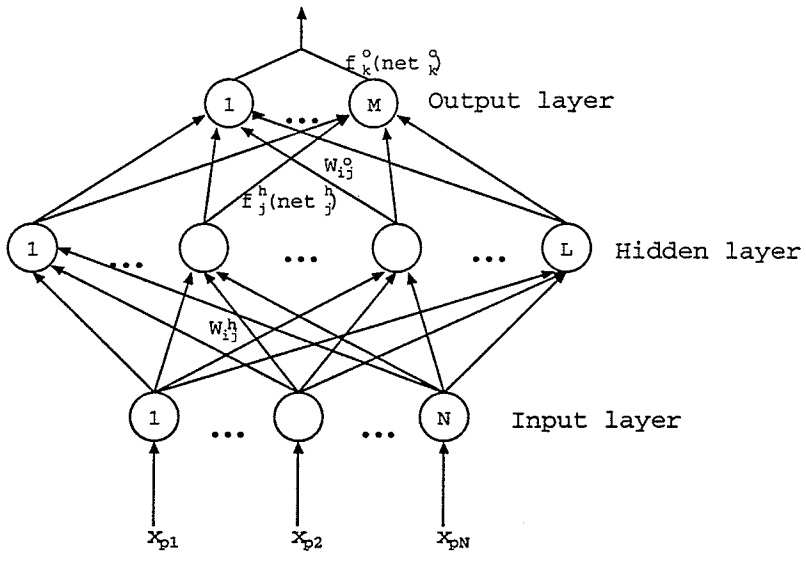

Figure 4-1. A typical structure of a multi-layer perception neural network trained by the back-propagation algorithm

4.4.2 Virtual Output Buffer Method

Call classification is used in CAC schemes based on neural networks. Each connection is categorized into one of k call classes according to the traffic parameter values declared in its call setup request. All connections in the same class are assumed to have almost the same cell generation characteristics. The status of an ATM switch, $\vec{n}$, managed by a neural network based CAC scheme is defined as:

$$\vec{n} = (n_1, n_2, \cdots, n_k),$$

where n_i denotes the number of connections of class i, $i = 1,2,\cdots k$.

One major difficulty faced by neural network based CAC schemes is how to sample the output buffer CLR at each possible switch status when the switch status is always changing. CLR QoS requirements are usually extremely small. When CLR QoS is well maintained, it is necessary to take very long times (maybe hours) to have a reasonably accurate CLR observation, which is not even practical during simulations. To overcome this problem, the virtual output buffer method was proposed in [HIRA94][HIRA95]. This method estimates the very small CLR at the physical output buffer from the CLR observations from a set of virtual output buffers. The basic idea of the virtual output buffer method is that parallel to the physical output link associated with an output buffer, there are a set of imaginary output links with their respective virtual buffers. The cell arrival process and the size of those virtual buffers are the same as those of the physical output buffer. However, the capacity of those imaginary output links are much less than that of the physical output link. Therefore, the CLR at each virtual buffer can be much higher than that of the physical output buffer. This enables the CLRs at those virtual output buffers to be observed in very a short period of time. The accuracy of the observed CLRs is also

improved. Each virtual output buffer can be implemented as a set of counters that count the buffer size, as well as the number of arriving, waiting and lost cells.

The observed CLRs at those virtual output buffers are stored in a pattern table with the imaginary output link capacities and the switch status at that time. By extrapolating the CLRs from those virtual buffers at certain status $\vec{n}$, it is easy to derive the minimum bandwidth v_0 needed to achieve a predefined CLR QoS requirement ε. See Figure 4-2.

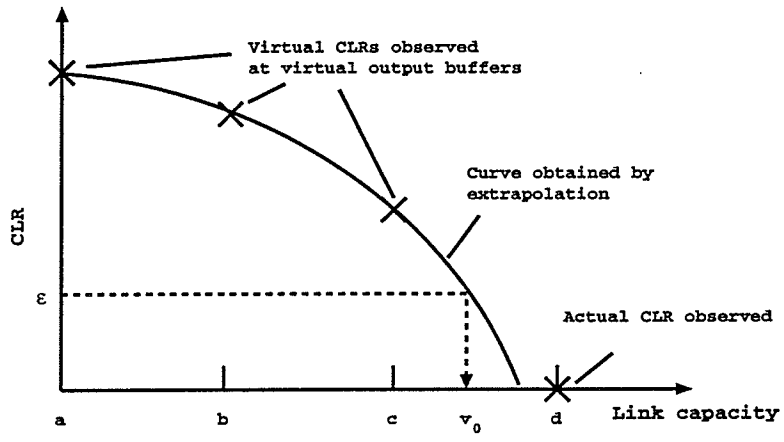

Figure 4-2. Minimum bandwidth needed to achieve the CLR QoS requirement ε at switch status $\vec{n}$

A multi-layer perception neural network is trained by the back propagation algorithm using the data from the pattern table to learn a non-linear capacity/CLR relationship for each switch status. The surface created by the neural network is called the CLR estimation surface. For inexperienced switch status, some other extrapolation algorithms are required to extrapolate the surface.

A CAC scheme based on a neural network trained by the virtual output buffer method is given in [HIRA94] and [HIRA95]. It works as follows. Suppose that the switch status before a new call setup request of class j arrives is $\vec{n} = (n_1, n_2, \cdots, n_k)$. The minimum bandwidth, v_0,

needed to achieve the CLR QoS requirement ε, at the new switch status, $\vec{n}^{+} = (n_1, n_2, \cdots, n_{j+1}, \cdots, n_k)$, is estimated by the neural network. The result is then compared to the output link capacity C. The call setup request is accepted if $v_0(\vec{n}^{+}) \leq C$, or rejected if $v_0(\vec{n}^{+}) > C$. The operation of the CAC scheme based on a neural network trained by the virtual output buffer method is illustrated in Figure 4-3.

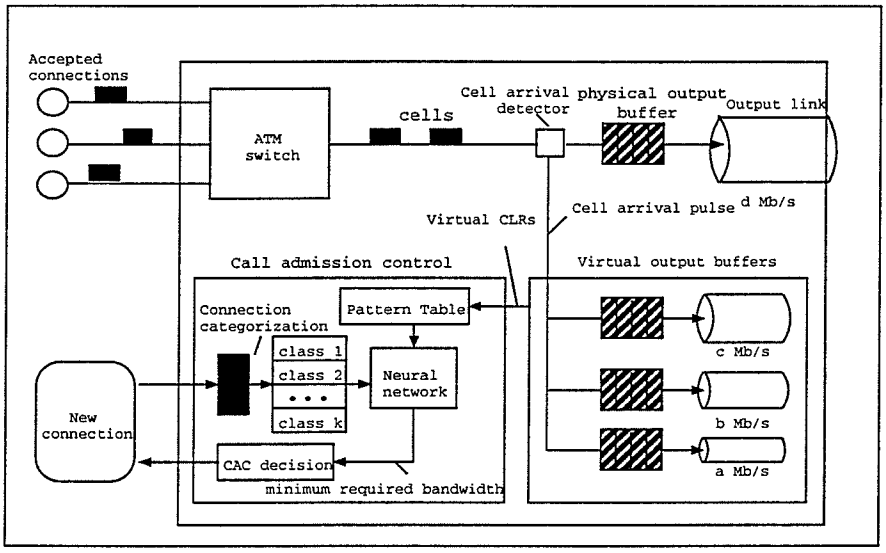

Figure 4-3. A CAC scheme based on a neural network trained by the virtual output buffer method

4.4.3 Advantages and Disadvantages

The major advantage of the CAC scheme based on a neural network trained by the virtual output buffer method is its flexibility. New services can be supported by adding additional classes and training the neural network for new switch status. Once the neural network is properly trained, CAC decisions can be made in real-time. The efficiency of this CAC scheme is expected to be high since no worst-case cell arrival process is assumed and the effect of statistical multiplexing can be fully exploited by the neural network. Robustness of this CAC scheme can be achieved by

training the neural network to estimate the upper bound of the minimum bandwidth needed to achieve the predefined CLR QoS requirement.

The major disadvantage of this CAC scheme is that it may require huge memory storage if the switch status space is large. The implementation of this CAC scheme is also quite difficult since the number of layers in the neural network, and the number of neurons in each layer have to be selected by intensive simulations and comparisons. Various traffic models are also needed to train the neural network off-line before any on-line operations.

4.5 OTHER CAC SCHEMES

A decision-theoretic approach for CAC was proposed in [KEY95] and [GIBB95]. Each traffic source is characterized by its PCR and CDVT. Acceptance decisions are based on whether the current traffic load is less than a pre-calculated threshold, and Bayesian decision theory provides the framework for the choice of the threshold.

An upper bound of the cell loss probability was given in [SAIT91], [SAIT92] and [SAIT94], where a dynamic CAC scheme based on comparing the upper bound of the CLR after the acceptance of a new call to the CLR QoS requirement was proposed. The traffic parameters specified by users are the maximum and average number of cells arriving during a fixed time interval. The fixed interval was set to be a half of the maximum delay in a buffer. For other upper bounds on the cell loss probability, see [RASM91].

Relationship between the CLR and the output buffer size can be found in [ZHU96]. For systems with general Markovian traffic sources, queuing analyses indicate that there exists a linear relationship between the logarithm of the cell loss probability and the buffer size. That is:

$\ln(CLR) \sim -\alpha - \beta B$, where B is buffer size, α and β are both positive constants. For systems with self-similar traffic sources, extensive simulation suggests that this linear relationship no longer exists. Instead, buffer size and $\ln(CLR)$ results in the following relationship: $\ln(CLR) \sim -\alpha B^{\beta}$, where α and β are both constants. A CLR QoS violation detection scheme based on the above relationship was proposed in [ZHU96], where linear regression is used to extrapolate the CLR at the physical buffer from the CLRs observed at a set of pseudo-buffers with much smaller buffer sizes.

Several queuing models have been studied in order to find the cell loss probability. In [BAIO91], the aggregate cell arrival process is approximated by a two state MMPP and the system is treated as a MMPP/D/1/K queue. In [YANG95], a MMFP is used to approximate the aggregate cell arrival process, and the system is studied as a MMFP/D/1/K queue.

CAC based on fuzzy inference was studied in [UEHA97], where CAC decisions are made by comparing the upper bound of the CLR after accepting a new connection to the CLR QoS requirement. The upper bound of the CLR is obtained from a CLR distribution estimated by a fuzzy inference scheme.

A review on CAC schemes was conducted in [PERR96].

CHAPTER FIVE

A NEW CAC SCHEME

In this chapter, two different approaches to design CAC schemes are discussed. The first approach is based on estimating the CLR after the acceptance of a new call. The second approach is based on estimating the available bandwidth of an output link. A new CAC scheme using the second approach is developed. The new CAC scheme and two other CAC schemes are implemented as process models in OPNET. This chapter is organized as follows. Section 5.1 discusses the problems faced by CAC schemes using the first approach. Section 5.2 explains why designing CAC schemes using the second approach may be a better approach. Section 5.3 presents the new CAC scheme based on the second approach. Both the advantages and the possible weaknesses of this new CAC scheme are addressed. Section 5.4 gives the details of the implementation of this new CAC scheme. The implementations of two other CAC schemes, peak bandwidth reservation and

effective bandwidth reservation, are also provided in this section. Section 5.6 summarizes this chapter.

5.1 PROBLEMS FACED BY CAC SCHEMES BASED ON ESTIMATING THE CLR AFTER THE ACCEPTANCE OF A NEW CALL

Except for CAC schemes using peak bandwidth reservation and effective bandwidth reservation, almost all other CAC schemes found in the literature are based on the approach of estimating the CLR after the acceptance of a new call. A call setup request is accepted if and only if the estimated CLR is less than a predefined CLR QoS requirement. A lot of effort has been put into designing CAC schemes using this approach. However, the performance of those CAC schemes is far from satisfying. There are several major problems faced by CAC schemes using this approach. They are summarized as follows:

1. CLR is rather a statistical property than a deterministic property. At any time point, CLR is a variable with certain distribution and not a deterministic number. A one-time CLR observation less than the CLR QoS requirement does not necessarily mean that congestion is not on its way. Buffer occupancy could have been statistically increasing and a wrong decision can cause severe cell losses. Experience shows that cell losses either do not occur or occur in bursts. Therefore, CLR observations are either much less than the CLR QoS requirement or much higher than the CLR QoS requirement. In most cases, CLR does not deteriorate gradually. All this suggests that designing CAC schemes based on this approach may not be such a good idea.

2. No estimation can be perfect. One can imagine the difficulty of estimating such an extremely small floating number like the CLR. A small absolute error can result in a very large relative error. One way to estimate the CLR is to derive the buffer overflow probability from some queuing models. Those models often have assumptions on the traffic arrival processes, which are not necessarily true. Another way to estimate the CLR is to introduce a set of pseudo buffers, either have much smaller buffer sizes, or much smaller service rates than those of the physical buffer. The CLRs associated with those pseudo buffers are usually much higher than that of the physical buffer and they can be observed in less time. The CLR associated with the physical buffer is then estimated by fitting and extrapolating a curve though the CLR observations from those pseudo buffers. There is no doubt that estimating the CLR from both ways could introduce some kind of error. Either an underestimation or an overestimation of the CLR can cause severe consequences. An underestimation of CLR may cause a CAC scheme to accept more calls than the output link can handle and thus results in severe cell losses and degraded QoS. This violates the role of a CAC scheme and almost all CAC schemes based on queuing models try to avoid it by considering the worst-case cell arrival process which result in an inevitable overestimation of the CLR. An overestimation of the CLR could cause a low output link utilization. In some cases, the output link utilization achieved by CAC schemes using this approach could be even lower than that achieved by the CAC scheme using peak bandwidth reservation.
3. Estimating the CLR usually takes a relatively large amount of time, resulting from either intensive computation associated with those very complex queuing models, or from the necessary observation time associated with those pseudo buffers. All CAC

schemes have to be capable of operating in real-time. However, most CAC schemes using this approach fail to meet this requirement.

5.2 A DIFFERENT APPROACH

CAC schemes can be designed using a different approach which avoids estimating the CLR completely. In this approach, CLR QoS requirement is satisfied by not allowing any cell loss at any time. The maximum extra traffic load, which could be added to the existing traffic load without causing any cell loss, is estimated and referred as the available bandwidth of an output link. A new call setup request is accepted if and only if its peak bandwidth is less than the available bandwidth. It is easy to jump to the conclusion that this approach could decrease the output link utilization significantly since it tries to offer a perfect, no cell loss service. However, the impact on the output link utilization by not allowing any cell loss is actually negligible. Intuitively, the difference between the traffic load that causes one cell loss in one billion cells and the maximum traffic load that causes no cell loss in one billion cells is just a cell. Peak bandwidth reservation and effective bandwidth reservation are the two most popular CAC schemes. Both of them are not based on estimating the CLR after the acceptance of a new call. CAC schemes based on estimating the available bandwidth of an output link have at least the following advantages over those based on estimating the CLR after the acceptance of a new call.

1. The available bandwidth of an output link is rather a deterministic property than a statistical property. At any time point, the available bandwidth is a deterministic

number and not a variable with a certain distribution. Also, available bandwidth is not a small number. A small absolute error will not result in a very large relative error. Estimation accuracy can be dramatically improved. Therefore, estimating the available bandwidth is much easier than estimating the CLR.

2. CAC decisions can be made easily. Once an estimation of the available bandwidth is made, a new call request can be accepted or rejected just by comparing its peak bandwidth to the available bandwidth. This simplicity enables CAC schemes based on estimating the available bandwidth to meet the real-time requirement.
3. No assumptions on the types of each traffic source are needed. This avoids assuming a worst-case traffic arrival process and therefore improves the output link utilization. Furthermore, CLR QoS requirement is always maintained.

However, research on CAC schemes based on estimating the available bandwidth is almost blank in the literature and very few algorithms on how to estimate the available bandwidth exist.

5.3 A NEW CAC SCHEME BASED ON ESTIMATING THE AVAILABLE BANDWIDTH

5.3.1 The New CAC Scheme

A new CAC scheme based on estimating the available bandwidth of an output link is developed and it works as follows. When a call setup request arrives, the call is accepted if its peak bandwidth is less than the available bandwidth, otherwise, it is rejected. If the new call request is accepted, its peak bandwidth is subtracted from the available

bandwidth. When a connection is released, its average bandwidth is added to the available bandwidth. The actual bandwidth assigned to each accepted call is between the call's peak bandwidth and the call's average bandwidth. An algorithm updates the available bandwidth every t seconds. This fixed period of time is referred to an available bandwidth update time interval. The algorithm is illustrated in Figure 5-1.

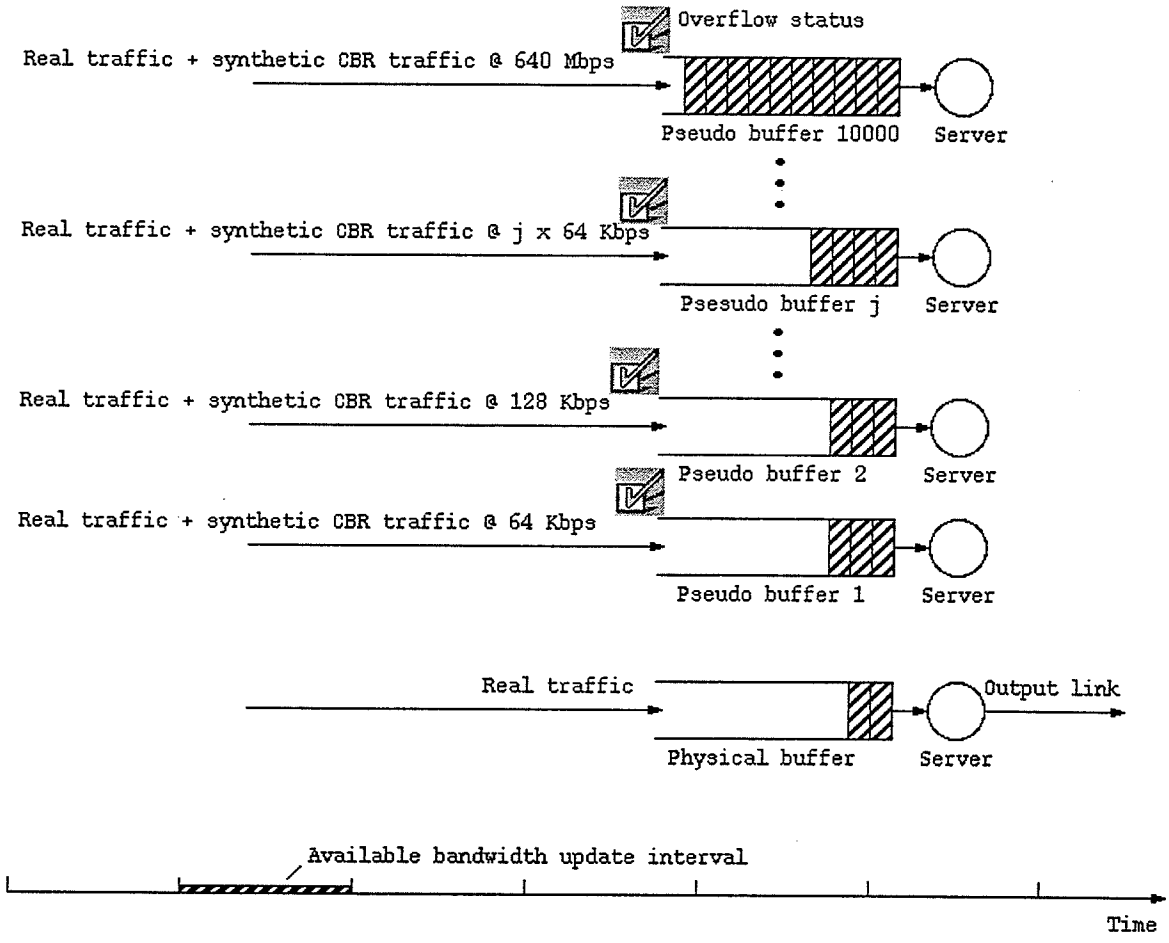

Figure 5-1. The available bandwidth estimation algorithm

The algorithm estimates the available bandwidth of an output link as follows. In addition to the physical buffer, there are a number of pseudo buffers. The size and the service rate of each pseudo buffer are the same as those of the physical buffer. Traffic to

the pseudo buffers i is composed of real traffic from existing connections plus a synthetic CBR traffic at the rate of $i\mu$ Kbps, where μ is a pre-defined basic data rate. The number of pseudo buffers, K, is set to be $\lfloor C/\mu \rfloor$, where C is the service rate of the physical buffer or equivalently, the capacity of the output link in Kbps. $\lfloor x \rfloor$ refers to the largest integer smaller than x. Each pseudo buffer is associated with a Boolean variable called overflow status. At the beginning of an available bandwidth update time interval, the overflow status of each pseudo buffer is set to be not overflow. During an available bandwidth update time interval, if cell loss occurs at a pseudo buffer, the overflow status of that pseudo buffer is set to be overflow. At the end of an available bandwidth update time interval, the algorithm checks the overflow status of each pseudo buffer and finds out the pseudo buffer which is not overflow and has the largest traffic load. If the index of that pseudo buffer is j, then the available bandwidth is set to $j\mu$ Kbps. An exception is that if all pseudo buffers are overflow, the available bandwidth remains unchanged instead of being set to 0.

5.3.2 Advantages of Disadvantages of the New CAC Scheme

The new CAC scheme is designed with four objectives: robustness, simplicity, efficiency, and flexibility, in mind. The new CAC scheme is robust as long as the length of each available bandwidth update time interval is set properly. The robustness of the new CAC scheme is achieved from two aspects. First, each accepted call is assigned its peak bandwidth at the call setup stage and when it is released, only its average bandwidth is added to the available bandwidth. Second, the available bandwidth is estimated from not

allowing any cell loss in the pseudo buffers during an available bandwidth update time interval. The new CAC scheme is able to operate in real-time. The computation cost at the call setup stage is kept at minimum by just comparing the PCR of a new call to the estimated available bandwidth. The computation cost associated with updating the available bandwidth spreads over an entire available bandwidth update time interval. The new CAC scheme is expected to be effective since the effect of statistical multiplexing is fully exploited by the available bandwidth estimation algorithm. The new CAC scheme can support calls of new traffic types without any change since only a mandatory traffic parameter, PCR and an optional traffic parameter SCR, are required for each call. The PCR is used to calculate the peak bandwidth of the call and the SCR is used to calculate the average bandwidth of the call. If PCR is the only traffic parameter specified by users, zero instead of its average bandwidth is added to the available bandwidth when a connection finishes.

One weakness of this CAC scheme is that its robustness depends on setting the length of each available bandwidth update time interval to an appropriate value. If the available bandwidth update time interval is set to be too short, the burstiness of the aggregate traffic may not be fully captured during an update time interval and the available bandwidth can be overestimated. An overestimation of the available bandwidth can cause the CAC scheme accept more calls that the output link can handle and results in a relatively large amount of cells to be discarded and a degraded CLR QoS. If the available bandwidth update time interval is set to be too large, no extra information on the burstiness of the traffic can be learned during the prolonged update time interval and the utilization of the output link decreases since the available bandwidth is updated less

frequently. The effect of the length of each available bandwidth update time interval on the performance of the new CAC scheme is also studied in Chapter Six.

5.4 IMPLEMENTATION OF CAC SCHEMES IN OPNET

5.4.1 IMPLEMENTATION OF THE NEW CAC SCHEME

The new CAC scheme is implemented as a process model in OPNET. In the implementation, only one synthetic CBR traffic source instead of a set of synthetic CBR traffic sources is used. Traffic from this traffic source is generated at the rate of $K\mu$ Kbps, where K is the number of pseudo buffers and μ is a predefined basic data rate. When a synthetic cell arrives at the process model, the queue length $q(i)$ at pseudo buffer i is updated as:

$$q(i) \leftarrow q(i) + \frac{i}{K}, \quad 1 \le i \le K .$$

The state transition diagram of the process model is shown in Figure 5-2. The process model has 6 model attributes which are listed in Figure 5-3. The capacity of the output link can be specified by the "output link capacity" model attribute. The size of the output buffer can be specified by the "buffer size" model attribute. The basic data rate μ can be specified by the "basic data rate" model attribute. This process model generates a data file and a log file during a simulation. Every time a call is accepted or released, an entry of the number of connections along with the corresponding simulation time is appended to the data file. Every time a call is accepted or released, or at the end of each available bandwidth update time interval, the number of connections, the available

bandwidth, the number of cells received, and the number of cells discarded, are written into the log file. The name of the data file can be specified by the "data file name" model attribute. The name of the log file can be specified by the "log file name" model attribute. This process also provides a vector statistic which shows the number of connections along with the progress of the simulation. The vector statistic can be viewed in the Analysis Tool in OPNET when the simulation is completed. There are eleven states in this process model: nine forced (init, arrival, svc_start, svc_comp, pesudo, cac, release, update, and file_fresh), and two unforced (idle and sim_end). They are described in Table 5-1. The process model has 10 transition conditions which are elaborated in Table 5-2. Three procedures are used in the enter and transition executives of the process model. Their descriptions are given in Table 5-3.

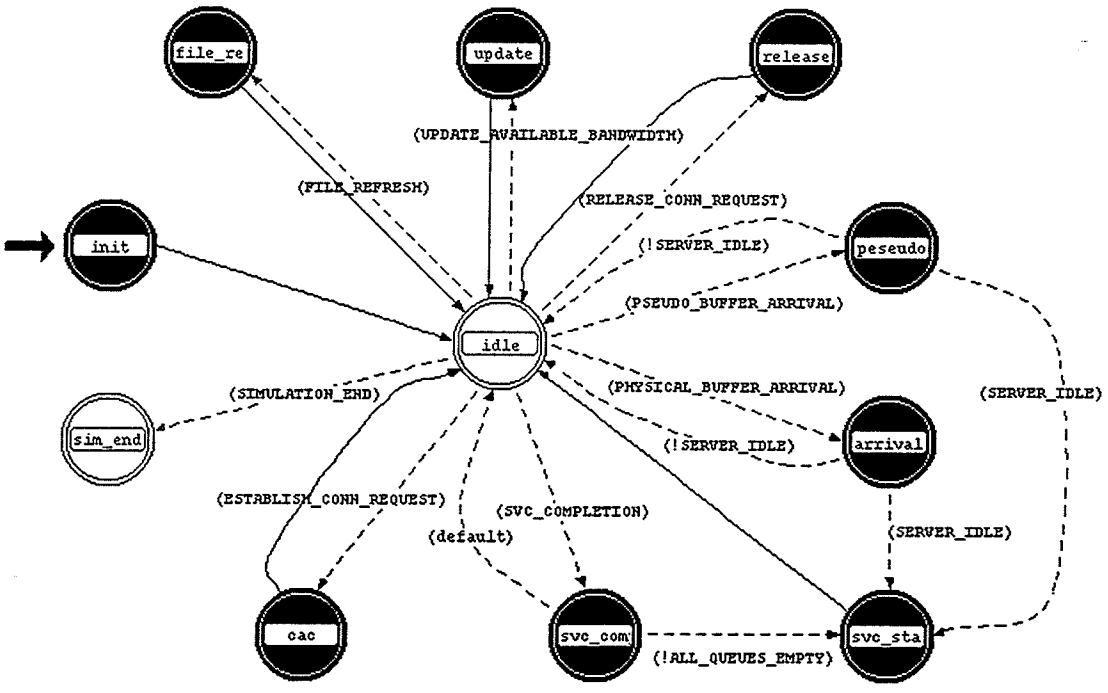

Figure 5-2. State transitions diagram of the process model of the new CAC scheme

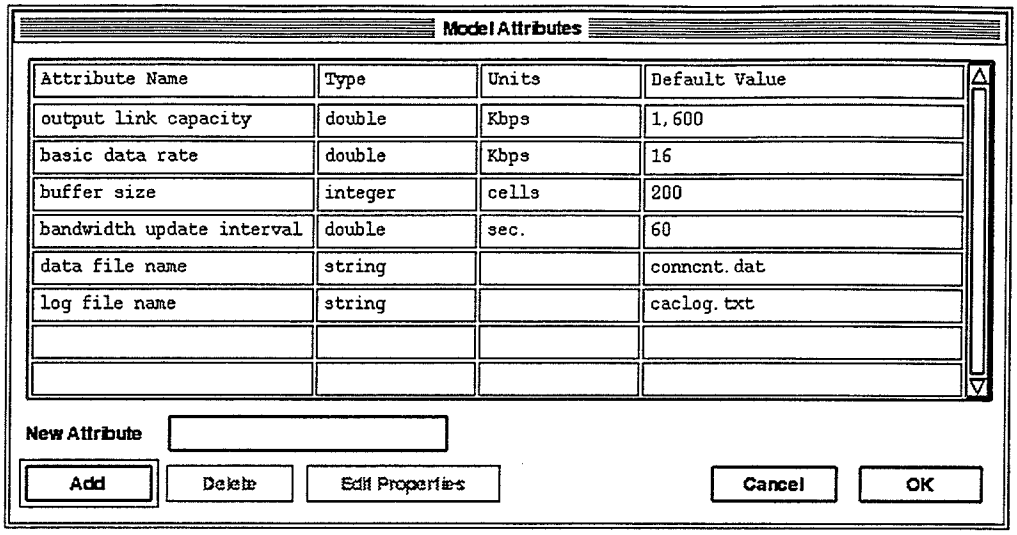

Figure 5-3. Model attributes of the process model of the new CAC scheme

State Descriptions

State	Description
init	The process starts at this state where user-supplied model attribute values are read and saved in corresponding state variables. A vector statistics, which is the number of connections along with the progress of the simulation, is declared. Several counters, such as the number of connections, the number of received call setup requests, the number of received cells, and the number of discarded cells are initialized. A connection list, which is used to save the identification and the average bandwidth of each connected call, is created. A data file and a log file is also created and opened. A self interrupt is scheduled in a fixed period of time to mark the arrival of the first synthetic cell at the process. The length of this fixed period of time is set as the cell length in Kbps divided by the data rate of the synthetic CBR traffic. A second self interrupt is scheduled in a fixed period of time in order to let the process go to the update state and update the available bandwidth for the first time. The length of this fixed period of time can be specified by the "available bandwidth update interval" model attribute. A third self interrupt is scheduled in a fixed period of time to let the process to go to the file_refresh state for the first time where the data and log files are closed and reopened to prevent them becoming stale and unreadable from being opened too long. The length of this fixed period of time is set to be 60 seconds by default.
idle	This state is entered right after the process completes all the actions at one of the forced states. The process rests in this state

	and waits to be invoked again.
arrival	This state is entered when a cell from one of the connected calls arrives. The number of cells received by this process is incremented by one. If the physical buffer is full, the number of cells discarded by this process is incremented by one. Otherwise, the queue length of the physical buffer is incremented by one. For each pseudo buffer, its queue length is incremented by one if it is not full. Otherwise, the overflow status of that pseudo buffer is set to be overflow.
svc_start	This state is entered right after the process leaves the arrival state or the pseudo_arrival state if the server is idle, or right after the process leaves the svc_comp state if the physical buffer or at least one of the pseudo buffers is not empty. A cell transmission starts and the status of the server is set to be busy. A self interrupt is scheduled in a fixed period of time to mark the end of this cell transmission. The length of this fixed period time is set as the cell length in Kb divided by the capacity of the output link.
svc_comp	This state is entered when a cell transmission is completed. The queue length of the physical buffer and the queue length of each pseudo buffer are decremented by one if they are not zero. The status of the server is reset to be idle.
pesudo_arrival	This state is entered when a synthetic cell arrives. The queue length of each pseudo buffer is updated. For pseudo buffer *i*, its queue length is incremented by *i/K* if there are spaces left in the buffer, where *K* is the number of pseudo buffers. Otherwise, its queue length is set to the buffer size specified in the "buffer size" model attribute and the overflow status of that pseudo buffer is set to be overflow.
cac	This state is entered when a call setup request is received by this process. The procedure "cac_available_bandwidth_setup()" is called.
release	This state is entered when a call release request is received by this process. The procedure "cac_available_bandwidth_release()" is called.
update	This state is entered at the end of each available bandwidth update time interval. The procedure "cac_available_bandwidth_update()" is called.
file_refresh	This state is entered every 60 simulation seconds to close and reopen the data file and the log file to prevent them becoming stale and unreadable from being opened too long.
sim_end	This state is entered just before the end of the simulation. The data file and the log file are closed. The number of cells received by this process and the number of cells discarded by this process during the simulation are printed out to the standard output. The cell loss ratio is calculated as the number of discarded cells

	divided by the number of received cells. The cell loss ratio is also printed out to the standard output.

Table 5-1. State descriptions of the process model of the new CAC scheme

Transition Condition Descriptions

Transition Condition	Description
PHYSICAL_BUFFER_ARRIVAL	A cell from one of the connected calls arrives.
PSEUDO_BUFFER_ARRIVAL	A synthetic cell arrives.
SERVER_IDLE	The server is idle.
ALL_QUEUES_EMPTY	The physical buffer and all pseudo buffers are empty.
SVC_COMPLETION	A cell transmission is completed.
ESTABLISH_CONN_REQUEST	A call setup request is received by the process.
RELEASE_CONN_REQUEST	A call release request is received by the process.
UPDATE_AVAILABLE_BANDWIDTH	An available bandwidth update time interval is over. It is time for the process to update the available bandwidth.
FILE_REFRESH	Sixty simulation seconds have passed. It's time for the process to close and reopen the data and log files.
SIMULATION_END	The simulation is either completed or stopped by the user.

Table 5-2. Transition condition descriptions of the process model of the new CAC scheme

Procedures Descriptions

Procedure	Description
cac_available_bandwidth_setup()	This procedure decides whether or not to accept a new call setup request by comparing the call's peak bandwidth to the estimated available bandwidth. The request is accepted and an "ACCEPTED" notification is sent to the traffic source if the call's peak bandwidth is less than or

	equal to the estimated available bandwidth. Otherwise, the request is rejected and a "REJECTED" notification is sent to the traffic source. If the call setup request is accepted, the number of connected calls is incremented by one and the call's peak bandwidth is subtracted from the available bandwidth. The call's identification and its average bandwidth are saved in a record which is appended to the end of a list of connected calls.
cac_available_bandwidth_release()	This procedure is called when a call release request is received by this process. The call is released and the number of connected calls is decremented by one. The record of this call is located and deleted from the list of connected calls. The call's average bandwidth is added to the available bandwidth.
cac_available_bandwidth_update()	This procedure estimates the available bandwidth by checking the overflow status of each pseudo buffer at the end of an available bandwidth update time interval. The overflow status of each pseudo buffer is then reset to be not-overflow.

Table 5-2. Procedure descriptions of the process model of the new CAC scheme

5.4.2 IMPLEMENTATION OF TWO OTHER CAC SCHEMES

The performance of the new CAC scheme will be compared to the performance of two other CAC schemes, the peak bandwidth reservation and the effective bandwidth reservation, in the next chapter. The peak bandwidth reservation is selected since it is the most commonly used CAC scheme in the implementations of existing ATM switch management software due to its simplicity and its guarantees on CLR QoS. The effective bandwidth reservation is selected since it is reported to have the best performance among

all available CAC schemes and is widely discussed in the literature. Both CAC schemes are implemented as process models in OPNET. The process models of both CAC schemes have the same state transition diagram. The only difference between them is the algorithms used in the call setup and release procedures. The state transition diagram of both process models is shown in Figure 5-4. The process model has 5 model attributes as listed in Figure 5-5. The capacity of the output link can be specified by the "output link capacity" model attribute. The size of the output buffer can be specified by the "buffer size" model attribute. The CLR QoS requirement can be specified by the "cell loss ratio requirement" model attribute. This process model generates a data file and a log file during a simulation. The name of the data file can be specified by the "data file name" model attribute. The name of the log file can be specified by the "log file name" model attribute. There are eight states in this process model: seven forced (init, arrival, svc_start, svc_comp, cac, release, and file_refresh), and two unforced (idle and sim_end). They are described in Table 5-4. The process model has 9 transition conditions which are elaborated in Table 5-5. Two procedures are used in the enter and transition executives of the process model for the CAC scheme using peak bandwidth reservation and three procedures are used in the enter and transition executives of the process model for the CAC scheme using effective bandwidth reservation. Their descriptions are given in Table 5-6.

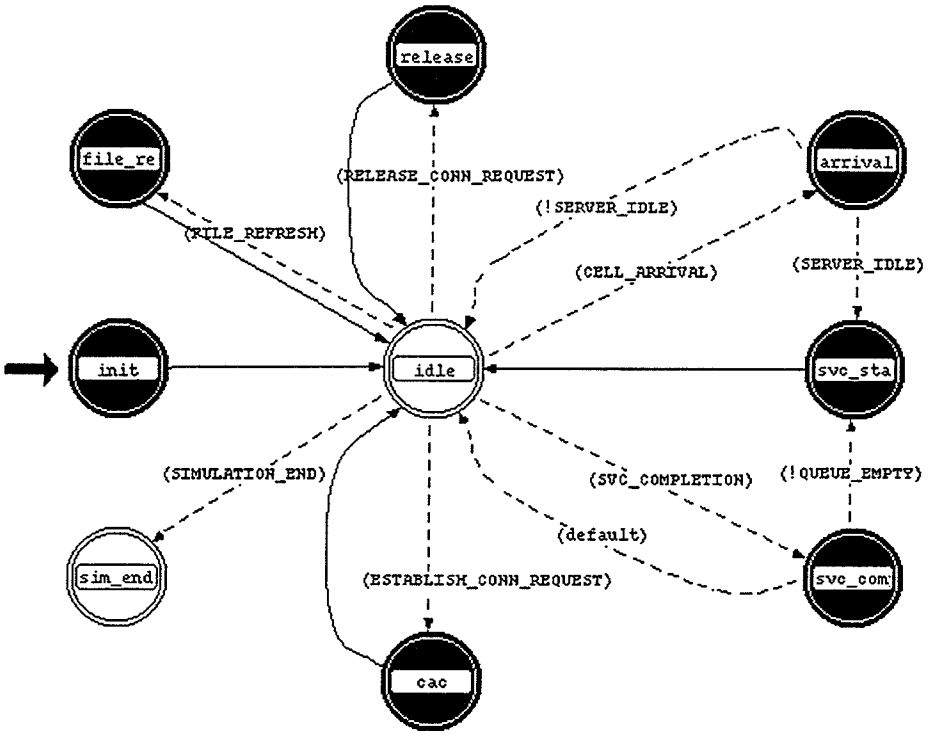

Figure 5-4. State transition diagram of the process model of the CAC schemes using peak bandwidth reservation and effective bandwidth reservation

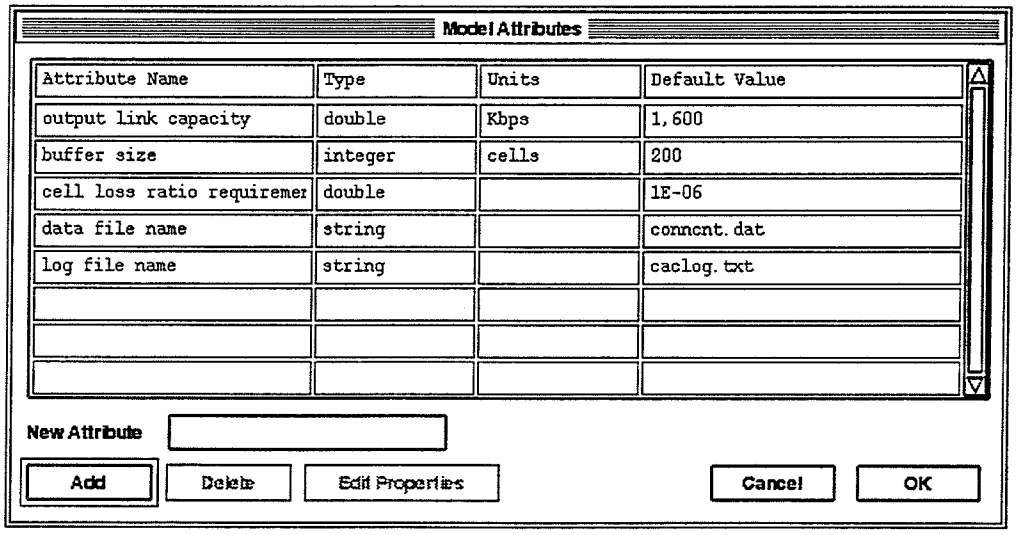

Figure 5-5. Model attributes of the process model of the CAC schemes using peak bandwidth reservation and effective bandwidth reservation

State Descriptions

State	Description
init	The process starts at this state where user-supplied model attribute values are read and saved in state variables. This process also provides a vector statistic which is the number of connected calls with respect to the progress of the simulation. Several counters, such as the number of connected calls, the number of received call setup request, the number of received cells and the number of discarded cells are initialized. A list which is used to save the record of the identification and the average bandwidth of each connected call is created. A data file and a log file is also created and opened. The name of the data file can be specified by the "data file name" model attribute. The name of the log file can be specified by the "log file name" model attribute. A self interrupt is scheduled in a fixed period of time to let the process close and reopen the data and log files for the first time to prevent them becoming stale and unreadable from being opened too long. The length of this fixed period of time is set to be 60 seconds by default.
idle	This state is entered right after the process completes all the actions at one of the forced states. The process rests in this state and waits to be invoked again.
arrival	This state is entered when a cell from one of the accepted calls arrives. The number of cells received by this process is incremented by one. If the output buffer is full, the number of cells discarded by this process is incremented by one. Otherwise, the queue length of the output buffer is incremented by one.
svc_start	This state is entered right after the process leaves the arrival state if the server is idle, or right after the process leaves the svc_comp state if the output buffer is not empty. A cell transmission starts and the status of the server is set to be busy. A self interrupt is scheduled in a fixed period of time to mark the end of this cell transmission. The length of this fixed period time is set as the cell length in Kb divided by the capacity of the output link.
svc_comp	This state is entered when a cell transmission is completed. The queue length of the output buffer is decremented by one and a Boolean variable called "QUEUE_EMPTY" is set to be false if the buffer is not empty. Otherwise, the variable is set to be true. The status of the server is set to be idle.
cac	This state is entered when a call setup request is received. For the CAC scheme using peak bandwidth reservation, the procedure "cac_peak_band_width_setup()" is called. For the CAC scheme using effective bandwidth reservation, the procedure "cac_effective_bandwidth_setup()" is called.
release	This state is entered when a call release request is received. For the CAC scheme using peak bandwidth reservation, the procedure

	"cac_peak_band_width_release()" is called. For the CAC scheme using effective bandwidth reservation, the procedure "cac_effective_bandwidth_release()" is called.
file_refresh	This state is entered every 60 seconds simulation time to close and reopen the data file and the log file in order to prevent them becoming stale and unreadable from being opened too long.
sim_end	This state is entered just before the end of the simulation. The data file and the log file are closed. The number of cells received by this process and the number of cells discarded by this process during the simulation are printed out to the standard output. The cell loss ratio is calculated as the number of discarded cells divided by the number of received cells. It is also printed out to the standard output.

Table 5-4. State descriptions of the process model of the CAC schemes using peak bandwidth reservation and effective bandwidth reservation

Transition Condition Descriptions

Transition Condition	Description
CELL_ARRIVAL	A cell arrives.
SERVER_IDLE	The server is idle.
QUEUE_EMPTY	The output buffer is empty.
SVC_COMPLETION	A cell transmission is completed.
ESTABLISH_CONN_REQUEST	A call setup request is received.
RELEASE_CONN_REQUEST	A call release request is received.
FILE_REFRESH	Sixty simulation seconds have passed. It's time for the process to close and reopen the data and log files.
SIMULATION_END	The simulation is either completed or stopped by the user.

Table 5-5. Transition condition descriptions of the process model of the CAC schemes using peak bandwidth reservation and effective bandwidth reservation

Procedures Descriptions

Procedure	Description
cac_peak_bandwidth_setup()	This procedure decides whether or not to accept a new call setup request by comparing the call's peak bandwidth to the available bandwidth. The request is

	accepted and an "ACCEPTED" notification is sent to the traffic source if the call's peak bandwidth is less than or equal to the available bandwidth. Otherwise, the request is rejected and a "REJECTED" notification is sent to the traffic source. If the call setup request is accepted, the number of connected calls is incremented by one and the call's peak bandwidth is subtracted from the available bandwidth. The call's identification and its peak bandwidth are saved in a record which is appended to the end of a list of connected calls.
cac_peak_bandwidth_release()	This procedure is called when a call release request is received by this process. The call is released and the number of connected calls is decremented by one. The call's record is located and deleted from the list of connected calls. The call's peak bandwidth is added to the available bandwidth.
cac_effective_bandwidth_setup()	This procedure decides whether or not to accept a new call setup request by comparing the call's effective bandwidth to the available bandwidth. The request is accepted and an "ACCEPTED" notification is sent to the traffic source if the call's effective bandwidth is less than or equal to the available bandwidth. Otherwise, the request is rejected and a "REJECTED" notification is sent to the traffic source. If the call setup request is accepted, the number of connected calls is incremented by one and the call's effective bandwidth is subtracted from the available bandwidth. The call's identification and its effective bandwidth are saved in a record which is appended to the end of a list of connected calls.
cac_effective_bandwidth_release()	This procedure is called when a call release request is received by this process. The call is released and the number of connected calls is decremented by one. The call's record is located and deleted from the list of connected calls. The call's effective bandwidth is added to the available

	bandwidth.
cac_effective_bandwidth_calculate()	This procedure calculates the effective bandwidth of a new call depending on the traffic type and the traffic parameter values of the call.

Table 5-6. Procedure descriptions of the process model of the CAC schemes using peak bandwidth reservation and effective bandwidth reservation

5.5 SUMMARY

In this chapter, the approach of designing CAC schemes based on estimating the available bandwidth of an output link is explored. A new CAC scheme based on this approach is developed. No specific traffic types are assumed and the new CAC scheme is able to operate in real-time. A mandatory traffic parameter, PCR, and an optional traffic parameter, SCR, are required for each call. The advantages and the disadvantages of the new CAC scheme are discussed. The new CAC scheme and two other CAC schemes, the peak bandwidth reservation and the effective bandwidth reservation, are implemented as process models in OPNET.

CHAPTER SIX

PERFORMANCE EVALUATIONS OF THE NEW CAC SCHEME

In this chapter, performance evaluations of the new CAC scheme are conducted. The new CAC scheme is also compared to two other CAC schemes under various scenarios. This chapter is organized as follows. In section 6.1, the simulation model used in the CAC scheme performance evaluations and comparisons is described. In section 6.2, the effects of the length of each available bandwidth update time interval, the output buffer size, PCR over-specification, and traffic specification without SCR, on the performance of the new CAC scheme are studied. In section 6.3, comparisons are performed on the new CAC scheme and two other CAC schemes, peak bandwidth reservation and effective bandwidth reservation, under five different types of traffic arrival processes: multiplexed deterministic ON/OFF traffic sources, multiplexed IPP traffic sources, multiplexed

MMPP traffic sources, multiplexed MMFP traffic sources, and multiplexed self-similar traffic sources. Section 6.4 summarizes this chapter.

6.1 CAC SIMULATION MODEL

An ATM switch has a number of input ports and output ports. Those input ports and output ports are connected by a switching fabric which is composed of identical switching elements interconnected in a specific topology. Cells are taken from the input ports, passed through the switching fabric, and sent to the appropriate output port. All ATM switches have buffers to temporarily store cells when two or more cells from different input ports contend for the same output port at the same time slot or when cell arrival rate exceeds the capacity of an output link. Buffers can be provided at the input ports (input buffering) or/and output ports (output buffering). With input buffering, it is possible that cells are blocked at an input port, waiting for another cell in the same input buffer to be served, even when those cells are destined for different output ports. This is the so-called 'head of the line" problem [PRYC95]. Because of this, output buffering is more commonly used. The architecture of an ATM switch with output buffering is shown in Figure 6-1.

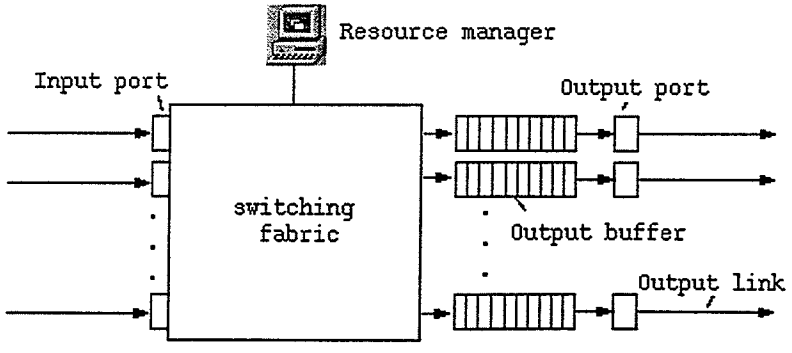

Figure 6-1. An ATM switch with output buffering

For ATM switches with output buffering, the points of congestion are at those output ports. Call admission control has to be applied to each output port to prevent congestion from happening. The simulation model used in the CAC scheme performance evaluations and comparisons is therefore obtained by isolating one of the output ports and its output buffer from the rest of the ATM switch. The simulation model is illustrated in Figure 6-2.

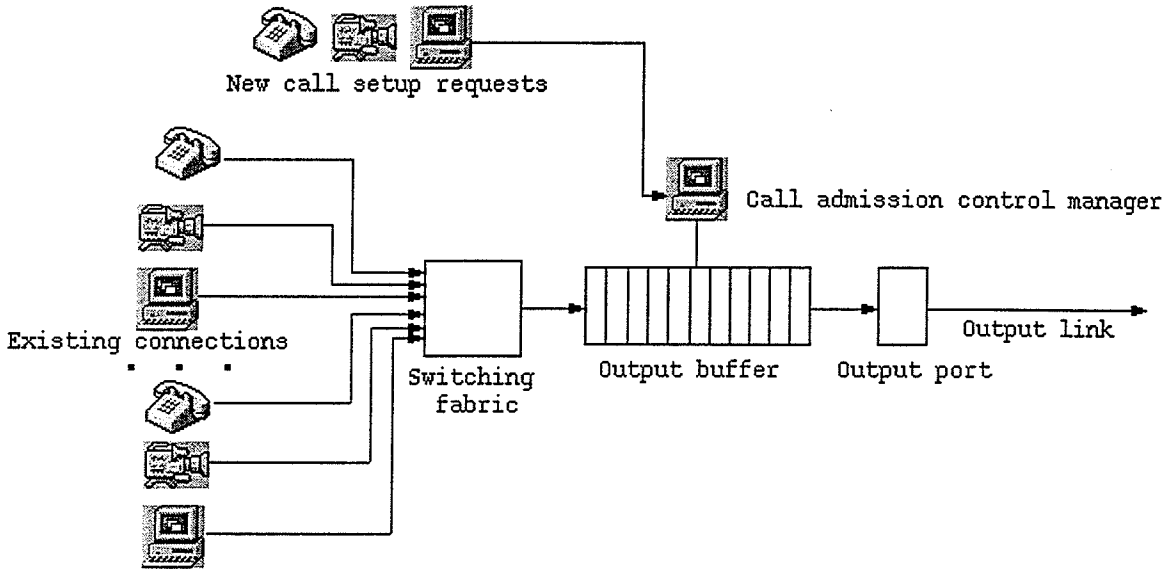

Figure 6-2. A simulation model for CAC scheme performance evaluations and comparisons.

The output port is connected to an output link. In all simulations conducted for the CAC scheme performance evaluations and comparisons, the capacity of the output link is set to be 1.6 Mbps. This value is much smaller than the capacity of a real ATM output link which is typically 150 Mbps or 622 Mbps. However, due to the limited computation power of todays' workstations, 1.6Mbps is chosen as a tradeoff for the length of the simulations. The length of each simulation is set to be an hour, which brings the amount

of real time needed to complete a simulation run to approximately an hour. The buffer size is set to be 200 cells for all simulations, except for one simulation run which studies the effect of the buffer size on the performance of the new CAC scheme. This value is chosen to bound the maximum cell delay within 53 milliseconds. The cell delay QoS requirements are then assumed to be always satisfied and the performance of a CAC scheme is judged from two criteria: the cell loss ratio and the output link utilization. For each simulation, there are 200 traffic sources making calls and sending ATM cells to the output port. The average call holding time is set to be 200 seconds and the average call interarrival time is also set to be 200 seconds. The traffic intensity is set to be approximately one and a half times of the capacity of the output link. If a call setup request from a traffic source gets rejected, that request is repeated every 10 seconds until a connection is made. The above two measures ensure that almost at any time, there are calls waiting to be connected. For all simulations of the performance evaluations of the new CAC scheme, the length of each available bandwidth update time interval is set to be 15 seconds except for the one simulation run which studies the effect of the length of each available bandwidth update time interval on the performance of the new CAC scheme. For each simulation of the CAC scheme comparisons, the length of each available bandwidth update time interval is optimized to be a multiple of 5 seconds to maximize the output link utilization while no cell loss is allowed. All calls are assumed to have the same CLR QoS requirement which is set to be 10^{-6}. For the new CAC scheme, the basic data rate of the synthetic CBR traffic is set to be 16 Kbps, which brings the number of pseudo buffers to 100.

6.2 PERFORMANCE EVALUATIONS OF THE NEW CAC SCHEME

6.2.1 Traffic Sources Used In the Performance Evaluation

Throughout the performance evaluation of the new CAC scheme, each traffic source is configured as a self-similar traffic generator. Traffic produced by such a traffic source has an SCR of 50 cells per second, a PCR of 200 cells per second, and a Hurst parameter estimated as 0.78612.

6.2.2 Effect of the Length of Each Available Bandwidth Update Time Interval

Intuitively, the longer the length of each available bandwidth update time interval is, the more likely cell losses can be observed at a certain pseudo buffer. From the fact that the available bandwidth is estimated from the overflow experience of a set of pseudo buffers during an available bandwidth update time interval, it can be expected that as the length of each available bandwidth update time interval increases, the available bandwidth estimated at the end of each available bandwidth update time interval decreases. As a result, the number of connections and the output link utilization decrease. Figure 6-3 shows the number of connections accepted by the new CAC scheme along with the progress of the simulation for four different choices of the length of each available bandwidth update time interval. Table 6-1 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each choice of the length of each available bandwidth update time interval.

Simulation results show that as the length of each available bandwidth update time interval increases linearly, the number of accepted connections and the output link utilization decrease linearly but at a much slower rate while the CLR decreasing exponentially fast. This indicates that setting the length of each available bandwidth update time interval to an appropriate value is vital for the new CAC scheme to achieve the required CLR QoS. Setting the length of each update time interval to be too small can cause a relatively large amount of cells to be discarded and result in a degraded CLR QoS. On the other hand, setting the length of each update time interval too large slowly decreases the output link utilization, but at the same time, improves the CLR QoS.

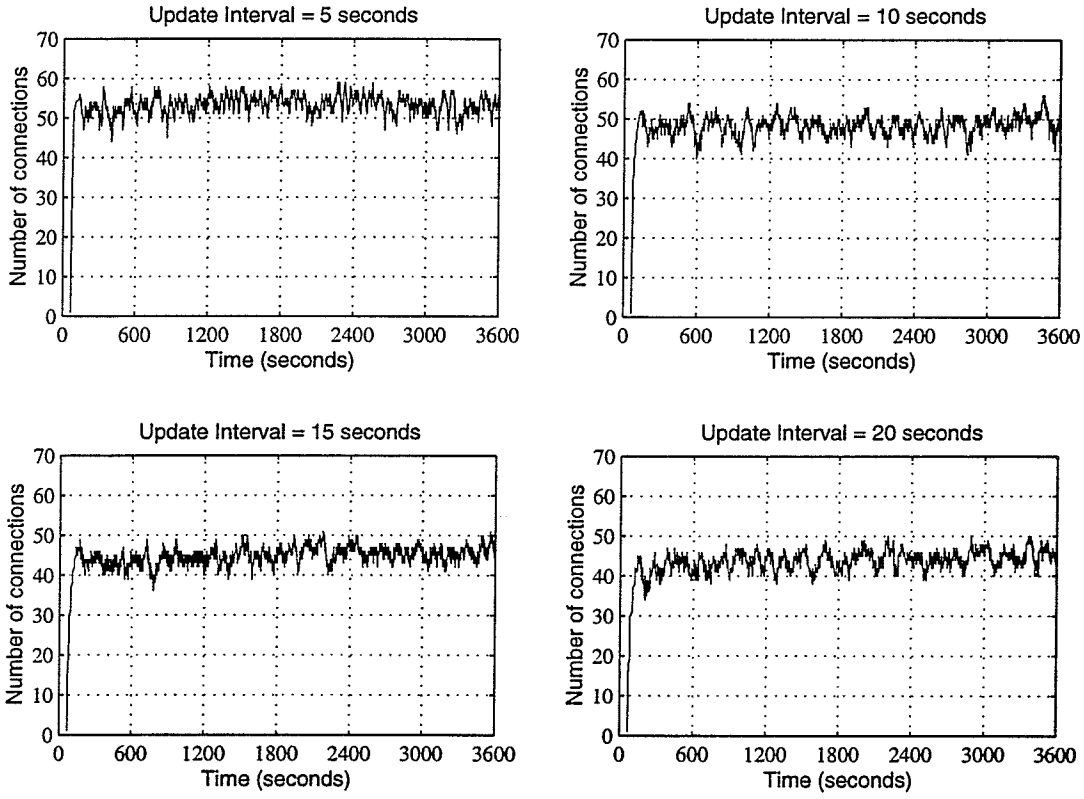

Figure 6-3. Effect of the available bandwidth update time interval length on the performance of the new CAC scheme

	Available bandwidth update time interval (seconds)			
	5	10	15	20
Maximum number of accepted connections	59	56	51	50
Average number of accepted connections	52.25	47.53	44.12	43.05
Number of cells received	11,700,077	10,738,295	9,994,764	9,649,573
Number of cells discarded	3173	82	0	0
Cell loss ratio	0.0002712	0.00000763	0	0
Output link utilization	86.13%	79.05%	73.57%	71.03%

Table 6-1. Simulation results of the effect of the available bandwidth update time interval length on the performance of the new CAC scheme

6.2.3 Effect of the Buffer Size

It can be expected that the number of connections and the output link utilization increase as the buffer size increases. However, the increment should be quite small since the buffer is mainly used to cope with cell-level congestion. For a well-designed CAC scheme, the buffer size should have very little effect on the observed CLR. Figure 6-4 shows the number of connections accepted by the new CAC scheme along with the progress of the simulation for four different choices of the buffer size. Table 6-2 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each choice of the buffer size. Simulation results show that as the buffer size increases, the number of accepted connections and the output link utilization increase very slightly, and at the same time, the required CLR QoS is always well maintained. This study indicates that the buffer size has very little effect on the performance of the new CAC scheme. Therefore, small or

medium sized buffers can be chosen to limit the maximum CTD and peak to peak CDV without decreasing the output link utilization considerably.

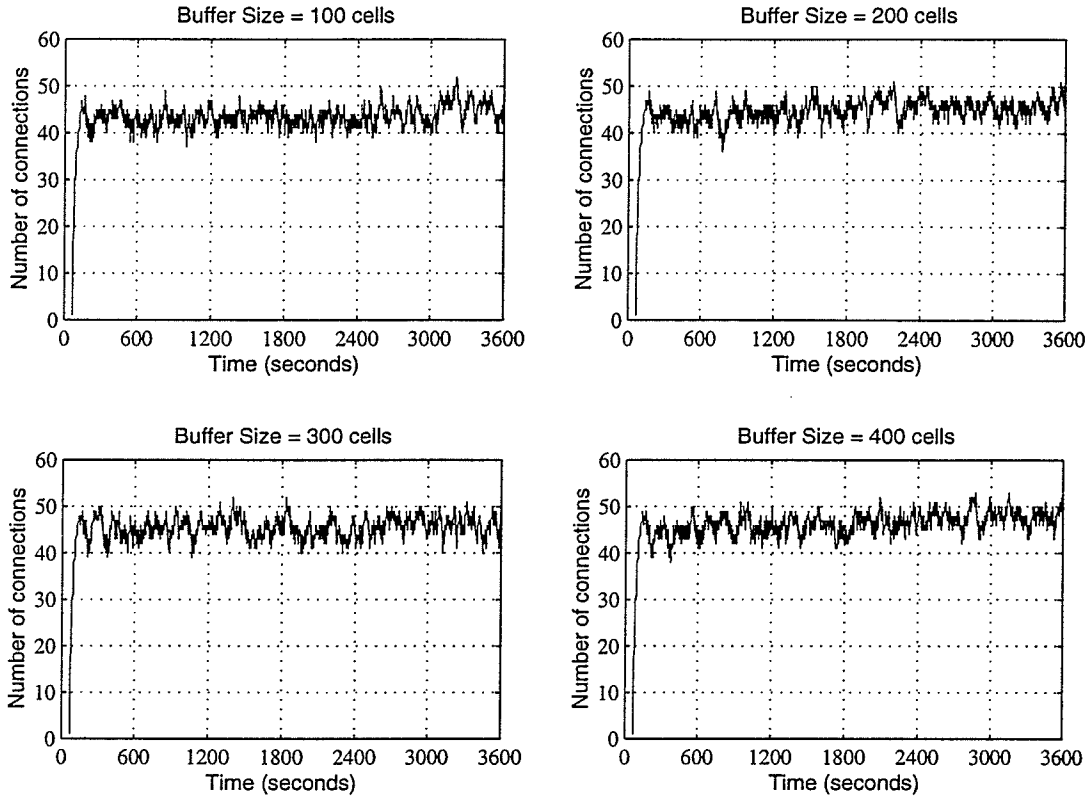

Figure 6-4. Effect of the buffer size on the performance of the new CAC scheme

	Buffer size (cells)			
	100	200	300	400
Maximum number of accepted connections	52	51	52	53
Average number of accepted connections	42.82	44.12	44.79	45.63
Number of cells received	9,615,302	9,994,764	10,132,661	10,299,263
Number of cells discarded	0	0	0	0
Cell loss ratio	0	0	0	0
Output link utilization	70.78%	73.57%	75.59%	75.81%

Table 6-2. Simulation results of the effect of the buffer size on the performance of the new CAC scheme

6.2.4 Effect of PCR Over-specification

In reality, it could be difficult for users to specify the PCR of their calls accurately. As a result, users may over-specify the PCR of their calls in order to avoid the risk of under-specifying it and getting punished by UPC. Therefore, it is important to study the effect of PCR over-specification on the performance of the new CAC scheme. Figure 6-5 shows the number of connections accepted by the new CAC scheme along with the progress of the simulation for four different specified PCR values. Table 6-3 provides detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each specified PCR value. The actual value for the PCR of each call is 200 cells per second. Simulation results show that the number of accepted connections and the output link utilization decrease as the specified PCR value increases. This decrement results from the fact that the new CAC scheme assigns each accepted call its peak bandwidth at the call setup stage and the available bandwidth is adjusted only at the end of each update time interval. However, compared to the rate of increment of the specified PCR value, the rates of decrement of the number of connections and the output link utilization are quite small. This study indicates that the new CAC scheme can still achieve a satisfying efficiency even when users over-specify the PCR of their calls. This feature makes the new CAC scheme more attractive than other CAC schemes in the presence of PCR over-specification.

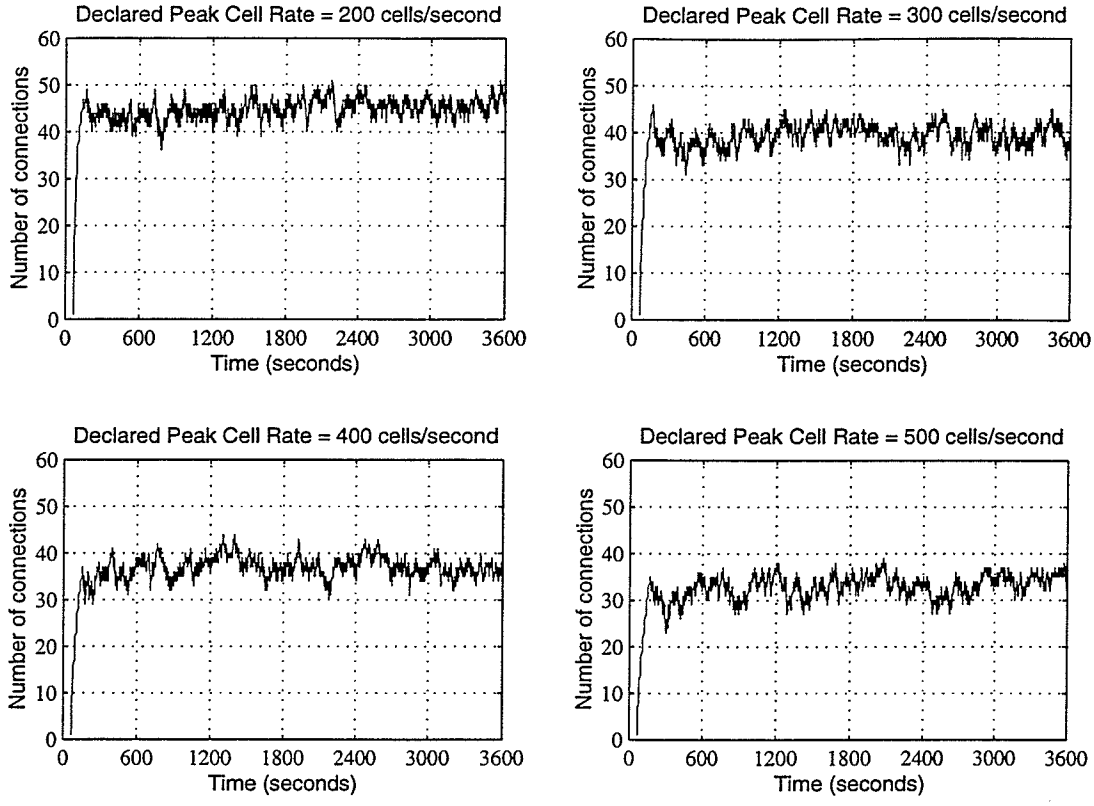

Figure 6-5. Effect of PCR over-specification on the performance of the new CAC scheme

	User-specified peak cell rate (cells)			
	200	300	400	500
Maximum number of accepted connections	51	46	44	39
Average number of accepted connections	44.12	38.73	36.27	32.34
Number of cells received	9,994,764	8,709,039	8,140,955	7,242,993
Number of cells discarded	0	0	0	0
Cell loss ratio	0	0	0	0
Output link utilization	73.57%	64.11%	59.93%	53.32%

Table 6-3. Simulation results of the effect of PCR over-specification on the performance of the new CAC scheme

6.2.5 Effect of Traffic Specification without SCR

For users, specifying the PCR of their calls accurately is not easy, and specifying the SCR of their calls accurately could be even more difficult. It is then desirable to use PCR as the only traffic parameter. The graph on the left side of Figure 6-6 shows the number of connections accepted by the new CAC scheme along with the progress of the simulation when each call is specified by two traffic parameters, PCR and SCR. The graph on the right side of Figure 6-6 shows the number of connections accepted by the new CAC scheme along with the progress of the simulation when each call is specified by its PCR only. In the latter case, no bandwidth instead of the average bandwidth is added to the available bandwidth when a call is released. Table 6-4 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for both cases. Simulation results show that the number of accepted connections and the output link utilization decrease if the SCR of each call is not specified. However, compared to the percentage of the SCR to the PCR, the percentage of decrement of the number of accepted connections or the output link utilization is negligible. This study indicates that the new CAC scheme can still work at a high efficiency when PCR is the only traffic parameter specified by users. This feature is very appealing since it removes the difficulty of estimating the SCR of a call from users.

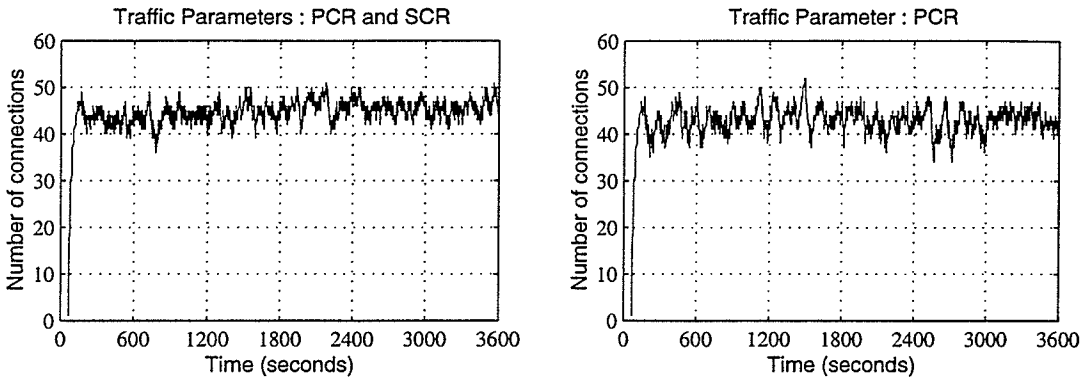

Figure 6-6. Effect of traffic specification without SCR on the performance of the new CAC scheme

	User-specified traffic parameters	
	PCR and SCR	PCR only
Maximum number of accepted connections	51	52
Average number of accepted connections	44.12	42.25
Number of cells received	9,994,764	9,519,855
Number of cells discarded	0	0
Cell loss ratio	0	0
Output link utilization	73.57%	70.08%

Table 6-4. Simulation results of the effect of traffic specification without SCR on the performance of the new CAC scheme

6.3 CAC SCHEME COMPARISONS

6.3.1 A Comparison Under the Traffic Arrival Process of Multiplexed Deterministic ON/OFF Traffic Sources

In this section, a CAC scheme comparison is conducted under the traffic arrival process of multiplexed deterministic ON/OFF traffic sources. Each deterministic ON/OFF traffic source has an SCR of 50 cells per second, a PCR of 250 cells per second, an average

burst length of 1 second, and a Hurst parameter estimated as 0.50869. The optimized value for the length of each available bandwidth update time interval for the new CAC scheme under this traffic arrival process is found to be 25 seconds. For the CAC scheme using effective bandwidth reservation, the effective bandwidth of each individual traffic source is calculated using the formula (1) given in section 4.2.2, and the effective bandwidth of the aggregate traffic is obtained using the formula (2) given in section 4.2.2. Figure 6-7 shows the number of connections as well as the average number of connections accepted by each of the three CAC schemes under comparison. Table 6-5 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each CAC scheme. Simulation results show that the output link utilization achieved by the new CAC scheme is approximately a 30 percent improvement of that achieved by the CAC scheme using effective bandwidth reservation, and is nearly a 130 percent improvement of that achieved by the CAC scheme using peak bandwidth reservation.

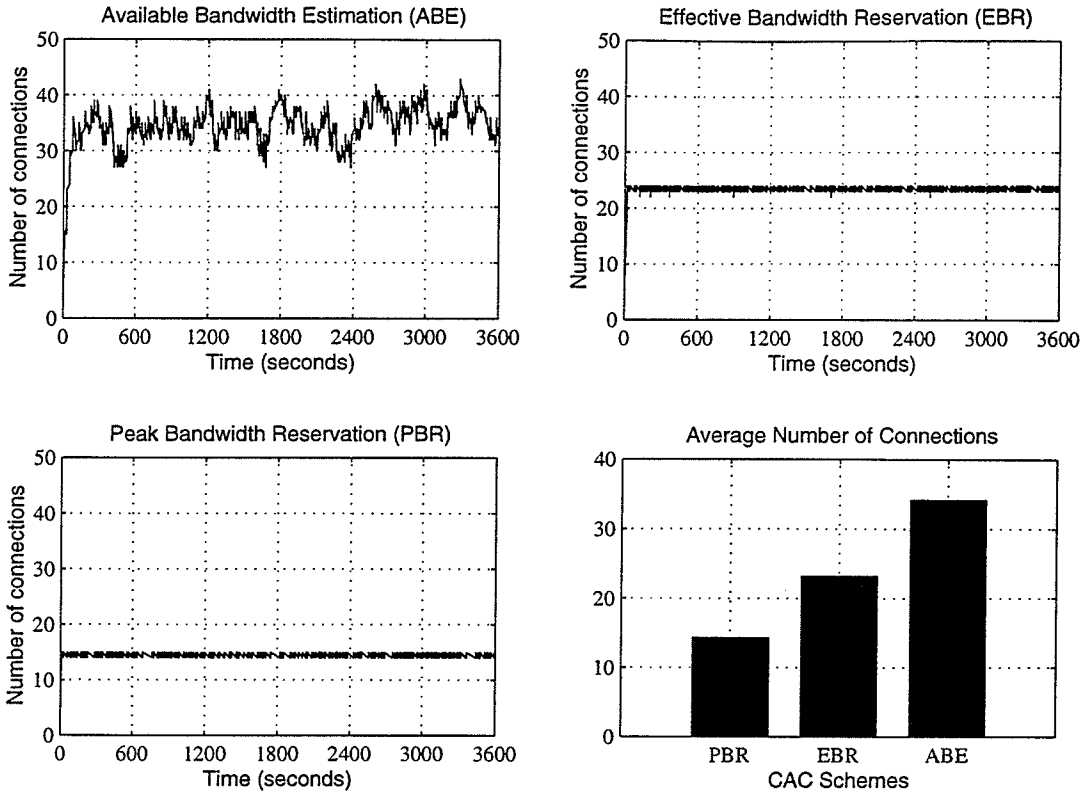

Figure 6-7. CAC scheme comparison under the traffic arrival process of multiplexed deterministic ON/OFF traffic sources

	CAC scheme		
	ABE	EBR	PBR
Maximum number of accepted connections	43	24	15
Average number of accepted connections	34.19	23.17	14.32
Number of cells received	6,346,715	4,294,156	2,696,174
Number of cells discarded	0	0	0
Cell loss ratio	0	0	0
Output link utilization	46.72%	31.61%	19.85%

Table 6-5. Simulation results of the CAC scheme comparison under the traffic arrival process of multiplexed deterministic ON/OFF traffic sources

6.3.2 A Comparison Under the Traffic Arrival Process of Multiplexed IPP Traffic Sources

In this section, a CAC scheme comparison is conducted under the traffic arrival process of multiplexed IPP traffic sources. Each IPP traffic source has an SCR of 50 cells per second, a PCR of 350 cells per second, an average burst length of 1 second, and a Hurst parameter estimated as 0.53097. The optimized value of the length of each available bandwidth update time interval for the new CAC scheme under this traffic arrival process is found to be 25 seconds. For the CAC scheme using effective bandwidth reservation, the effective bandwidth of each individual traffic source is calculated using the formula (4) given in section 4.2.2, and the effective bandwidth of the aggregate traffic is obtained by adding the effective bandwidth of each individual traffic source. Figure 6-8 shows the number of connections as well as the average number of connections accepted by each of the three CAC schemes under comparison. Table 6-6 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each CAC scheme. Simulation results show that the output link utilization achieved by the new CAC scheme is approximately a 110 percent improvement of that achieved by the CAC scheme using effective bandwidth reservation, and is nearly a 200 percent improvement of that achieved by the CAC scheme using peak bandwidth reservation.

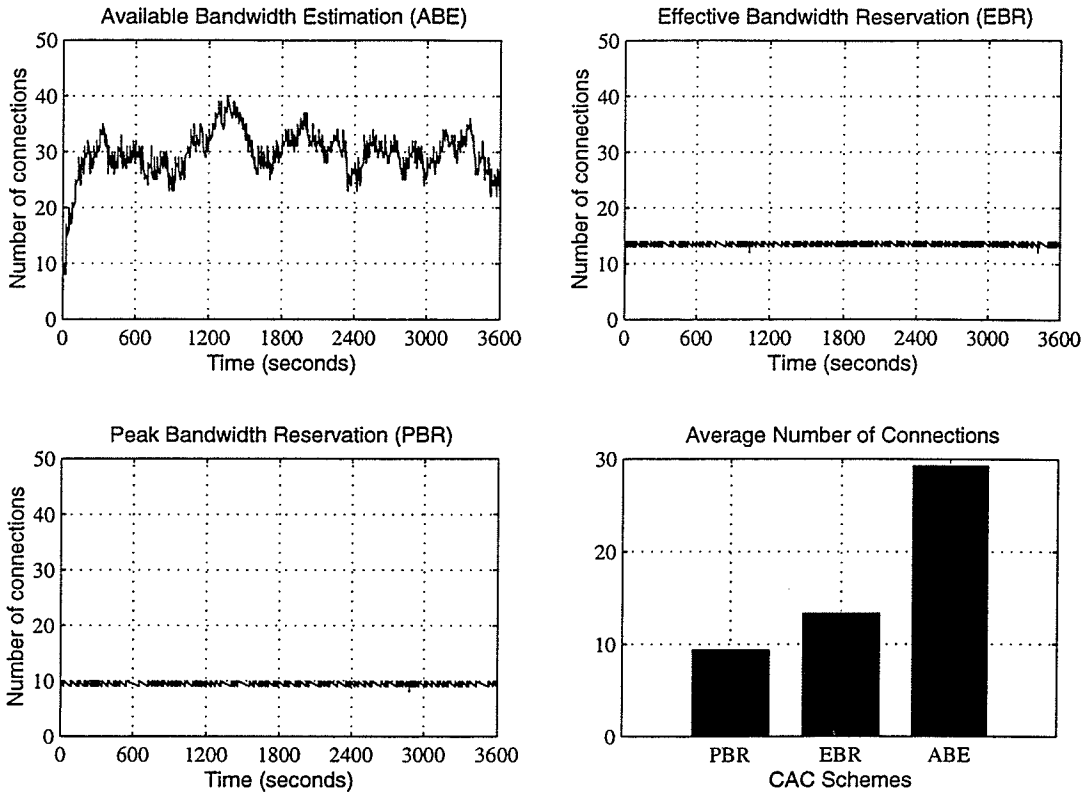

Figure 6-8. CAC scheme comparison under the traffic arrival process of multiplexed IPP traffic sources

	CAC scheme		
	ABE	EBR	PBR
Maximum number of accepted connections	40	14	10
Average number of accepted connections	29.28	13.33	9.37
Number of cells received	5,408,009	2,547,169	1,776,373
Number of cells discarded	0	0	0
Cell loss ratio	0	0	0
Output link utilization	39.81%	18.75%	13.08%

Table 6-6. Simulation results of the CAC scheme comparison under the traffic arrival process of multiplexed IPP traffic sources

6.3.3 A Comparison Under the Traffic Arrival Process of Multiplexed MMPP Traffic Sources

In this section, a CAC scheme comparison is conducted under the traffic arrival process of multiplexed MMPP traffic sources. Each MMPP traffic source has an SCR of 50 cells per second, a PCR of 300 cells per second, and a Hurst parameter estimated as 0.50893. The optimized value for the length of each available bandwidth update time interval for the new CAC scheme under this traffic arrival process is found to be 20 seconds. For the CAC scheme using effective bandwidth reservation, the effective bandwidth of each individual traffic source is calculated using the formula (4) given in section 4.2.2, and the effective bandwidth of the aggregate traffic is obtained by adding the effective bandwidth of each individual traffic source. Figure 6-9 shows the number of connections as well as the average number of connections accepted by each of the three CAC schemes under comparison. Table 6-7 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each CAC scheme. Simulation results show that the output link utilization achieved by the new CAC scheme is approximately a 120 percent improvement of that achieved by the CAC scheme using effective bandwidth reservation, and is nearly a 210 percent improvement of that achieved by the CAC scheme using peak bandwidth reservation.

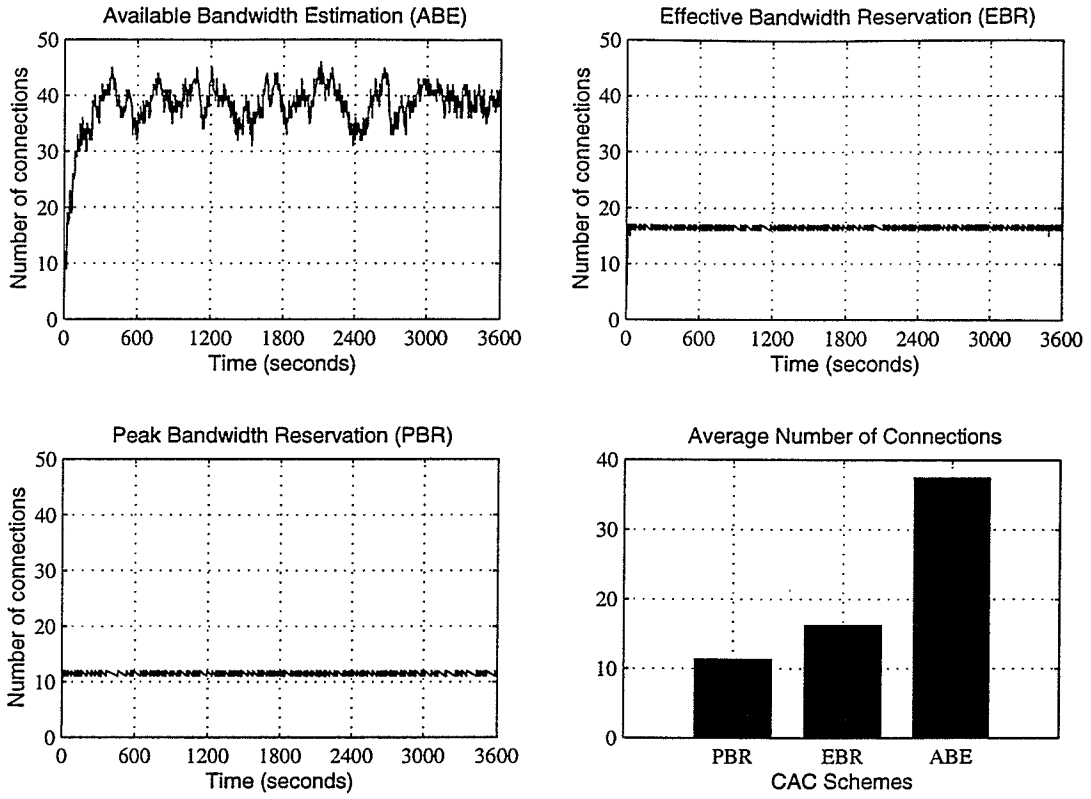

Figure 6-9. CAC scheme comparison under the traffic arrival process of multiplexed MMPP traffic sources

	CAC scheme		
	ABE	EBR	PBR
Maximum number of accepted connections	46	17	12
Average number of accepted connections	37.46	16.25	11.36
Number of cells received	6,913,546	3,068,706	2,187,067
Number of cells discarded	0	0	0
Cell loss ratio	0	0	0
Output link utilization	50.89%	22.59%	16.10%

Table 6-7. Simulation results of the CAC scheme comparison under the traffic arrival process of multiplexed MMPP traffic sources

6.3.4 A Comparison Under the Traffic Arrival Process of Multiplexed MMFP Traffic Sources

In this section, a CAC scheme comparison is conducted under the traffic arrival process of multiplexed MMFP traffic sources. Each MMFP traffic source has an SCR of 50 cells per second, a PCR of 210 cells per second, and a Hurst parameter estimated as 0.54178. The optimized value for the length of each available bandwidth update time interval for the new CAC scheme under this traffic arrival process is found to be 25 seconds. For the CAC scheme using effective bandwidth reservation, the effective bandwidth of each individual traffic source is calculated using the formula (3) given in section 4.2.2, and the effective bandwidth of the aggregate traffic is obtained by adding the effective bandwidth of each individual traffic source. Figure 6-10 shows the number of connections as well as the average number of connections accepted by each of the three CAC schemes under comparison. Table 6-8 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each CAC scheme. Simulation results show that the output link utilization achieved by the new CAC scheme is approximately a 110 percent improvement of that achieved by the CAC scheme using effective bandwidth reservation, and is nearly a 140 percent improvement of that achieved by the CAC scheme using peak bandwidth reservation.

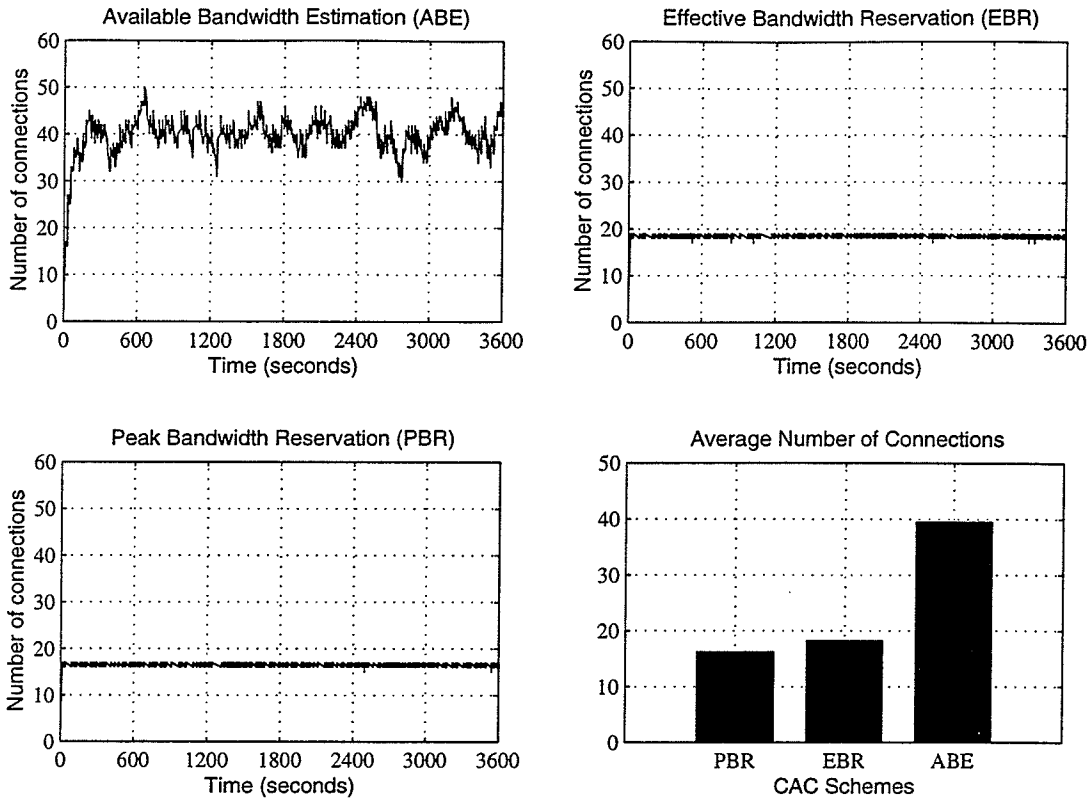

Figure 6-10. CAC scheme comparison under the traffic arrival process of multiplexed MMFP traffic sources

	CAC scheme		
	ABE	EBR	PBR
Maximum number of accepted connections	50	19	17
Average number of accepted connections	39.51	18.23	16.27
Number of cells received	7,211,559	3,435,714	3,030,503
Number of cells discarded	0	0	0
Cell loss ratio	0	0	0
Output link utilization	53.09%	25.29%	22.31%

Table 6-8. Simulation results of the CAC scheme comparison under the traffic arrival process of multiplexed MMFP traffic sources

6.3.5 A Comparison Under the Traffic Arrival Process of Multiplexed Self-Similar Traffic Sources

In this section, a CAC scheme comparison is conducted under the traffic arrival process of multiplexed self-similar traffic sources. Each self-similar traffic source is obtained by multiplexing 50 heavy-tailed ON/OFF sources, and has an SCR rate of 50 cells per second, a PCR of 300 cells per second, and a Hurst parameter estimated as 0.70012. The optimized value for the length of each available bandwidth update time interval for the new CAC scheme under this traffic arrival process is found to be 10 seconds. For the CAC scheme using effective bandwidth reservation, the effective bandwidth of each heavy-tailed ON/OFF traffic source is estimated using the formula given (1) in section 4.2.2, the effective bandwidth of each individual self-similar traffic source as well as the effective bandwidth of the aggregate traffic is obtained using the formula (2) given in section 4.2.2. Figure 6-11 shows the number of connections as well as the average number of connections accepted by each of the three CAC schemes under comparison. Table 6-9 provides more detailed information on the maximum and the average number of accepted connections, the CLR, and the output link utilization for each CAC scheme. Simulation results show that the output link utilization achieved by the new CAC scheme is approximately the same as that achieved by each of the CAC scheme using effective bandwidth reservation, and is nearly a 330 percent improvement over that achieved by the CAC scheme using peak bandwidth reservation.

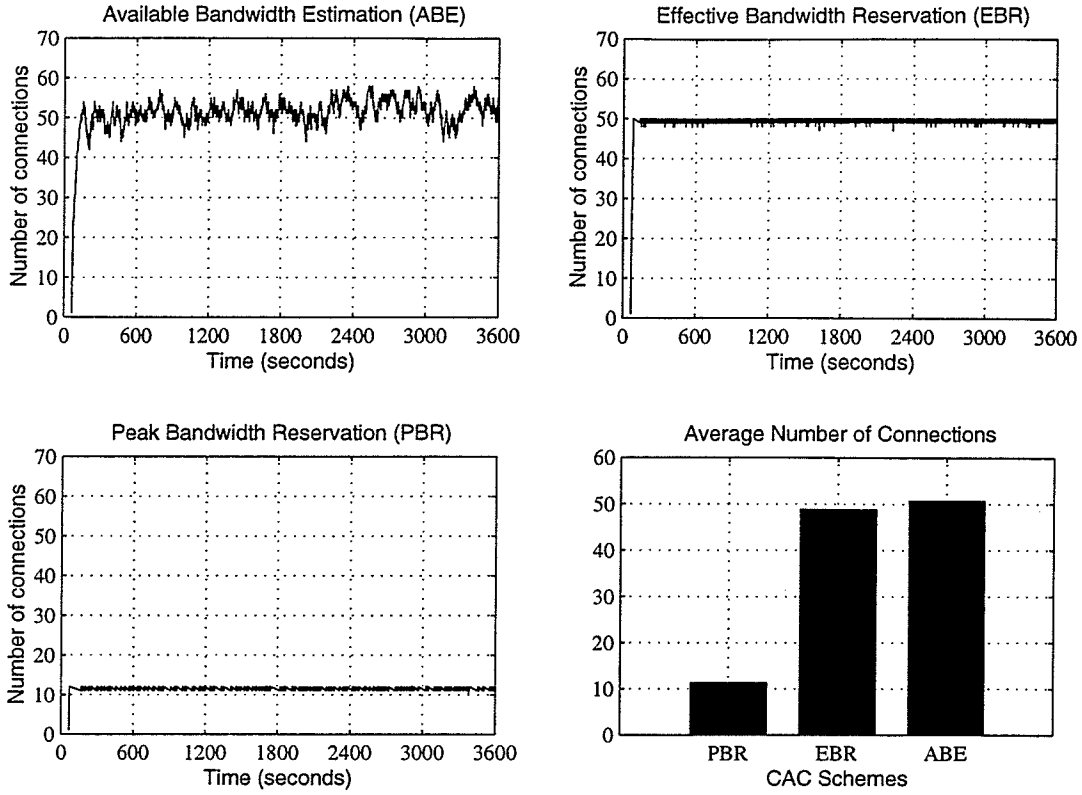

Figure 6-11. CAC scheme comparison under the traffic arrival process of multiplexed self-similar traffic sources

	CAC scheme		
	ABE	EBR	PBR
Maximum number of accepted connections	58	50	12
Average number of accepted connections	50.72	48.84	11.36
Number of cells received	9,546,056	9,254,628	2,202,851
Number of cells discarded	0	0	0
Cell loss ratio	0	0	0
Output link utilization	70.27%	68.12%	16.22%

Table 6-9. Simulation results of the CAC scheme comparison under the traffic arrival process of multiplexed self-similar traffic sources

6.4 SUMMARY

In this chapter, performance evaluations of the new CAC scheme are conducted. It was found that the length of each available bandwidth update time interval is the only parameter vital for the new CAC scheme to achieve required CLR QoS. It was also found that buffer size has very little effect on the performance of the new CAC scheme. Therefore, small or medium sized output buffer can be chosen to limit the maximum CTD and peak to peak CDV without decreasing the output link utilization considerably. The new CAC scheme can still achieve a satisfying efficiency when users over-specify the PCR of their calls or when PCR is the only traffic parameter specified by users. The new CAC scheme is further compared to two other CAC schemes, peak bandwidth reservation and effective bandwidth reservation, under five types of traffic arrival processes. Simulation results show that the CAC scheme outperforms the two other CAC schemes under all scenarios, with a significant improvement of the output link utilization by 100-200 percent in most cases. Furthermore, CLR QoS requirement is always well maintained by the new CAC scheme.

CHAPTER SEVEN

CONCLUSIONS

7.1 CONTRIBUTIONS

The contributions of this thesis are summarized as follows:

1. A generalized traditional traffic generator and a self-similar traffic generator are implemented as process models in OPNET. The generalized traditional traffic generator includes the deterministic ON/OFF, the two state MMFP, the IPP, and the two state MMPP traffic models as special cases. The self-similar traffic generator is based on multiplexing a number of ON/OFF sources with heavy-tailed ON/OFF periods. Both traffic generators are able to produce realistic ATM traffic at a very high speed of 25 millions cells per hour. They can be configured to generate packets of various formats as well. They are also capable of setting up and releasing calls as well as specifying the traffic parameters of each call. They can be used directly for the

performance evaluations of ATM traffic control schemes. Properties of the synthetic traffic traces generated by both traffic generators, such as PCR, SCR, cell (or packet) count plots at various time scales, variance-time curve, and Hurst parameter, can be estimated, produced, or calculated using many auxiliary OPNET process models, C++ programs and MATLAB routines. The relations between the model attributes of the self-similar traffic generator and the Hurst parameters of the synthetic self-similar traffic traces produced by the generator are also studied. The results can be used for matching the self-similar traffic collected from real networks in terms of the SCR and the Hurst parameter.

2. A comprehensive literature review on existing CAC schemes has been carried out. The advantages and disadvantages of each of them have been identified. It was found that all CAC schemes available in the literature have one or more of the following shortcomings: i) having assumptions on the type of traffic sources, ii) using the output link inefficiently, iii) being time consuming thus not able to operate in real time, and iv) being inflexible to support new types of traffic.
3. The approach of designing CAC schemes based on estimating the available bandwidth of the output link is explored. A new CAC scheme based on this approach is developed. No specific traffic types are assumed and the new CAC scheme is able to operate in real-time. A mandatory traffic parameter, PCR, and an optional traffic parameter, SCR, are required for each call. The new CAC scheme and two other CAC schemes, peak bandwidth reservation and effective bandwidth reservation, are also implemented as process models in OPNET.
4. Performance evaluations of the new CAC scheme are conducted. It was found that the

length of each available bandwidth update time interval is the only parameter vital for the new CAC scheme to achieve required CLR QoS. It was also found that the buffer size has no significant effect on the performance of the new CAC scheme. Therefore, small or medium sized output buffer can be chosen to limit the maximum CTD and peak to peak CDV without decreasing the output link utilization considerably. The new CAC scheme can still achieve a satisfying efficiency when users over-specify the PCR of their calls. The new CAC scheme is further compared to two other CAC schemes, peak bandwidth reservation and effective bandwidth reservation, under five types of traffic arrival processes. Simulation results show that the new CAC scheme outperforms the two others under all scenarios, with a significant improvement of the output link utilization by 100-200 percent in most cases. Furthermore, CLR QoS requirements are always well maintained by the new CAC scheme.

7.2 RECOMMENDATIONS FOR FUTURE WORK

The following issues can be further studied in the future:

1. Both traffic generators produce cells (or packets) in continuous time rather than in discrete time. This means that cells (or packets) can arrive at any time instead of at the beginning of a time slot. This simplification has no effect on the accuracy of the CAC scheme performance evaluations and comparisons, and it is therefore not implemented. A good extension to our traffic generators is to provide the flexibility to produce cells (or packets) in both continuous time and discrete time.
2. No formula is given for the choice of the length of each available bandwidth update time interval. Based on experience, the length of each available bandwidth update time

interval can be set as 10-30 times of the average burst length or 1-2 times of the average length of an ON/OFF cycle of some typical ATM services. A more dynamic approach is to set the length of the available bandwidth update time interval to an initial value and adjust it every few minutes according to the output link utilization and the observed CLR. For example, if for the last few minutes, the output link utilization is only 40 percent and the observed CLR is less than the required CLR QoS, the length of each available bandwidth update time interval may be decreased. If for the last few minutes, the output link utilization is 70 percent and the observed CLR is larger than the required CLR QoS, the length of each available bandwidth update time interval should be increased. Formulas or procedures for choosing the length of each available bandwidth update interval are a topic for future research.

3. In the CAC scheme performance evaluations and comparisons, no UPC is performed on the traffic of each accepted call. Therefore, the effects of UPC on the traffic, the QoS of users' calls, and the performance of the CAC schemes are not studied. Those impacts could be an interesting topic to be investigated.
4. In the CAC scheme performance evaluations and comparisons, traffic sources are assumed to have the same CLR QoS requirement and the maximum cell delay is satisfied by engineering the buffer size. In reality, different ATM services may have different CLR QoS requirements. In this case, different cell loss priorities can be assigned to each of those services. At the time of congestion, cells with low cell loss priorities are discarded first to protect cells of those connections with stricter CLR QoS requirements. Also, buffer size can be set to be larger than that can guarantee the cell delay requirements of real-time services. In this case, different cell delay priorities

can be assigned to each of those services and scheduling mechanisms are needed to make sure that cells from real-time services do not experience great cell delay. A good extension of our work could be adding those features to the OPNET process models we developed for both the traffic generators and the CAC schemes.

5. Only CBR and VBR services are supported by our traffic generators and CAC scheme implementations. However, ABR and UBR services can be supported by adding extra functions to our traffic generators and CAC scheme implementations. Different services can share the same network resources or each type of service can have its own dedicated network resource. The impact of ABR and UBR services on the congestion control, the complexity and the performance of a CAC scheme, and the successful operation of an ATM network is also an interesting topic to study.

REFERENCES

[ABE94] S. Abe and T. Soumiya, "A traffic control method for service quality assurance in an ATM network ", *IEEE J. Select. Areas Commun.*, vol. 12, no. 2, pp. 322-331, February 1994.

[ATMForumTM] The ATM Forum, Traffic Management Specification, Version 4.0, April 1996.

[AXNE97] D. Axner, "ATM carrier services: a progress report", *Telecommunications*, pp. 32-38.

[BERA95] J. Beran, R. Sherman, M. S. Taqqu, W. Willinger, "Long-range dependence in variable-bit-rate video traffic", *IEEE Trans. Commun.*, vol. 43, no. 4, pp. 1566-1579, April 1995.

[CHAN95] C. S. Chang, J. A. Thomas, "Effective bandwidth in high-speed digital networks", *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1091-1100, August 1995.

[CHEN96] T. M. Chen, etc. "The Available Bit Rate Service for Data in ATM Networks", *IEEE Commun. Mag.*, vol. 34, no. 5, pp. 56-71, May 1996.

[CHOU96] G. L. Choudhury, D. M. Lucantoni, and W. Whitt, "Squeezing the most out of ATM", *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 203-217, February 1996.

[ELWA93] A. Elwalid and D. Mitra, "Effective bandwidth of general Markovian traffic sources and admission control of high speed networks", *IEEE Trans. Networking*, vol. 1, no. 3, pp. 329-343, June 1993.

[ELWA95] A. Elwalid, D. Mitra, and R. H. Wenworth, "A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node", *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1115-1127, August 1995.

[FAN96] Z. Fan and P. Mars, "Effective bandwidth in ATM netwoks", *Electronics Letters*, vol. 32, no. 16, pp. 1438-1439, 1st August 1996.

[FREE92] Freeman and D. M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison Wesley, 1992.

[GARR94] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic", *Proc. ACM Sigcom*, London, UK, pp. 269-280, 1994.

[GIBB95] R. J. Gibbens, F. P. Kelly, and P. B. Key, "A decision-theoretic approach to call admission control in ATM networks", *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1101-1114, August 1995.

[GUER91] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high speed networks", *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 968-981, 1991.

[HEFF86] H. Heffes and D. M. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance", *IEEE J. Select. Areas Commun.*, vol. 4, pp. 856-868, 1986.

[HIRA94] A. Hiramatsu, "ATM call admission control using a neural network trained with a virtual output buffer method", *Proc. IEEE Int'l Conf. on Neural Networks '94*, Orlando, Florida, pp. 3611-3616, 1994.

[HIRA95] A. Hiramatsu, "Training techniques for neural network applications in ATM", *IEEE Commun. Mag.*, pp. 58-67, October 1995.

[HUI88] J. Y. Hui, "Resource allocation for broadband networks", *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1115-1127, December 1988.

[KESI93] G. Kesidis, J. Walrand, and C. S. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources", *IEEE Trans. Networking*, vol. 1, no. 4 , pp. 424-428, August 1993.

[KEY95] P. B. Key, "Connection admission control in ATM networks", *B. T. Tech. J.*, vol. 13, no. 3, July 1995.

[KUCZ73] A. Kuczura, "The interdeparture Poisson process as an overflow process", *Bell. Syst. Tech. J.*, vol. 52, pp. 437-448, 1973.

[KULK95] V. G. Kulkarni, L. Gün, and P. F. Chimento, "Effective bandwidth vectors for multiclass traffic multiplexed in a partitioned buffer", *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1039-1047, August 1995.

[LAU95] W. Lau, A. Erramilli, J. Wang, and W. Willinger, "Self-similar traffic generation: The random midpoint displacement algorithm and its properties", *Proc. IEEE Int. Conf. Commun.* , pp. , June 1995.

[LEE96] T. H. Lee, K. C. Lai, and S. T. Duann, "Design of a real-time call admission controller for ATM networks", *IEEE Trans. Networking*, vol. 4, no. 5, pp. 758-765, October 1996.

[LELA94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)", *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1-15, February 1994.

[MAGL88] B. Maglaris, D. Anaastassiou, P. Sen, G. Karlsson, and J. D. Robbins, "Performance models of statistical multiplexing in packet video communications", *IEEE Trans. Commun.*, vol. 36, no. 7, pp. 834-844, July 1988.

[MILL96] M. A. Miller, *Analyzing Broadband Network --- ISDN, Frame Relay, SMDS, & ATM*, Second Edition, M&T Books, 1996.

[MURA91] T. Murase, H. Suzuki, S. Sato, and T. Takeuchi, "A call admission control scheme for ATM networks using a simple quality estimate", *IEEE J. Select. Areas Commun.*, vol. 9, no. 9, pp. 1461-1470, December 1991.

[OPNET1] Modeling Framework, OPNET Modeling Manual, Volume 1, MIL 3 Inc. OPNET Doc. Viewer, Release 3.0.A.

[OPNET2] Modeling Overview, OPNET Modeling Manual, Volume 1, MIL 3 Inc. OPNET Doc. Viewer, Release 3.0.A.

[OPNET3] OPNET Simulation Kernel Manual, Volume 1, MIL 3 Inc. OPNET Doc. Viewer, Release 3.0.A.

[RASM91] C. Rasmussen *et al.*, "Source independent call acceptance procedures in ATM networks", *IEEE J. Select. Areas Commun.*, vol. 9, pp. 351-358, April, 1991.

[PATH91] E. Rathgeb, "Modeling and performance comparison of policing mechanisms for ATM networks", *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 982-989, April, 1991.

[PAXS95a] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling", *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 226-244, June 1995.

[PAXS95b] V. Paxson, "Fast approximation of self-similar network traffic", *Report* LBL-36750, Lawrence Berkeley Laboratory, Univ. of California at Berkeley, 1995.

[PERR96] H. G. Perros and K. M. Elsayed, "Call admission control schemes: a review", *IEEE Commun. Mag.*, pp. 82-90, November 1996.

[PRUT95] P. Pruthi and A. Erramilli, "Heavy-tailed on/off source behavior and self-similar traffic", *Proc. IEEE Int. Conf. Commun*, pp. 18-22, June 1995.

[PRYC95] M. D. Prycker, *Asynchronous Transfer Mode --- Solution for Broadband ISDN*, Third Edition, Prentice Hall PTR, 1995.

[RUED96] A. Rueda and W. Kinsner, "A survey of traffic characterization techniques in telecommunication networks", *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, vol. II, pp. 830-833, Calgary, Canada 1996.

[SAIT91] H. Saito and K. Shiomoto, "Dynamic call admission control in ATM networks", *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 982-989, September 1991.

[SAIT92] H. Saito, "Call admission control in an ATM network using upper bound of cell loss probability ", *IEEE Trans. Commun.*, vol. 40, no. 9, pp. 1512-1521, September 1992.

[SAIT94] H. Saito, *Teletraffic Technologies in ATM Networks*, Artech House, 1994.

[SHIM94] C. Shim, I. Ryoo, J. Lee, and S. B. Lee, "Modeling and call admission control algorithm of variable bit rate video in ATM networks", *IEEE J. Select. Areas Commun.*, vol. 12, no. 2, pp. 332-344, February 1994.

[STAL97] W. Stallings, *Data and Computer Communications*, Fifth Edition, Prentice Hall PTR, 1997.

[TANE96] A. S. Tanenbaum, *Computer Networks*, Third Edition, Prentice Hall PTR, 1996.

[TRAN92] P. Tran- Gia and O. Gropp, "Performance of a neural net used as admission controller in ATM systems ", *IEEE INFOCOM*, pp. 1303-1309, 1992.

[TSYB97] B. Tsybakov and N. D. Georganas, "On self-similar traffic in ATM queues: definitions, overflow probability bound, and cell delay distribution", *IEEE/ACM Trans. Networking*, vol. 5, no. 3, pp. 397-409, June 1997.

[UEHA97] K. Uehara and K. Hirota, "Fuzzy connection admission control for ATM networks based on possibility distribution of cell loss ratio", *IEEE J. Select. Areas Commun.*, vol. 15, no. 2, pp. 179-190, February 1997.

[WILL97] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high variability: statistical analysis of ethernet LAN traffic at the source level", *IEEE/ACM Trans. Networking*, vol. 5, no. 1, pp. 71-86, February 1997.

[YANG95] T. Yang and D. H. K. Tsang , "A novel approach to estimating the cell loss probability in an ATM multiplexer loaded with homogeneous on-off sources", *IEEE Trans. Commun.*, vol. 43, no. 1, pp. 117-126, January 1995.

[YOUS97] S. A. Youssef, I. W. Habib, and T. N. Saadawi, "A neurocomputing controller for bandwidth allocation in ATM networks", *IEEE J. Select. Areas Commun.*, vol. 15, no. 2, pp. 191-199, February 1997.

[ZHU96] H. Zhu and V. S. Frost, "In-service monitoring for cell loss quality of service violations in ATM networks", *IEEE Trans. Networking*, vol. 4, no. 2 , pp. 240-248, April 1996.