A hybrid queueing model for fast broadband networking simulation

Liu, Enjie

A Hybrid Queueing Model for Fast Broadband Networking Simulation

By

Enjie Liu

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Department of Electronic Engineering

Queen Mary, University of London

March 2002

To my parents

Abstract

This research focuses on the investigation of a fast simulation method for broadband telecommunication networks, such as ATM networks and IP networks. As a result of this research, a hybrid simulation model is proposed, which combines the analytical modelling and event-driven simulation modelling to speeding up the overall simulation.

The division between foreground and background traffic and the way of dealing with these different types of traffic to achieve improvement in simulation time is the major contribution reported in this thesis. Background traffic is present to ensure that proper buffering behaviour is included during the course of the simulation experiments, but only the foreground traffic of interest is simulated, unlike traditional simulation techniques. Foreground and background traffic are dealt with in a different way.

To avoid the need for extra events on the event list, and the processing overhead, associated with the background traffic, the novel technique investigated in this research is to remove the background traffic completely, adjusting the service time of the queues for the background traffic to compensate (in most cases, the service time for the foreground traffic will increase). By removing the background traffic from the event-driven simulator the number of cell processing events dealt with is reduced drastically.

Validation of this approach shows that, overall, the method works well, but the simulation using this method does have some differences compared with experimental results on a testbed. The reason for this is mainly because of the assumptions behind the analytical model that make the modelling tractable.

Hence, the analytical model needs to be adjusted. This is done by having a neural network trained to learn the relationship between the input traffic parameters and the output difference between the proposed model and the testbed. Following this

training, simulations can be run using the output of the neural network to adjust the analytical model for those particular traffic conditions.

The approach is applied to cell scale and burst scale queueing to simulate an ATM switch, and it is also used to simulate an IP router. In all the applications, the method ensures a fast simulation as well as an accurate result.

Acknowledgements

During the course of my research, I have been supported by a Teaching Company Associated (TCS) project, which was co-operative research between the Teaching Company Directorate, Nortel Networks, and Queen Mary, University of London.

I would like to thank my supervisor, Professor Laurie Cuthbert, and Dr. John Schormans, for the guidance and support provided throughput the course of this research, for the experiment facilities provided during the practical work, and for encouragement whenever I made a progress. I am deeply grateful to them.

I also would like to thank Mr. Gary Stoneley for his support and help during the two years that I spent in Nortel and to Miss Celine Ganne for her help at the beginning of this research. Thanks to Caroline and Dave for their help in the Router Lab tests.

Many thanks go to the staff of the Electronic Engineering Department, Prof. Jonathan Pitts, Dr. Chris Phillips, Mrs. Lynda Rolfe, Andy Martin, Ho, Husam, Arif, Rob, Veselin, and many others.

Finally, my love and gratitude go to my parents, husband and my daughter, for their endless support and encouragement.

Table of Contents

List of Figures

List of Tables

Glossary

ANN	Artificial Neural Network
ATM	Asynchronous Transfer Model
BBS model	a detailed ATM switch model
B-ISDN	Broadband Integrated Service Digital Network
BP	Back-Propagation
CAC	Connection Admission Control
CCITT	Consultative Committee on International Telegraphy
CDF	Cumulative Density Function
CDMA	Code Divided Multi-Address
CLP	Cell Loss Probability
CLR	Cell Loss Rate
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DPR	Direct Probability Re-distribution
DR	Decay Rate
ER	Excess Rate
FBM	Fractional Brownian Motion
FIFO	First In First Out
HOL	Head of Line
HQ	Hybrid Queueing
IET	Inter Exit Time
IN	Interconnection Network
IP	Internet Protocol
IPC	Input Port Controller
IS	Importance Sampling
LAN	Local Area Network
LP	Logical Processor
LRD	Long Range Dependent
MAC	Media Access Control
MC	Monte Carol

M/D/1	Queueing model: Poisson arrival, fixed service time, and one server
M/G/1	Queueing model: Poisson arrival, Generic service time, and one server
M/M/1	Queueing model: Poisson arrival, Exponential service time, one server
MSE	Minimum Square Error
NIC	Network Interface Card
NN	Neural Networks
OH	Occurrence Histogram
OPC	Output Port Controller
OPNET	Optimised Network Engineering Tool
OSPF	Open Shortest Path First protocol
QoS	Quality of Service
RIP	Routing Information Protocol
RMS	Root Mean Square
RRL	Research Router Lab
SE	Switching Element
SRD	Short Range Dependent
TCS	Teaching Company Scheme
TCP	Transport Control Protocol
VBR	Variable Bit Rate
VC	Virtual Channel
VCI	Virtual Channel Identifier
VP	Virtual Path
VPI	Virtual Path Identifier

1. Introduction

1.1 Motivation behind this research

Evaluating the performance of modern telecommunications networks is desirable for telecommunication service providers and telecommunication product manufacturers alike. Such evaluation ensures that service providers understand the utilisation of their infrastructure networks and so can provide satisfactory services to customers. It also provides the manufacturers with information on the requirements and performance of their equipment in networks.

Measurement, mathematical analysis and system simulation are three general techniques for network modelling, performance evaluation and network design. They are complementary and each approach has its place in the design life cycle of a communication network. However, in many cases (such as a new broadband network) measurement is impossible and mathematical analysis intractable; in such cases simulation is the only viable modelling approach.

Simulation modelling of an overall network generally is based upon modelling of a number of elemental network nodes (for instance, ATM switches in ATM networks). The level of detail at which each node is modelled has an enormous impact on (i) the level of detail and accuracy of the overall simulation and (ii) the length of time the simulation takes to run. For ATM networks, modelling has generally been undertaken at *cell level* and *burst level*. [Pitts95] is a good example of how simulation time can be drastically reduced by using burst level. Burst level simulation can significantly reduce the number of events the simulator has to generate to achieve a certain accuracy, compared with cell level simulation.

The problem, however, with burst level simulation is that it still requires a large number of events when a complete network is modelled. The approach to be described in this thesis overcomes this problem by modelling only the traffic of interest and changing the queuing behaviour of the model in order to allow for the

effect of the other traffic. This results in a very large increase in simulation speed, yet at the same time allows features of network components to be included that could only previously be modelled by detailed cell-scale models. The approach is termed "Hybrid Queuing Simulation", or for brevity "HQ Simulation".

This research was initiated in a TCS project between Nortel Networks and Queen Mary, but the research reported in this thesis has been done solely by the author. The joint project allowed access to Nortel test facilities to validate the simulator against network hardware.

1.2 Scope of this research

The research reported here uses high-level abstracted analytical models, based on prior intensive investigation of packet queuing behaviour, to substantially speed up the simulation of ATM networks without losing the accuracy in the same way that other fast simulation techniques do. The accuracy is maintained by using a neural network to learn the adjustments required to the queuing model in order to maintain the vital features in real components. The vision is that different queuing models, with different parameters set by the neural network, could be used to model different elements (perhaps from different manufacturers) within the network.

This approach is particularly applicable in the area of large-scale network design and planning where concern is more about the overall performance of the network than the detailed structure of a network node, although the features of different types of node must still be included.

1.3 Novelty and contribution of this research

In the simulation modelling described in this thesis, the foreground and background traffic are treated in a novel way, completely different from those existing techniques reported in the literature. Moreover, the application of a neural network to refine the model parameters is also new. In more detail:

1.3.1 Novelty

This is the first effort to investigate traffic modelling by simulating the foreground traffic only, and mathematically modelling how the background traffic affects the foreground traffic. The method was also applied in a priority queue environment, which is aimed at addressing the QoS in a telecom network.

It is the first attempt to apply a neural network to re-align a hybrid model to further enhances accuracy. The neural network can also learn to compensate for the difference between the hardware testbed and the simulation model.

1.3.2 Specific Contributions

A novel approach has been proposed in speeding up the simulation. This approach is not only applicable to simulating rare events, but it can also be used as a general fast simulation method.

The approach has been used to model both ATM and IP traffic and the models have been validated against hardware or more detailed (and slower) simulations.

Decay rate of a buffer was used in parameterisation of the service rate for foreground cells/packets in considering the affect of the background traffic.

Several comparison model-pairs have been established. The idea and the topology of these establishments can also be used for other purposes of simulation.

1.4 Layout of this thesis

The thesis comprises three parts: background of the research, description of the proposed method and the applications.

In chapter 2, some basic concepts of the simulation method are addressed. Since this research is about an accelerated simulation method, some existing fast simulation

methods in the literature are also summarised here. In particular, cell-rate simulation and importance sampling are addressed in detail.

Chapter 3 describes the basic structure of an ATM switch, especially how to model an ATM switch. The queuing models that have been used in this research are presented. Finally, the basic idea of this research is given and the speed-up factor is calculated by a practical example at the end of the chapter.

The HQ simulation approach comprises an analytical model and a simulation model. Derivation of the analytical model is based on prior intensive investigation of packet queuing behaviour and its high-level abstraction enables the hybrid model to speed up the simulation substantially. However, the "cost" of this acceleration is that the accelerated model is not accurate enough in some cases, and hence generates some modelling errors when validated against a testbed. This modelling error can be eliminated by embedding a neural network in the HQ model. The embedded neural network is capable of representing the complex non-linear relation between the results from the HQ model and the testbed by self-learning. Hence, the final adjusted model is *fast* thanks to the analytical model and *accurate* thanks to the embedded neural network. Chapter 4 illustrates the application of the neural networks.

ATM switch simulation can be classified into cell-scale and burst-scale simulation. Chapter 5 and 6 address the application of the HQ simulation approach at the cell-scale and burst-scale respectively, and presents the derivation of the service rate in each case. The simulation results are compared with those from a detailed simulation that simulates both foreground and background traffic, and hence acts as a "software testbed". This comparison shows that there is some difference between the HQ simulation and the software testbed, and hence a neural network is applied to learn the relation between the input parameters and the output difference. Upon completing the self-learning, the neural network is then used to 'fine-tune' the HQ model accordingly. New simulation results indicate that the HQ simulator that consists of

analytical model and neural network achieves fast simulation while remaining accurate.

Given the growing impact of TCP/IP based traffic on current telecommunication networks, routers are becoming more and more popular in Local Area Networks (LANs). In Chapter 7, the HQ approach is applied in simulating a router. A HQ based simulation model is built, and used as a building block in a network simulation. The testbed used here is a "real" (i.e. hardware) router.

Finally, discussions and summary of this work are given in Chapter 8. In addition, some of the possible ways to continue this research are also addressed.

2. Simulating telecommunication networks

This chapter discusses some basic concepts of simulation, especially those important to this particular research. Some fast simulation methods published in the literature such as parallel simulation, Importance Sampling (IS) and cell-rate simulation are addressed in detail.

2.1 Network performance analysis techniques

The measures of interest of network performance depend on the type of network being analysed, the services carried and whether the network is circuit-switched or packet-switched. In general, the important performance measures for circuit-switched networks are *probability of blocking* (of calls, i.e. grade of service) and *maximum connection attempts*, while examples for packet-switched networks are *throughput* (number of messages transmitted per unit time), *delay*, *delay variation*, *packet loss probability*, *buffer occupancy statistics*, and *link utilisation*.

Three general approaches towards network modelling, analysis, and design can be identified as follows:

- Measurement
- Mathematical analysis
- Simulation

Measurement provides the most direct means of network performance evaluation. It is also the most expensive in the sense that it has to be on an existing network or on a prototype. In addition, if experimentation is required, it may not be economically feasible to suspend the operation of an ongoing network in order to perform these tests, and the tests may take far too long to produce results in time for them to be of any use.

Mathematical analysis models require a high degree of abstraction; considerable effort and skill may be required on the part of the network modeller to develop a

performance model that accurately reflects the system under study. The analytic model itself, however, can generally be solved rather quickly. However a tractable analytic model often restricts the range of system characteristics that can be explicitly considered in a performance model. For example, systems with complex job scheduling policies, finite buffers that cause blocking, simultaneous resource possession by a job, complex timing constraints, or tightly coupled interactions between various parts of the model are in general quite difficult to solve analytically. As a result, a modeller may end up with a correct solution, but to the wrong problem. Nevertheless, analytical techniques can be effective when carefully applied in practice and can be extremely useful for efficiently and broadly exploring a network's design space.

With *simulation* techniques, a network can be modelled to an arbitrary level of detail. Simulation thus typically serves to examine portions of the design space in more detail. Since the system may be modelled to any arbitrary degree of detail, less abstraction is required and the process of model parameterisation is a more straightforward task (although it remains advantageous from a solution standpoint to abstract out as many secondary system details as possible). The solution of a simulation model requires significantly more computer time than an analytical model.

In some cases, simulation is the only viable modelling approach. [Kurose88] gave an example of how often people tend to use simulation instead of other approaches. Statistics show that among users of a particular modelling package, RESQ [Sauer83], which can be used to develop and solve either analytic or simulation models of a communication system, over 99% of all models developed use simulation. In modern ATM or TCP/IP networks, because of the size and the complexity of the network, it is very difficult, or even impossible, to investigate performance using analytic methods. As a result, simulation plays an ever more important role in performance analysis for both switch/router vendors and service providers.

2.2 Simulation techniques

There have been several impediments to even wider acceptance and usefulness of simulation. First, models used to study large-scale systems tend to be very complex, and writing computer programs to execute them can be an arduous task, although this has been made much easier in recent years by the development of software products that automatically provide many of the features needed to program a simulation model, such as OPNET[1] [Mil3]. A second, and perhaps more important, aspect with the simulation of complex systems is that a large amount of computer time is often required. Although it might be thought that this issue is becoming much less of a problem as computers become faster, the networks that are being modelled are even more complex.

Simulation uses a computer program to imitate the operations of various kinds of real-world facilities or processes. The facility or process of interest is usually called a *system*, and in order to implement a simulation, it is usually necessary to make a set of assumptions about how it works. These assumptions, which usually take the form of mathematical or logical relationships, constitute a *model* that is used to try to gain some understanding of how the corresponding system behaves. The simulation evaluates a model *numerically*, and data are gathered in order to *estimate* the desired true characteristics of the model.

2.3 Comparison of simulation models

It is useful to classify simulation models (for simulation in general, not just for applications in the telecommunications area) along three different dimensions as in [Law99]. These are (i) *static* vs. *dynamic simulation models*, (ii) *deterministic* vs. *stochastic* simulation models and (iii) *continuous* vs. *discrete* simulation models.

[1] OPNET™ is a commercial simulation tool with established libraries

2.3.1 Static vs. dynamic simulation models

A *static* simulation model is a representation of a system at a particular time, or one that may be used to represent a system in which time simply plays no role; examples of static simulation are "Monte Carlo" models. On the other hand, a *dynamic* simulation model represents a system as it evolves over time.

2.3.2 Deterministic vs. stochastic simulation models

If a simulation model does not contain any probabilistic (i.e. random) components, it is called *deterministic*. In deterministic models, the output is "determined" once the set of input quantities and relationships in the model have been specified, even though it might take a lot of computer time to evaluate what it is. Many systems, however, must be modelled as having at least some random input components, and these give rise to *stochastic* simulation models. Stochastic simulation models produce output that is itself random, and must therefore be treated as only an estimate of the true characteristics of the model; this is one of the main disadvantages of stochastic simulation.

2.3.3 Continuous vs. discrete simulation models

Continuous simulation is about the modelling over time of a system by a representation in which the state variables change continuously with respect to time. Typically, continuous simulation models involve differential equations that give relationships for the rates of change of the state variables with time.

Discrete event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate point in time. These points in time are the ones at which an *event* occurs, where an event is defined as an instantaneous occurrence that may change the state of the system. Although discrete-event simulation could conceptually be done by hand calculations, the amount of data that must be stored and manipulated for most real-world systems dictates that discrete-event simulations be done on a computer.

A discrete event-driven simulation model is used in this research because, packet arrivals are discrete events, and hence the remainder of this chapter will concentrate on this type of simulation.

2.4 Discrete event-driven simulation

2.4.1 Application area

The simulation technique, typically used to study a model of a communications network or protocol, is a *stochastic discrete event simulation*. In discrete event simulation, various components of the actual network under study (for example, the communication links, buffers and access strategies, network control structures) are represented within a computer program. The events that would occur during the actual operation of the network (for example the arrival, transmission, routing, and departure of messages as well as error conditions such as message loss or corruption, link/node failure and recovery) are then mimicked during the execution of the program. The function of the simulation is thus simply to generate events and the network's response. The simulation program typically also performs other ancillary tasks such as recording and, later, analysing output data as well.

A lot of work has been done on discrete event-driven simulation for network performance modelling, analysis, and design of computer networks. [Sauer83] [Schoemaker82] [Computer82] are just a few examples.

2.4.2 Common components in discrete event-driven simulation

Although simulation has been applied to a wide range of real-world systems, discrete-event simulation models all share a number of common components and there is a logical organisation for these components that promotes the programming, debugging, and future changing of a simulation model's computer program. In particular, the following components will be found in most discrete-event simulation models using the next-event time-advance approach programmed in a general-purpose language.

- *System state*: the collection of state variables necessary to describe the system at a particular time.
- *Simulation clock*: a variable giving the current value of simulated time.
- *Event list*: a list containing the next time when each type of event will occur.
- *Statistical counters*: variables used for storing statistical information about system performance.
- *Initialisation routine*: a sub-program to initialise the simulation model at time zero.
- *Timing routine*: a sub-program that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.
- *Event routine*: a sub-program that updates the system state when a particular type of event occurs (there is one event routine for each event type).
- *Library routine*: a set of sub-programs used to generate random observations from probability distributions that were determined as part of the simulation model.
- *Report generator*: a sub-program that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends.

2.4.3 Simulation clock

Because of the dynamic nature of discrete-event simulation models, it is necessary to keep track of the current value of simulated time as the simulation proceeds, and also to have a mechanism to advance simulated time from one value to another. The variable in a simulation model that gives the current value of simulated time is called the *simulation clock*. The unit of time for the simulation clock is never stated explicitly when a model is written in a general-purpose language, and it is assumed to be in the same units as the input parameters. Also, there is generally no relationship between simulated time and the time needed to run a simulation on the computer. When speedup factor is mentioned in this thesis, it refers to comparisons between the actual times taken to run the simulation, of which the simulated times are the same.

Historically, two principal approaches have been suggested for advancing the simulation clock: *next-event time* advance and *fixed-increment time* advance. The first approach is used by all major simulation software (and by most authors coding their own model in a general-purpose language) because in most real systems events do not occur at fixed time intervals. Hence, the next-event time advance approach will be used for all discrete-event simulation models discussed in this thesis.

With the next-event time-advance approach, the simulation clock is initialised to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the most imminent of these future events, at which point the state of the system is updated to account for the fact that an event has occurred, and the knowledge of the times at which future events occur is also updated. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc. This process of advancing the simulation clock from one event to another is continued until eventually some pre-specified stopping condition is satisfied. Since all state changes occur only at event times for a discrete event simulation model, periods of inactivity are skipped over by jumping the clock from one event time to the next. (Fixed-increment time-advance does not skip over these inactive periods, and hence could waste a lot of computer time.) It should be noted that the successive jumps of the simulation clock are generally variable in size.

2.4.4 Random number generation

A simulation of any system or process in which there are inherently random components requires a method of generating or obtaining numbers that are *random*, in some sense. In this thesis, the interarrival time of successive cells, service time of a cell, etc, are all ‘drawn’ from some specified distributions, such as an exponential distribution. The numbers produced by a random number generator should appear to be distributed uniformly on [0,1] and should not exhibit any correlation with each other; otherwise, the simulation’s results may be completely invalid.

The random number generator used in OPNET uses a single random number sequence initialised with the value of the *seed* environment attribute. The generator used to create this sequence is provided by the host computer's operating system [Mil3].

Sun Solaris™ is the operating system that is used in the simulation in this thesis. The random function in Solaris™ uses a non-linear additive feedback random-number generator employing a default state array size of 31 long integers to return successive pseudo-numbers in the range from 0 to 2^{31}-1. The period of this random number generator is approximately 16 * (2^{31}-1) [Sun].

2.5 Confidence interval

Since the HQ simulation models use random variables as input, the simulation output data are themselves random. "Confidence interval" is a parameter that researchers often use to describe the accuracy of the simulation results. It can be established by extracting range information from gathered samples and giving a likelihood that the "real average" lies somewhere in that range. The term "real average" is used to mean the value that would be obtained if a vast number of samples in the real system was measured and their average computed.

Let $X_1, X_2, \ldots, X_n$ be samples obtained from the random variable under evaluation, which has a finite mean μ and a finite variance σ^2 ($\sigma^2>0$). The sample mean is $\overline{X}(n)$ and the sample variance is $S^2(n)$

$$\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n} \tag{2.1}$$

$$S^2(n) = \frac{\sum_{i=1}^{n} [X_i - \overline{X}(n)]^2}{n-1} \tag{2.2}$$

Define a random variable Z_n

$$Z_n = [\overline{X}(n) - \mu] \Big/ \sqrt{S^2(n)/n} \qquad (2.3)$$

Its distribution function is denoted as $F_n(z)$ for a sample size of n. From the central limit theorem, $F_n(z) \to \Phi(z)$ as $n \to \infty$, where $\Phi(z)$ is the standard normal distribution function, given by:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-y^2/2} dy \qquad -\infty < z < \infty \qquad (2.4)$$

For a sufficiently large n, Z_n is approximately standard normal distributed. Hence,

$$\Pr\left\{ -z_{1-\alpha/2} \le \frac{\overline{X}(n) - \mu}{\sqrt{S^2(n)/n}} \le z_{1-\alpha/2} \right\} \approx 1 - \alpha$$

which means that for a given set of samples $X_1, X_2, \ldots, X_n$, an approximate 100(1- α) percent confidence interval for μ is given by:

$$\overline{X}(n) - z_{1-\alpha/2} \sqrt{\frac{S^2(n)}{n}} < \mu < \overline{X}(n) + z_{1-\alpha/2} \sqrt{\frac{S^2(n)}{n}} \qquad (2.5)$$

The confidence interval for μ in the simulation tests in this thesis is processed by OPNET. The confidence interval is defined [Mil3] to be the interval of real number $[\Theta_L, \Theta_R]$ such that the probability that μ lies within this interval has a particular value α, where $\Theta_L = \overline{X}(n) - z_{1-\alpha/2} \sqrt{\frac{S^2(n)}{n}}$ and $\Theta_R = \overline{X}(n) + z_{1-\alpha/2} \sqrt{\frac{S^2(n)}{n}}$. This interval is referred to as the 100α percent confidence interval for μ. In this thesis, $\alpha = 0.95$ is generally used, which is called the 95% confidence interval for μ.

2.6 Speeding up a simulation

This research is intended to lead to a fast simulation approach, so it is necessary to review existing fast simulation methods first.

As addressed above, a major disadvantage of a simulation, as opposed to analysis, for network performance modelling is that a large amount of time is needed to simulate a model. As network protocols become more sophisticated and complex, so too do their performance models. Moreover, rapidly increasing computation, communication, and switching speeds of communication networks are resulting in ever-increasing message traffic rates. Clearly, simulating the operation of even one real switch in detail for any nontrivial amount of time is a formidable computational task.

This problem is exacerbated in cases where the events of interest (for example buffer overflow) only happen extremely infrequently: i.e. simulation of *rare events*. This means that to get a reasonable number of occurrences of the rare event (in order to get some form of decent statistical measure) an extremely large total number of events have to be simulated, leading to a long simulation time. For example, in an ATM network, the cell loss probability is normally very low, perhaps 10^{-7} or less. ([Bromirski96] quotes 10^{-7} for a TV distribution using real-time VBR traffic). If, for example it is required to generate around 100 of such events, the total number of events to be simulated will be of the order of 10^{9}. The Sun™ Workstation used in this research generates 20,000 events per second; therefore, it takes nearly 14 hours to simulate these events.

To get round this problem, several fast simulation methods have been proposed, of which the *Importance Sampling* method (see 2.6.2.1) is probably the most popular. Another possibility is to use a hybrid method where simulation and analytic method are used together. A simple classification gives three possibilities: (i) *parallel* simulation, (ii) *accelerated* simulation and (iii) *hybrid* simulation.

2.6.1 Parallel simulation method

Several individual computers or processors linked together into *parallel* or *distributed* computing environments, it is then possible to "distribute" different parts of a simulation task across individual processors operating at the same time, or *in parallel*, thus reducing the overall time to complete the task.

Several types of parallel simulation have been proposed, for instance space parallelism [Turner92] [Unger94] and time parallelism [Nikolaidis93] [Nikolaidis94]. Parallel simulation is the process of using multiple processors simultaneously for executing a single simulation, with the goal of reducing the total execution time. It can be performed on commodity machines such as multiprocessor personal workstations and servers, and on workstation clusters.

A parallel discrete event simulation program can be viewed as a collection of interacting sequential simulators, called logical processors (LPs). The computation performed by each LP is a sequence of event computations, where each event represents some interesting action in the model, such as the arrival of a new packet on an input link. Each event contains a timestamp indicating when the event occurs. LPs interact by scheduling events for each other.

There are two commonly used methods for process synchronisation: *conservative* and *optimistic* algorithms. In the *conservative* approach, an LP can only process an event when it has a guarantee that no other event containing a smaller timestamp may arrive at some later time. A possible drawback of this approach is that LPs may be forced to wait unnecessarily until they acquire a guarantee of timestamp-ordered event processing. In contrast, *optimistic* mechanisms do not prevent out-of-order event processing; rather, LPs process events as they arrive without concern for potential causality violation. When an out-of-order event arrives, the LP uses a mechanism called rollback to recover from potential error.

According to [Phillips91], the architecture of the simulator, including decoupling buffering between the various modules, has a significant impact on the performance. For large simulator systems where connectivity is limited and there are many processors, the communication latency can be considerable. The communication infrastructure greatly offsets the performance benefits. Only applications that can be decomposed into units that exhibit a relatively large degree of autonomy are likely to experience performance gains. Of the three schemes examined, namely: Centralized Event-List, Distributed Event-List and Distributed Event-Buffering, only the latter approach yielded worthwhile gains within a transputer environment.

In all three approaches, speedup is obtained by increasing the number of processors, and splitting the network model into domains. In [Wagner89], it is suggested that the potential parallelism and resulting speedup with a network model may be significantly less than anticipated because of the model's characteristics. In the third approach, the relatively large number of components within a domain suggests that the poor performance is because of communication overheads rather than concurrency limitations.

According to [Bhatt98], the feasibility of applying parallel simulation techniques to diverse, large, and complex network models has been recently illustrated in [Nicol98]. Several other good research-oriented parallel network simulators are available, such as [Maisie/PARSEC], [TeleSim].

In [Bhatt98] it was noted that parallel simulation techniques are not yet commonplace in network simulation. The lack of established easy-to-use-modelling methodologies suitable for parallel execution and the absence of mature software environments have so far prevented the widespread use of parallel simulations in network research and industrial practice.

2.6.2 Accelerated simulation

Accelerated simulation uses means of manipulating the simulation in such a way that the results can be determined without having to process so many events.

One way of doing this is to use a technique called *Importance Sampling* that is based on the notion of modifying, or biasing (as it is commonly called) the underlying probabilities in such a way that rare events occur much more frequently (and hence the overall number of events to be simulated is decreased). To correct for this modification, the results are weighted in a way that yields a statistically unbiased estimator.

Another approach is to treat the simulation model at levels where events can be aggregated, for example at connections, bursts of cells (or packets) and calls rather then at individual cells (or packets). For instance, [Pitts95] gives a cell-rate model and [Nikolaidis93] [Nikolaidis94] used a burst-level simulation approach, representing a burst of ATM cells as a simulation event. Similarly, Ahn and Danzig [Ahn96] applied a flow-level traffic representation in TCP/IP networks as well as ATM multiplexer simulations, instead of traditional packet or cell level representations. [Kesidis 96] also explored the feasibility of fluid event-driven simulation for ATM networks.

2.6.2.1 Importance sampling

Importance sampling is widely used to speed-up telecommunications network simulations. IS is a proven method and its essential purpose is to speed-up the simulation in getting statistics for rare events, such as Cell Loss Rate (CLR) in ATM networks.

IS has been used to estimate rare event probabilities in communication networks, [Townsend98] is an overview of the Importance sampling applications in communication networks.

As described in [Townsend98], the IS technique has been widely used to speed up several rare-event simulations with noted success. In each case, an additional parameter, the importance sampling parameter θ, is added to the simulation, and an appropriate value θ is selected to achieve speed up. However, if the variance is to be reduced effectively, the parameter θ must be selected with care, often in a problem-dependent manner.

Key features of Importance sampling can be explained using the following simple example. Assume that the objective is to determine the area of the two-dimensional region B in Figure 2-1. Region B represents a (not so rare) probability of interest. The analytical solution would require a mathematical description of the boundary of region B, as well as a complex integration procedure. In many cases this knowledge is not obtainable; in such cases computer simulation using the Monte Carlo (MC) method is one alternative. The MC method for estimating this area would require generating uniformly distributed random samples over the entire space A. The estimate of the area of Region B would be given by $\hat{B} = N_B/N$, where N_B is the number of hits within Region B, and N is the total number of samples generated. Assuming statistically independent samples, the variance of this estimate is inversely proportional to the number of samples N.

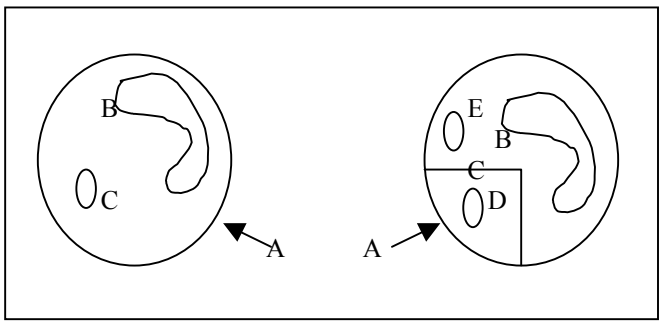


Figure 2-1 Example showing sample space A and regions of interest

In general, the precision of the estimate is related to the number of hits in the important region. Thus, if the objective is to estimate the much smaller area in region C, a much larger number of samples N would have to be generated for an equivalent estimator variance.

Using IS, it would be possible to modify or *bias* the sampling procedure to increase the fraction of samples that result in hits. In the context of this example, this could be done by arbitrarily *doubling* the probability that samples are generated in the quadrant labelled by D. Thus, the average number of samples in region C is doubled, increasing the estimator precision. In this example, each sample that results in a hit in region D must be weighted by a factor ½ to yield the correct, statistically unbiased result.

One of the practical difficulties when developing an IS solution is ensuring that the regions in the space with increased sampling frequency (region D) include the important region (region C). To illustrate this, assume that the region of interest is actually region E in Figure 2-1. Because of insufficient prior knowledge of the system behaviour, samples occur with half the probability in region E with the biasing scheme used here, thus *reducing* the number of hits and the corresponding estimator precision by a factor two. The weight of each hit in region E is 2, rather than ½.

In the context of a communication network, region A represents the state space, sampling represents the evolution of the sample path, and IS in a network is a modification of the stochastic behaviour in a way that increases the probability that the system will visit the preferred region(s).

The IS method is used in solving several problems found in the literature. [Hidelberger98] considered the problem of extremely low packet loss rates in a voice-data multiplexor, via simulation. The multiplexor gives priority to voice packets unless the data queue length exceeds a threshold. The authors claim that to efficiently simulate such low loss rates requires the proper use of importance sampling. They describe an importance-sampling technique that is guaranteed to provide an exponential decrease in variance over standard simulation. This IS strategy has two phases, with a different importance sampling strategy in each phase. Exponential results are presented that demonstrate the effectiveness of this approach.

In [Ben97], the authors used the IS method to research DS/CDMA with and without coding in multipath fading channels. Because of the system complexity, the performance of CDMA systems in such an environment is difficult to access analytically. Monte Carlo simulations can offer an alternative performance evaluation tool. Unfortunately, in practice, such an alternative is often limited by an excessive computational burden. [Ben97] describes the basic principles, and illustrates its use as a computational efficient and accurate tool for computer-aided design and analysis of un-coded as well as coded CDMA in wireless communications.

There are also pitfalls of the IS method. In very complex systems, it is often difficult to determine effective IS methods because of the increasing indirect relationship between the modifiable parameters and the system response, and re-evaluation for changing system topologies involves re-implementing the IS method for each topology. Furthermore, if the relationship between the modifiable parameters and the system response is not properly identified, improper use of IS could result in the phenomenon of underestimation [Devetsikiotis93].

As illustrated in [Akyamac99], *Splitting* is a rare event simulation method that can potentially overcome some of the difficulties associated with IS. Splitting partitions the system states into indexed subnets, where higher indices correspond to the rare events. When a subnet is entered during system simulation, numerous retrials are initiated with the current subnet as the starting state, splitting the main system trajectory into a number of sub-trajectories.

[Akyamac99] presents a technique to implement multidimensional *Splitting*. To obtain efficient and accurate statistical estimates of rare event probabilities, it is necessary to perform the following: (i) to select system parameters as *Splitting* parameters that properly represent the rare event; (ii) to identify which regions of the multidimensional parameter space are closely related to the occurrence of the rare event; and (iii) to set the *Splitting* factors such that the raw steady state probabilities in the important region are equalised. They used a *Splitting* technique called direct

probability re-distribution (DPR) to map multidimensional splitting to a splitting problem with one control parameter. They did the tests in two cases, and both show that improper mapping of the multidimensional parameter space to the subnet indices results in estimates with poor accuracy, and sometimes in underestimation.

[Gallardo99] also proposed a method based on regenerative IS in solving long-range dependent bursty traffic in broadband telecommunication networks. Their work is a generalisation to the model proposed by [Norros95] in the sense that, rather than limiting the marginal distribution of the process to be Gaussian, an alpha-stable distribution is used. The alpha-stable distribution is out of the scope of this thesis; however, the information is given in [Feller66]. The distribution describes the limit behaviour of normalised sums of a relatively large number of independent identically distributed random variables, therefore allowing a better agreement to be achieved between the burstiness of the artificial process and that of the real traffic by selecting the proper stability coefficient.

The speedup factors in different applications mentioned above are different. For example, [Townsend98] claimed in the paper the speedup factors vary between 10^4 and 10^{13}.

In contrast to IS method, the work proposed in this research tries to use a more general method for speeding up a simulation. It is not only used to simulate rare events, but also used to speed up the simulation generally.

2.6.2.2 Cell rate method

[Pitts95] developed the cell rate method specifically for ATM teletraffic studies, taking advantage of the nature of ATM queuing behaviour. It modelled only the burst scale component by manipulating cell *rates* in an alternative model of the queuing mechanism. In this model, an event is a change in the cell rate of a connection. The combined cell rate of all connections is compared with the service rate of the queue to determine whether the queue is increasing or decreasing in size or losing cells. Cell

rate simulation handled many cells per event, rather than one cell per event in the case of the cell-by-cell technique. This reduces the processing time required per simulated cell, though at the expense of algorithmic complexity.

The basic unit of traffic is a 'burst' of cells, described as a cell rate lasting for a particular time period during which the inter-cell time does not vary. The time instant between bursts of different fixed cell rate is an event, i.e. the end of one burst marks the beginning of the next. This differs from cell level modelling, where an event marks a cell arrival or service.

A queue is described by two parameters: the buffer capacity and the cell service rate. The state of the queue at any moment in time is determined by the combination of the input rates from all virtual channels (VCs), the current size of the queue and the queue parameter values. The combined cell rate of all connections is compared with the service rate of the queue to determine whether the queue is increasing, decreasing, or losing cells.

However, the cell-rate method may not work very well when the traffic is not 'bursty', or the edge of the burstiness is not easy to determine. This could happen when there are several traffic streams aggregated together; as consequence, a lot of cells arrive, but it is hard to distinguish the start and end of a 'burst'.

2.6.3 Hybrid simulation models

Generally speaking, a hybrid simulation method could be any combination of methods. For example, it could be an analytical model plus a simulation model, or two different kinds of simulation models.

An advantage of a hybrid simulation model is that it can combine the best features of both methods. In the case of the analytical model plus simulation, the combined model can be faster because of the use of the analytical model; whist still being manageable because simulation is used for the intractable aspects.

[Tunnicliffe00] proposed a hybrid simulation approach (called HYB_SIM) that has a certain similarity to what is done here in this research. They used the method proposed in [Pitts95] cell-rate simulation approach, and made improvements, so that the simulation results are still accurate whilst network traffic is not 'bursty'. The HYB_SIM method combines two different simulation algorithms to produce accurate and efficient ATM-type network simulation. The method contains two separate simulator cores: cell-level simulator and fluid flow simulator. The area of the network of particular interest is partitioned off and subjected to a complete cell-level simulation, while the remainder (those where its traffic is not of interest) is given the fluid-flow treatment only.

The similarity between the above method and HQ method proposed in this thesis is in the way the two methods treat the input traffic. Although [Tunnicliffe00] did not classify input traffic as is done here, the ideas of separation are similar. The authors did not mention how fast the hybrid method is in the paper. Considering the fact that the *traffic that is not of interest* will also be simulated in [Tunnicliffe00], but is analytically modelled in the HQ approach, it can be concluded that the HQ approach is faster.

[Guo00] proposed a time-stepped hybrid simulation method for large-scale networks. The hybrid is in the sense that the method can be used with other simulation methods as well. The authors conducted extensive simulation experiments on fluid networks and TCP networks and obtained a significant speed-up by exploiting parallel simulation technologies.

Time-stepped hybrid simulation splits time into small time intervals (called *time-steps*) that are of fixed length. Packets from the same session falling into the same time-step are grouped into a *chunk* and are assumed to be evenly spaced within the time-step, which the authors denote as *packet smoothing*. The total number of bytes carried by the chunk is maintained as fluid information, and is used to determine the

chunk's service time. The network can be evaluated at different time scales by choosing the time-step. The computational effort is inversely proportional to the length of the time-step.

[Guo00] stated that by carefully choosing the time step, the time-stepped simulation can achieve considerable speedup, without significantly sacrificing simulation accuracy. For example, the simulation with a time-step of 10ms reduces CPU time by more than 99.5% over CPU time of the simulation with time-step 0.1ms.

2.7 Simulation modelling verification and validation

In order to determine whether a network simulation model accurately represents a real system, it needs to be *verified* and *validated*. Verification determines whether the simulation model performs as intended and the validation determines whether the conceptual simulation model is an accurate representation of the system under study. If the simulation model and its results are valid and are used as an aid in making decisions, then the model is said to be credible [Law99].

In [Law99], several techniques are described to verify and validate a simulation model; those used in this research are described later.

The HQ simulation models in this thesis are verified using the OPNET debugging functionality, allowing several kinds of traces and breakpoints to be applied during the execution of the simulation. The intermediate results and the overall simulation results have been checked for consistency and coherency.

The most definitive test of a simulation model's validity is to establish that its output data closely resemble the output data that would be expected from the actual system [Law99]. The actual systems that are used in this research are an already verified simulation models (the BBS model and the standard OPNET router model), and a hardware router. In addition, the results of the HQ simulation model, for example, the Cell Loss Probability (CLP) have also been compared to theoretical results.

2.8 Introduction to this research

As described in the introduction, this research work is about speeding-up simulation of large networks. Broadly speaking, it is a hybrid method that combines an analytical method and simulation together.

Foreground traffic is defined as the traffic of interest when investigating the network performance and the rest of the traffic streams in the network are denoted as *background traffic*. The major difference between this method and the other methods is that only the foreground traffic is actually simulated cell by cell (or packet by packet); background traffic (and its effect on the foreground behaviour) is represented using an analytical method. The rest of this thesis will explain this approach and its applications in detail.

This method not only targets the rare event as IS does; it does not need more than one computer to run the simulation, as parallel simulation does; it does not care if the traffic is 'bursty', as the cell-rate method does.

The following terminology is used in the later chapters:

- *HQ model* (or Hybrid Queuing model) is the term given to the approach described in this thesis.
- *Testbed* refers to the hardware equipment used for comparison with the HQ simulator.
- *Software testbed* refers to, for example, simulation models implemented in OPNET typically. It is the detailed cell-by-cell models of real products used for comparison with the HQ simulator.
- *Conventional model* refers to the conventional simulation modelling approach.

3. Speeding up a simulation using hybrid queuing

This chapter describes the principle of the proposed HQ approach in the context of simulating an ATM network. The queuing analysis of the ATM switch node is given and an example of the speed-up obtained is presented at the end of the chapter.

3.1 Background

As explained earlier, conventional simulation of packet-switched networks involve the use of discrete event simulators that simulate every individual cell through the network (cell level simulation). Each cell's arrival at, or departure from, a network element is represented by an event. However, for statistical reasons, a very large number of cells have to be simulated in order to guarantee the accuracy of those results. This always results in very long simulation time, often amounting to many hours of 'real' time just to simulate a few minutes of 'simulated' time.

Accelerated simulation techniques attempt to reduce simulation times, not only to enable results to be obtained more quickly, but also to increase the potential for dynamic interaction between simulators and experimental network management systems. Furthermore, off-line modelling of corporate networks is important too for network configuration management systems that are connected to real networks in order to assess the impact of a configuration decision before it is committed.

The original motivation of the HQ approach was to build a simulation model that is fast enough to emulate a real network. This could be run in parallel with the real network taking values from the management system to "keep in step" and then "disconnected" to analyse "what-if" scenarios in a faster and more flexible way. Hence, speed, with accuracy, is a major concern of the proposed HQ approach.

The major reason that the proposed HQ method can achieve speedup is that the method only generates and simulates a certain proportion of the traffic, the remaining traffic being taken into account by the analytical model. Hence, in an event-driven

simulation model, the number of events in the event-list is reduced, so substantially speeding up the simulation.

In this Chapter, the major issues of simulating an ATM switch are discussed, and the key analysis method (used in Chapters 5 and 6) is given.

3.2 Basics of ATM

3.2.1 Introduction

This section considers the basics of ATM, so that the context of simulation is apparent; it is not intended to be an in-depth treatment of the topic about which the reader can find much more detail in [Pattavina98][Handel94].

ATM is a packet-oriented transfer mode with the information flow being organised into blocks of fixed size, (called cells). Using fixed-length protocol data units simplifies the processing required at each ATM node and enables the use of high data rates. Cells are 53 bytes long each, divided into a payload (48 bytes) for user information and a header (5 bytes) for control data.

ATM is connection-oriented, i.e. cells are transferred onto virtual channels previously set up and identified by a label carried in the cell header. The main purpose of the ATM network is to provide connectivity, which is an end-to-end link for the user's information in ATM cells. To achieve this requires *switching* and *transmission*, together with the necessary *control* and *management*. However, *transmission*, *control* and *management* are beyond the scope of this research.

Switching defines how a connection is routed through a network from the source to the destination, and how channels of intermediate links are associated with each other to form a connection between two end points. Cells from many connections are multiplexed onto a link, and then queued at the switching node before each cell is individually switched to the appropriate destination link.

The ATM transport network is structured as a number of layers, the main area being the ATM layer and the physical layer. The transport functions of the ATM layer are independent of the physical layer implementation and are structured hierarchically. The transmission medium provides transport service to the virtual path (VP), and the VP provides transport service to the virtual channel (VC). From the protocol point of view, the function supported by the transmission medium itself is described in the physical layer and the functions supported by the VP and VC are in the ATM layer.

VCs and VPs are virtual connections between adjacent routing entities in an ATM network. A logical connection between two end-users passing through *n* switching nodes consists of a series of *n*-1 virtual connections.

This research focuses on the speeding up of an ATM network simulation, and hence the modelling of a switch. As an elemental building block of an ATM network, it is fundamental in the modelling of an ATM network. In most cases, a network simulator is only concerned with modelling user traffic and switching nodes connected in a certain topology, the details of the transmission network not being taken into account. In this circumstance, the modelling complexity of an ATM network simulation is mainly in the modelling of the ATM switch and for this reason the next section considers the ATM switch in more detail.

3.2.2 ATM switch architecture

ATM switching is the operation of transferring cells from the input to the output of the switch. All the functionalities relevant to the set-up and teardown of the virtual connection are handled by signalling. The signalling is responsible for setting up the necessary routing tables in the switch.

In this research, only a structure outline of an generic ATM switch is considered, not involving any detail or specific features a real ATM switch could have in its implementation. In brief, most ATM switch structures adopt the interconnection

network (IN - the switch core, see Figure 3-1) of multistage arrangements of very simple *switching elements* (SE) each, for example, using the *packet self-routing* [Handel94] concept. This technique consists in allowing each SE to switch (route) autonomously the received cell(s) by only using a self-routing tag preceding the cell that identifies the physical output link of the switch [Pattavina98].

The general diagram of an *N x N* switch (taken from [Pattavina98]) is illustrated in Figure 3-1 . This reference switch includes *N input port controllers* (IPC), *N output port controllers* (OPC) and an *interconnection network* (IN). The IN is usually a multistage arrangement of very simple switch elements (SEs), typically 2 x 2.

There are three types of SE:

- Input buffering
- Output buffering
- Shared buffers, or unbuffered (interconnection network queuing).

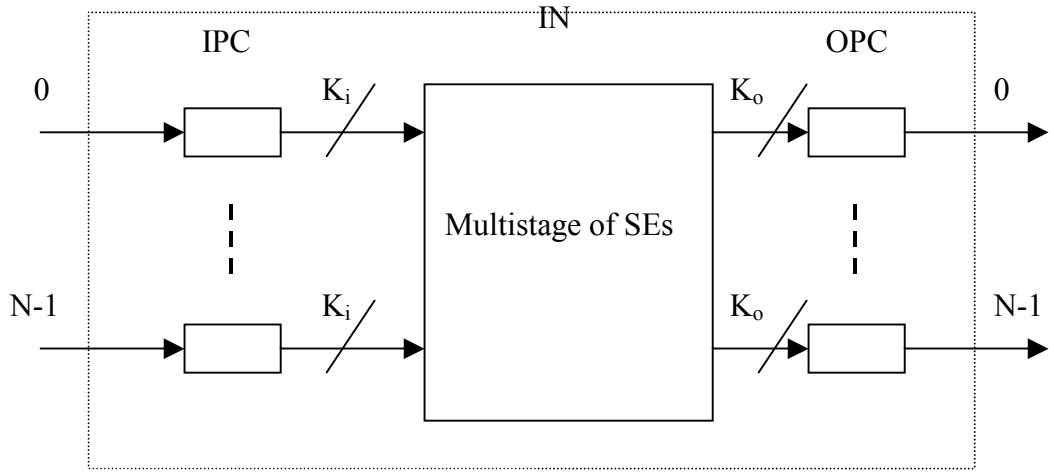


Figure 3-1 Model of ATM switch (adopted from [Pattavina98])

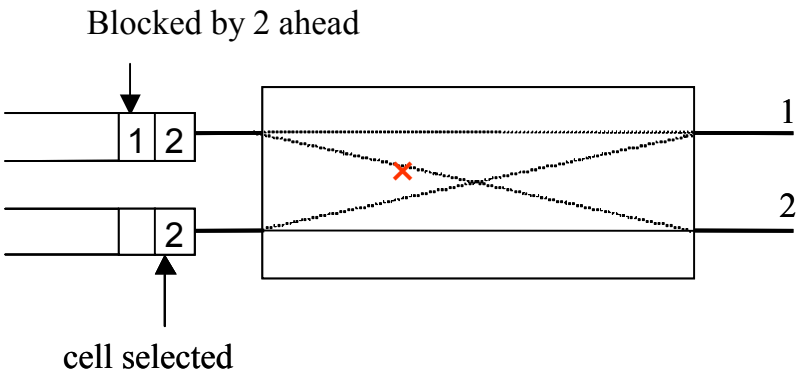
For input and output queuing, the buffering takes place at the IPC or OPC, respectively; shared queuing is accomplished by means of additional hardware associated with the IN. In the case of the unbuffered structure, the 'ideal' structure is

the *crossbar* network, in which each of the N^2 *crosspoints* in IN is dedicated to a specific input/output couple.

3.2.3 Buffering

3.2.3.1 Input *vs.* output buffering

Queuing in these nodes depends on how the switches can be arranged to provide the buffering. Output buffering is more commonly used and is generally preferable to input buffering since pure input buffering has a bad performance due to Head-Of-Line (HOL) blocking. This is simply demonstrated by using the illustration in Figure 3-2 where input buffering has the HOL blocking that the output buffering does not.



Number shown indicates destination

Figure 3-2 Input buffering HOL blocking, 2X2 switch

Figure 3-2 represents a 2X2 switch, with each input capable of being connected to either of the two outputs, with the cells queuing at each input buffer. Only one cell can be selected and sent to the output port at a time, the other one being blocked. For example, the bottom cell is selected, and then the upper one is blocked. In the upper buffer, the cell behind the blocked cells is to be routed to output port 1, and even though this port is available, the cell is blocked because it is behind the blocked cell to port 2. This results in a reduction of throughput and is known as HOL blocking [Schwartz96].

Because of HOL blocking in input buffering, output buffering is generally used more commonly in an ATM switch and hence, in this research, only output buffering is considered.

It is worth noting that there are ways to deal with HOL blocking and improve input buffering in the literature, such as [Lee99].

3.2.3.2 Output queuing

With output buffering model, each ATM node can be modelled as output buffers plus a high-speed switch whose performance is nearly non-blocking [Shwartz96]. Therefore, if the emphasis is put on an ATM network along a VC connection, the node can be modelled as a single server queue, where a server corresponds to a VP and a service performed by the server corresponds to transmission of a cell.

In order to give a clearer image of the architecture of a SE with output buffering, Figure 3-3 shows more details, including VP and VC.

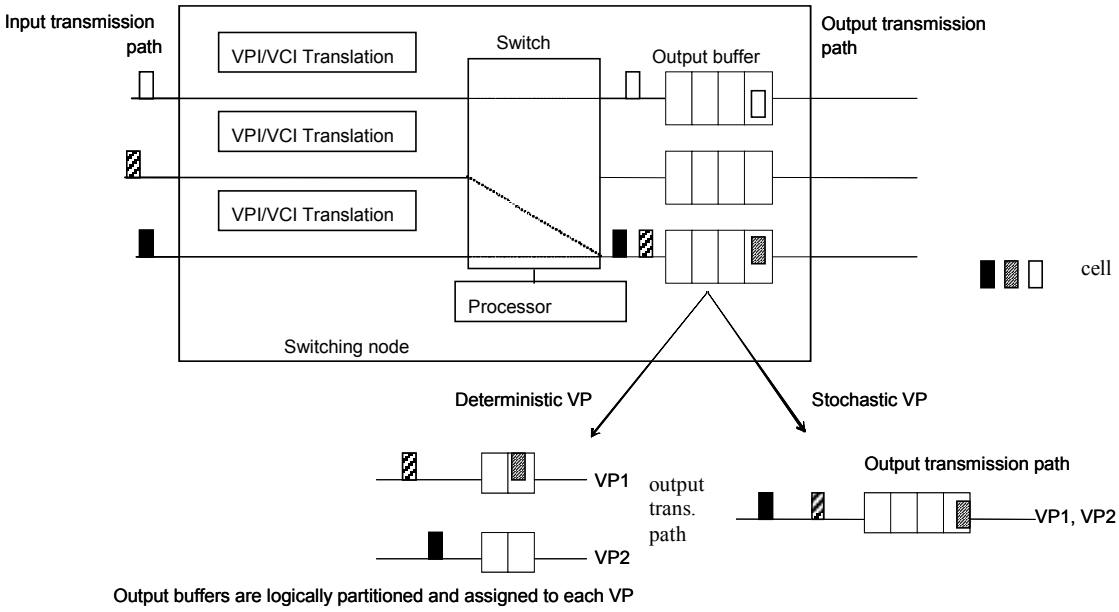


Figure 3-3 ATM switching node (adapted from [Saito93])

3.3 ATM switch modelling in this thesis

3.3.1 Structure

As described in Chapter 2, in a simulation, the level of detail used for the modelling depends on the problem to be solved. Therefore, the ATM node structure is simplified in this research, and shown in Figure 3-4. Figure 3-4 is a commonly used representation [Schwartz96] [Pitts01] of an ATM switch in modelling and is used throughout this thesis. Actually, only buffers are used in this thesis. The switch can switch according to the VPI or VCI values, depending on what the research focus is; the detailed internal structure of the switch is not of interest. This research is only concerned about how to model this switch as fast and as accurately as possible. To achieve this, it is more important to look into the traffic characteristics and the buffering behaviour.

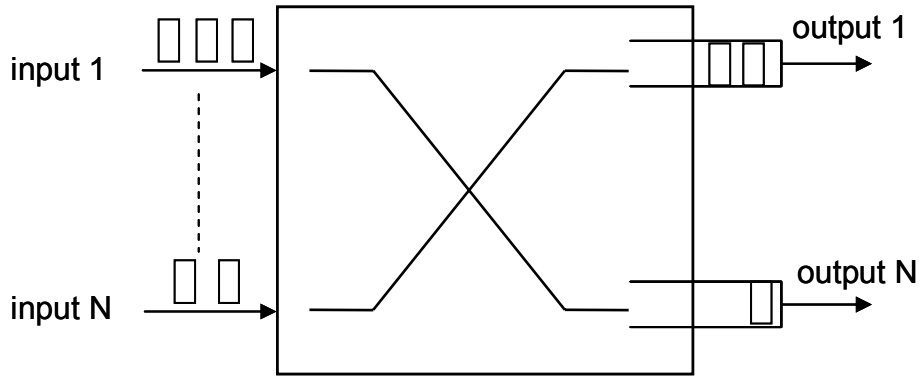


Figure 3-4 Output buffering

3.3.2 Poisson Processes

Poisson process is central to physical process modelling and plays a pivotal role in classical queueing theory. In most of the elementary queueing systems, the arrival and service processes are assumed to be Poisson.

Assume, in Figure 3-4, that there are N input ports and N output ports in the ATM switch. Considering output port j, the cells in this particular output port all come from N input ports and, the arrival process to this output port, in any time slot, is often modelled as a Binomial distribution [Ng97], where "Pr" denotes "probability of".

Pr [k arrivals in a timeslot] = $\binom{N}{k} p^k (1-p)^{N-k}$

When the number of ports N increases, the probability p that a cell is destined to a specific output port from a specific input port decreases proportionally, but the product of Np maintains constant, i.e. $Np = \lambda$. Then the above Binominal distribution converges to Poisson distribution as follows:

Pr[k arrivals]

$$= \lim_{N\to\infty} \binom{N}{k} \left(\frac{\lambda}{N}\right)^k \left(1-\frac{\lambda}{N}\right)^{N-k}$$

$$= \frac{(\lambda)^k}{k!} \left[\lim_{N\to\infty} \frac{N(N-1)...(N-k+1)}{N^k}\right] \left[\lim_{N\to\infty} (1-\frac{\lambda}{N})^N\right]$$

$$= \frac{(\lambda)^k}{k!} \left[\lim_{N\to\infty} \left\{(1-\frac{\lambda}{N})^{N/\lambda}\right\}^{-\lambda}\right]$$

$$= \frac{(\lambda)^k}{k!} e^{-\lambda}$$

In the above expression, use can be made of the identity:

$$e = \lim_{N\to\infty} \left(1-\frac{\lambda}{N}\right)^{N/\lambda}$$

The above derivation shows that, as the size of the ATM switch increases, with the number N of input and output lines getting very large, (i.e., $N \to \infty$), the Binominal arrival processes approach the Poisson distribution. Because the cells have a fixed sizes in ATM networks, the M/D/1 queuing model is often used to model ATM traffic [Pitts01].

3.3.3 ATM queuing behaviour

ATM queuing behaviour is governed by two different effects in the cell arrival process [Roberts91] [Norros91][Pitts95] called *cell scale* queueing and *burst scale* queueing.

Cell scale queueing occurs when a bunch of cells from different sources happens to arrive together, even though the aggregate arrival rate is less than the buffer service rate. It results in cell loss for a small buffer size, but the cell loss probability drops rapidly as the buffer size increases. The Poisson arrival process often applies to modelling the buffer queueing. The *cell scale queueing* is described in detailed in [Roberts91] [Norros91][Pitts01].

Burst scale queueing occurs when the aggregate arrival rate temporarily exceeds the buffer service rate, causing the buffer to fill and overflow. The *burst scale queueing* is the key factor causing cell loss and cell delay variation [Roberts91] [Norros91], and is also described in detail in Pitts and Schormans' book [Pitts01].

For a given offered traffic load, the cell loss probability varies with buffer size, as depicted in Figure 3-5 (from [Schwartz96]). There are two distinct queueing behaviours to this relation corresponding to congestion. The two components can be more strictly expressed as probabilities of the joint events {buffer saturation and arrival rate less than multiplex capacity} and {buffer saturation and arrival rate greater than the multiplex capacity}, respectively. For simplicity, each component is approximated by a straight line on the log-linear axes.

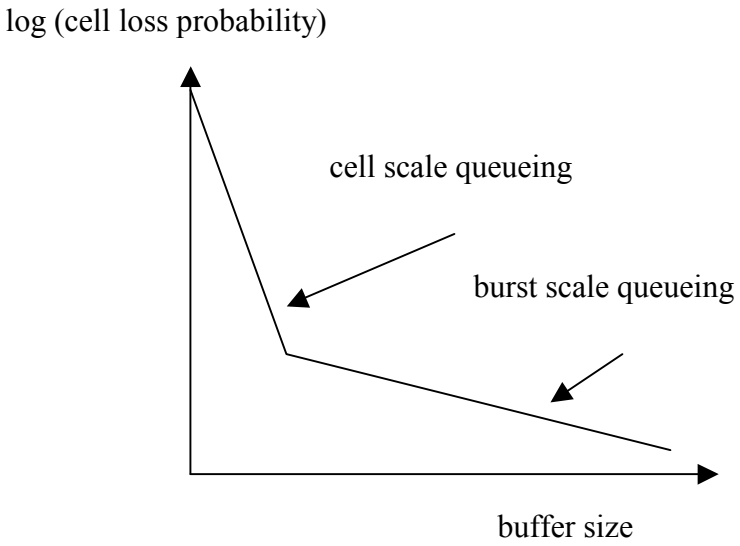


Figure 3-5 Cell loss probability at cell scale queueing and burst scale queueing

To account for the two queuing behaviours, however, that appear with more complex traffic types than the simple Poisson model, one must resort to more realistic models of certain doubly-stochastic processes [Schwartz96] that is beyond this thesis. On the other hand, it is possible to separate these two congestions when only considering the behaviour (see [Baiocchi91] [Roberts91] [Norros91]) with models tailored either to the cell or burst scales. This research deals with the issue in this way, and makes full use of the distinct characteristics of cell scale and burst scale queuing. A brief description of cell scale and burst scale queueing is presented in the following sections.

3.3.3.1 Cell scale queuing

Cell scale queueing occurs due to simultaneous arrival of cells from independent sources while the overall cell arrival rate of active sources is less than its multiplexer capacity.

As explained earlier, it is reasonably accurate to model the total cell arrival process from all the sources as a Poisson process when the number of sources is large enough and the cells appear randomly at the input lines.

Consider the example shown in Figure 3-6.

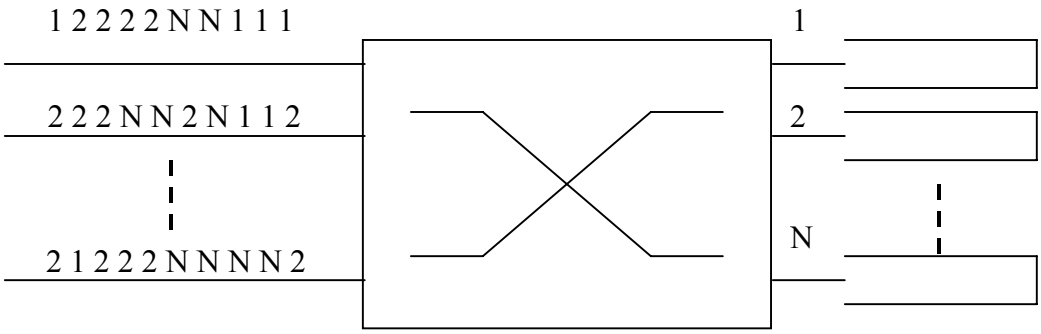


Figure 3-6 An example for cell scale queuing

At the input ports, the numbers assigned are the destination output ports and each number represents a cell. For example, at slot t+6, the process has overloaded the queuing system because two cells have arrived, one more than the buffer can transmit. The process provides short periods during which its instantaneous arrival rate is greater than the cell service rate. The queue length increases until at some time there are no arrivals, when it can start to decrease.

Cell scale queueing arises not only as a result of source multiplexing, but also happens in switching. The cell streams on the input to the switching element are the output of another buffer. The same queueing principle applies at the switch output buffer as at the source. The sources may all be CBR, and the individual input ports to the switch may contain cells such that their aggregate arrival rate is less than the output rate of either of the switch output ports, but still there can be cell loss in the switch.

In the cell scale, queueing is a 'local phenomenon' occurring while the composition of active sources remains constant. When the overall arrival rate remains below multiplex capacity, the system behaves like the so-called $N \cdot D / D / 1$ queue [Norros91], [Eckberg79]. The input process comprises N independent periodic sources, each source with the same period, D. [Pitts01] has the detailed analysis of an $N \cdot D / D / 1$ queue in the case of cell-scale queueing. The M/D/1 model can be used as approximation for the heavy traffic.

3.3.3.2 Burst scale queuing

Burst scale queuing occurs when the overall arrival rate is momentarily greater than multiplexer capacity; buffer content continues to grow until saturation as long as the arrival rate excess exists.

Here the basic reason for a queue to build up is because the service rate is smaller than the arrival rate. In one of the nodes in a real network, when *N* sources are aggregated together, the aggregated arrival rate can be greater than the service rate over a certain time period. During this time period the queue builds up and when there are no cell arrivals, or when the aggregate arrival rate is less than the service rate, the queue size decreases. Provided that the period of congestion is sufficiently short, bursts can be accommodated in the buffer and analysis can assess the demand for buffering capacity. If very low probability of cell loss is required in a practical system, however, the buffer often has to be very large, with a consequential impact on the delay time. For this reason, loss-sensitive services (requiring a large buffer) and delay sensitive services (requiring a short buffer) are often separated.

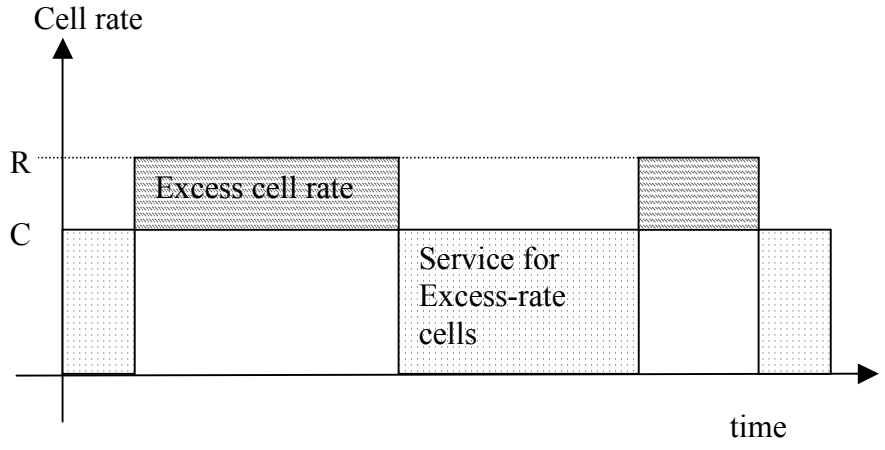


Figure 3-7 Burst scale queuing with single ON-OFF source (from [Pitts 01])

The analysis of burst scale queuing in this research is based on previous research results described in [Schormans96]: the so-called *Excess Rate* (ER) queueing. ER arrivals refer to those cells that must be buffered as they represent an excess of

(instantaneous[2]) arrivals rate over the service rate. Therefore, if the service rate is less than the arrival rate, this represents one, or more, excess arrivals during the service period. The excess-rate cells are the cells that cannot be served upon arrival because they arise from "excess-rate" bursts.

There are two separate steps to analyse the effects of the excess-rate:

- Determine the probability that an arriving cell becomes an excess-rate cell/packet.
- Determine the probability that such a cell is lost due to the buffer overflow.

Therefore, the overall loss probability arising from burst scale queuing is:

$$\Pr\{\text{cell is lost}\} = \Pr\{\text{buffer overflow} \mid \text{cell needs buffer}^3\} * \Pr\{\text{cell needs buffer}\}$$

[Pitts01] shows that there are two main approaches to the analysis of burst scale queuing. The historical approach is to model the flow of cells into the buffer as though it was a continuous fluid, ignoring the structure of the flow (e.g. bits, octets, or cells). The alternative approach is the *discrete* fluid-flow approach, which actually models the individual excess-rate cells. This research uses the latter approach when considering burst-scale effects.

Instead of finding the state probabilities at the end of a time slot, the ER method finds the probability that an arriving excess-rate cell finds *k* cells in the buffer. If the arriving excess-rate cell finds the buffer full, the cell is lost, and the Cell Loss Probability for the excess-rate is the probability of this event occurring.

---

[2] In the case of cell-scale

[3] 'Cell needs buffer' means a cell arrives but cannot be served immediately and therefore has to be buffered.

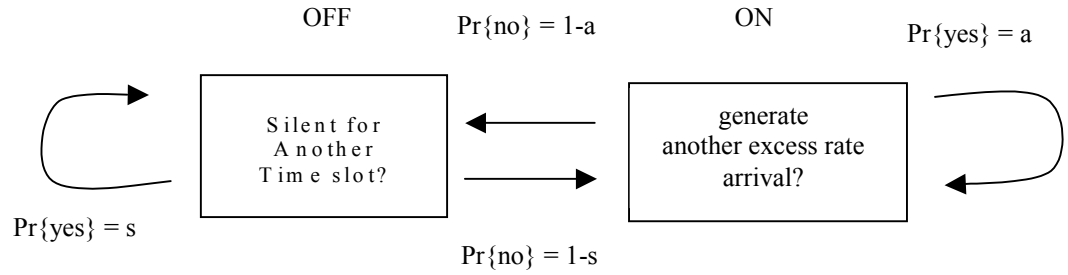


Figure 3-8 The ON/OFF model source for the discrete "fluid-flow" approach

This is given as follows (from [Pitts01]), where R is mean cell arrival rate during ON period and C is service rate of queue:

- If the source is in the OFF state AND the buffer is empty, then it remains empty;
- If the source is in the OFF state AND the buffer is NOT empty, then it empties at rate C;
- If the source is in the ON state AND the buffer is NOT full, then it fills at a rate R-C
- If the source is in the ON state AND the buffer is full, then cells are lost at a rate R-C

The queue decreases only when there is no arrival in a service period, and it remains the same when there is exactly one arrival in a service period. For the queue to decrease by *k* cells/packets between excess-rate arrivals there must be *k* service times with no arrivals and any number of service times with one arrival.

3.4 Foreground and background traffic

This is an important aspect of this research as the concept is applied in a new way.

When investigating the performance of a telecom network, only a few connections are generally of interest; these are defined here as the *foreground traffic*. It can be, for example, a data connection across a typical ATM network, or a single voice connection, or a bundle of connections, such as a VP. The definition of what is the foreground traffic in any study is entirely under the user's control.

The rest of the traffic flows, no matter where they are generated, and where they are terminated, are regarded as the *background traffic*. The background traffic affects the

performance of the foreground traffic and it is essential to include it in the model to ensure correct representation of the functioning of the network, but measurements on it are not the focus of the study

In the method described in this thesis, the background traffic is represented by an analytical method and is not simulated packet by packet. This reduces the number of events in an event-driven simulation, thereby reducing the simulation time. In particular, in large simulations where there is a lot of background traffic, the number of events is reduced tremendously; hence, the simulation time is significantly reduced.

The actual effect of the background traffic is to increase the service time for the foreground traffic and so this method modifies the queue behaviour in response to the background traffic.

Figure 3-9 illustrates what happens in a buffer with the actual traffic (foreground + background) and foreground traffic only, but where the foreground cells still occur at the correct positions. It appears that there is "invisible" background traffic that means that foreground traffic is still delayed. As shown in Figure 3-9, this is handled by modifying the service time so that the new times ST_1 and ST_2 output the foreground cells at the correct time. Notice that this is a distribution, not a fixed new value, and that the service time is *increased* on average.

The aim of this research is to find this modified service time distribution and parameters so that the inter-exit time of the foreground cells is exactly the same as it would have been with the background events present. The inter-exit time (IET) is the time interval between two-successive cells leaving the switch. By removing the background traffic from the event-driven simulator the number of cells actually being dealt with is reduced, so that the number of events is reduced, and hence the simulation time is also decreased.

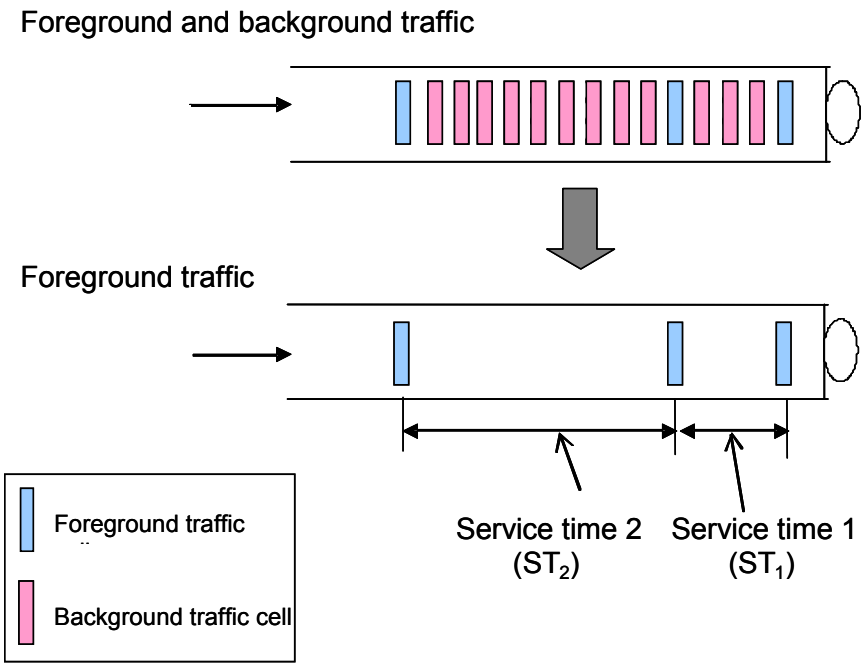


Figure 3-9 Foreground vs. background traffic

In this research, three scenarios have been considered:

- a single ATM buffer;
- a multiple priority ATM buffer;
- communication via an ATM link featuring burst-scale queuing.

In the first two cases, Poisson distribution is assumed for the arrivals; in the last scenario, ON-OFF sources are used. Validation is described in the next chapter.

This approach is completely novel, although [Hao98] used the same definition of foreground and background traffic. However, the method used there is not at all the same as proposed in this thesis.

The main idea behind the approach in [Hao98] is to run the simulation in full detail, while observing the traffic at critical points in the network. Then the authors used the observations to derive aggregate representations, which are then substituted for the fine-grained representations during subsequent simulation runs. The assumption was that the background traffic must be stationary, and the foreground traffic is relatively small. They chose three kinds of aggregation models: (i) inter-arrival-based, (ii)

ON/OFF burst-based and (iii) multi-state ON/OFF. The paper concluded that the ON/OFF burst-based model is accurate and easy to implement.

3.5 Derivation of the equivalent service time parameters

The major concern in this research is the service time for the foreground traffic, taking into account the effect of the "invisible" background traffic. It is understood from queuing theory that there are many aspects to buffering in WANs that affect the traffic performance as seen by the user: delays (both mean and tail probabilities) and cell loss probability being dominant. However, it is an essential point, see [Abate95], that both the delay and loss aspect of buffering can be incorporated via the *state probabilities of the buffer as seen by arriving cells.* In this thesis the specific aspect of the queue that was forced to be equivalent is the decay rate of the state probabilities (specifically the state probabilities associated with the number of cells in the buffer at cell arrival instants).

In queuing applications, interest is often in the tail probabilities of the steady-state waiting time. These tail probabilities often have approximately an exponential form with parameters that can be determined. [Abate95] discussed corresponding exponential approximations for the sojourn time (response time, i.e. waiting time plus service time) and the workload (virtual waiting time), and geometric approximations for the queue lengths (at arbitrary times, departure epochs, and arrival epochs).

Suppose that W is the steady-state waiting time. A simple exponential approximation for suitably large x is [Abate95]:

$$P_r\{W > x\} \approx \alpha \cdot e^{-\eta x} \qquad (3.1)$$

where the *decay rate* η and the *constant* α are fixed positive real numbers independent of x. It is important to note that (3.1) does not hold for all x, but only for suitably large x, i.e. $x \to \infty$,

Let p denote the probability that the steady-state waiting time is shorter than a certain value. Only the case where p is large, for instance, $p = 0.95$, is of interest. w_p is the value such that $\Pr\{W > w_p\} = 1 - p$. Taking natural logarithms of both side of the equation in (3.1), gives

$\ln e^{\eta w_p} = \ln(\frac{\alpha}{1-p})$, then:

$$w_p = \ln(\frac{\alpha}{1-p}) \cdot \frac{1}{\eta} \qquad (3.2)$$

Moreover, the percentile (3.2) often does not depend greatly on the constant α, so a relatively crude approximation of α=1 is often used. This simplification in (3.2) obviously becomes more appropriate as p and α approach 1. Having the approximation not depend critically on α is helpful because an appropriate constant α is often much more difficult to determine than an appropriate decay rate η.

The remarkable quality of exponential approximations for waiting-time tail probabilities in the GI/GI/s queue is pointed out with abundant numerical evidence in [Tijms86] and [Seelen85].

The above description conceptually addresses an exponential presentation of the decay rate, which is employed for a continuous waiting-time distribution. However, this research deals with queueing with a discrete waiting-time. For a discrete waiting-time distribution, a more convenient presentation of geometric distribution is employed for the decay rate. It is formulated as (3.3) for suitably large *x*:

$$\Pr\{W > x\} = C\eta^x \qquad (3.3)$$

where η is decay rate and *C* a constant parameter.

The decay rate of geometric distribution is adopted in the formula derivation in Chapter 5, 6 and 7. Figure 3-10 is drawn to illustrate the decay rate method in approximating the tail of waiting-time distribution. The illustration implies a buffer size of 10, beyond which the probability distribution is referred to as the tail distribution of interest, and dealt with by decay rate approximation.

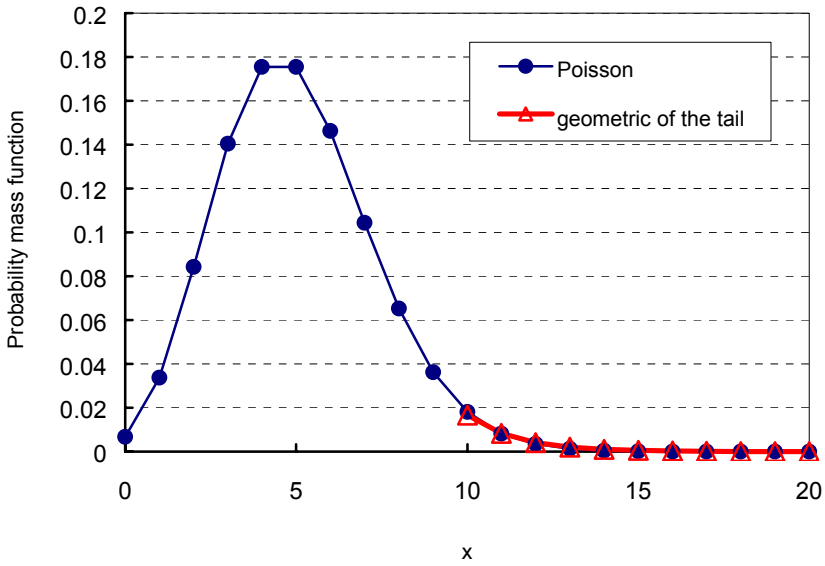


Figure 3-10 Illustration of decay rate approximating waiting-time tail distribution

3.6 Magnitude of the speed-up factor.

The speedup factor depends very much upon the proportion of the foreground traffic to the total traffic. In an extreme case where the foreground traffic is equal to the total traffic, the simulation results have actually no difference from the traditional ones. However, a quick illustration of the speed-up factor shows that the concept does have significant potential.

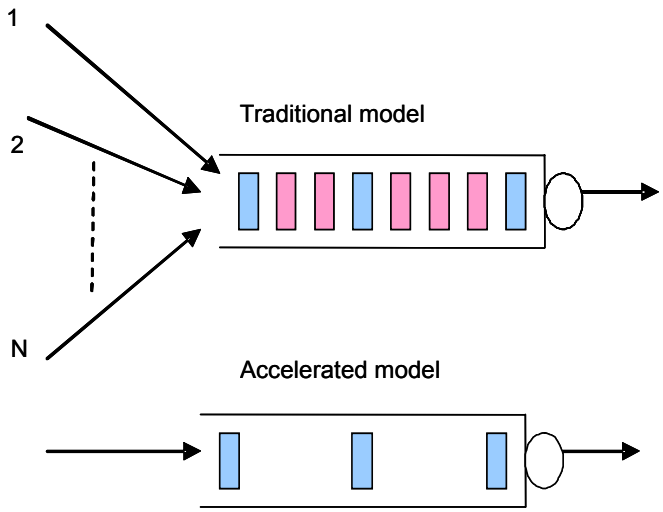


Figure 3-11 Example of the speed up magnitude

To quantity the magnitude of the speedup factor in this research, consider the case illustrated in Figure 3-11. The foreground traffic is a cell stream with rate *b*, which is a specific connection of interest. The background traffic consists of *N*-1 sources identical to the foreground traffic. The traditional techniques simulate all the cells belonging to the *N* sources, shown in the upper part of Figure 3-11, while the accelerated simulation only simulates the traffic of interest, which is one specific stream, as shown in the lower part of Figure 3-11. The total traffic takes up ρ of link capacity C. Therefore, for each foreground cell event, the traditional model would simulate $N = \rho \cdot C / b$ events.

To get *M* lost foreground cells at a cell loss rate of γ, it would be necessary to simulate $M \cdot (\rho \cdot C / b) / \gamma$ events in the traditional simulations, and only M / γ in the accelerated one respectively.

Thus, the speedup factor S is: $S = \dfrac{M \cdot (\rho \cdot C / b) / \gamma}{M / \gamma} = \rho \cdot C / b$

Using a voice service as example, typical values of the above entities can be:

ρ = 0.5 (medium load)

C = 150 Mbit/s (ATM)

b = 64 kbit/s

γ = 10^{-6} (this is the typical cell loss probability in the ATM network, it could be less than this, depending on the request)

M = 50 (minimum quantity to reach a usable simulation result)

$$S = \frac{50*150*10^6}{64*10^3} = 1172$$

This is an example of speedup achieved by HQ approach with a simple queue. In the tests in Chapters 5 and 6, the speedup factors between software testbed, conventional model and the HQ simulation are calculated individually. The tests were done on the same Sun Solaris workstation so simulation times could be realistically compared. Although, the speedup value varies, this method does have the potential to significantly speed up the simulation, even at moderate values of network utilisation.

4. Making the fast simulation more accurate

Chapter 3 proposed a fast simulation approach, which used an analytical model for the effect of the background traffic. The method has the potential to substantially speed up the simulation.

However, the model is, of necessity, an approximation and so the results cannot be expected to be as accurate as a fully detailed switch simulation. The deviation between the model and reality can be reduced by adjusting the parameters of the model. In this chapter, the second part of the HQ approach is explained and it is this addition that is used to make the model more accurate. A neural network is introduced to learn the difference between the HQ model and a conventional model or even a testbed. Once the neural network is trained, there is no need to run the conventional model or the testbed, the embedded neural network's prediction is used to 'fine-tune' the model accordingly. The simulation results indicate that the HQ method, which comprises of analytical model and neural network, achieves fast simulation while remaining accurate.

4.1 Validation of the HQ model

As explained earlier, it is necessary to validate a new model or a simulator to determine whether the model does give an accurate representation of the system under study.

The HQ approach was tried out in three scenarios: (i) the single ATM buffer, (ii) ATM buffer with multiple priorities, and (iii) burst-scale queuing. In each case, the validation of the method itself is divided into two steps: (i) validation against an analytic solution if it is applicable and (ii) validation of the approach by comparing the simulation result to a traditional, already validated simulation model, such as OPNET library model, or even a testbed.

The validation tests are detailed in Chapter 5.2.4, 6.5, and 7.2; all the tests results in these sections illustrated that the analytical method works well.

The differences can be reduced by adjusting the service rate parameters so that the analytic model gives a better representation of the background traffic. This chapter is concerned with how to make that adjustment. This *concept* is illustrated in Figure 4-1, where setting the "control" of the service time to be "faster" reduces the inter-exit times of the foreground traffic, hence reducing the effect of the background traffic.

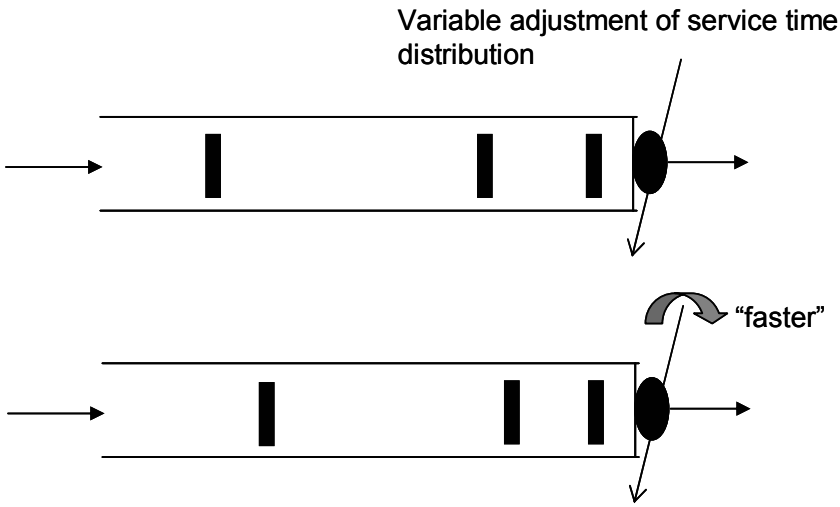


Figure 4-1 Concept of refining the model

The problem is now how to set the value of the control to make the model as accurate as possible. This can be done by using a neural network to learn the adjustment for different conditions by performing a series of training runs under different conditions. These training runs can use both a conventional event simulator as the "real" system, and, where appropriate, a physical network or network element (testbed). Once trained, the neural networks were used to predict the inherent difference between the HQ model and the system being simulated and hence gave more accurate results. Chapters 5 and 6 show how the neural networks were used in the application of the HQ approach in cell scale and burst scale queuing respectively.

4.2 Re-alignment of the HQ model

The difference between the outputs of the HQ model and the standard model is obtained by running the HQ model and the standard model separately under the same input conditions, the intention being to find a way of knowing the output difference without running the standard model.

Let u denote the observed output of the abstracted model, v the observed output of the standard model when the input vector $\mathbf{X}$ is applied to both models. There exists an unknown relation of (u - v) to $\mathbf{X}$, i.e.

$$e = u - v = f(\mathbf{X}) \tag{4.1}$$

By running both models over a range of the input, a series of sample pairs ($f(\mathbf{X}_i)$, $\mathbf{X}_i$) (i=0,1,…m) can be observed. To find the unknown relation f, traditionally it is necessary to select a function class $\varphi = \{\varphi_0, \varphi_1, ..., \varphi_n\}$, and use a certain criterion, say, minimum square error (MSE). Function $S^*(\mathbf{X})$ will best represent $f(\mathbf{X})$ in class φ if it minimises the following error:

$$\sum_{i=0}^{m} \delta_i^2 = \sum_{i=0}^{m} [S^*(\mathbf{X}_i) - f(\mathbf{X}_i)]^2 = \min_{S \in \varphi} \sum_{i=0}^{m} [S(\mathbf{X}_i) - f(\mathbf{X}_i)]^2 \tag{4.2}$$

where $S(\mathbf{X}) = a_0\varphi_0(\mathbf{X}) + a_1\varphi_1(\mathbf{X}) + \cdots + a_n\varphi_n(\mathbf{X})$ (n<m). This sort of method has the difficulty of choosing how to select a suitable function class and how to find an efficient search algorithm to determine a_0, a_1,... a_n, especially when the underlying relation is a complicated non-linear one. In the scenario here, this is a relation of input traffic parameters to the simulation results, which is complicated. However, neural networks have proved to be more powerful and faster than traditional methods in solving this kind of problem.

To approximate an unknown function $f(\mathbf{X})$, a feedforward neural network trained by the *error backpropagation algorithm* [Fausett94] is considered. The neural network is

described by $f(\mathbf{X}, \mathbf{W})$, where **W** is a vector representing variable weights. The task is to train $\hat{f}$ to approximate f by sequentially applying input vectors (training data) to the neural network, while adjusting network weights according to a predetermined performance objective.

The HQ model and the standard model ran separately under the same series of traffic inputs to generate the training data. Once trained, the neural network can compensate for the output difference between the standard model and the simulator by changing control inputs to the HQ model before it is run. This allows the HQ model to more accurately represent the real system when new traffic inputs are applied without having to run the standard model. This ensures that the HQ model gave accurate results and ran faster. The diagram of the overall model is shown in Figure 4-2.

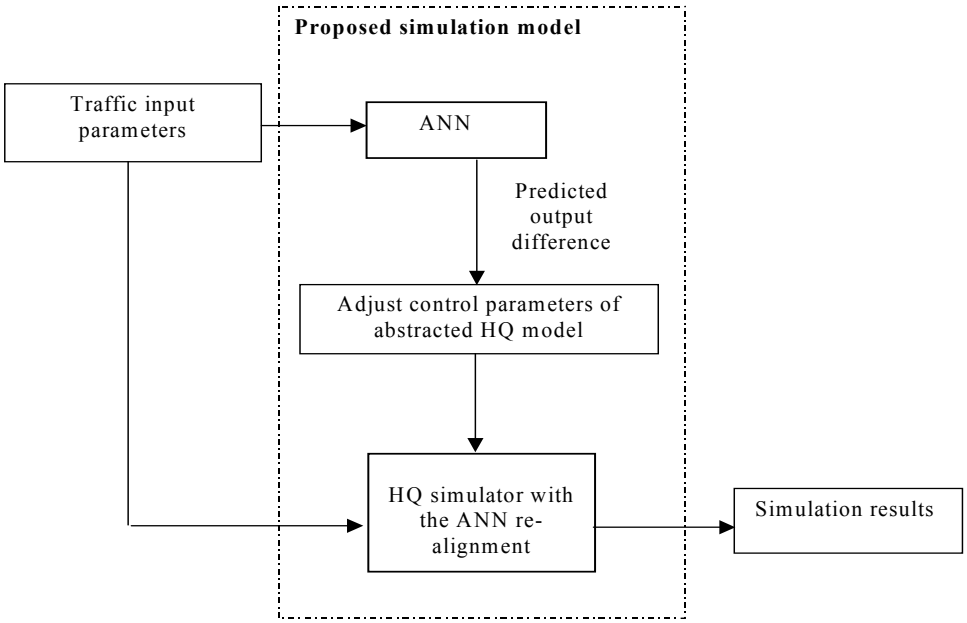


Figure 4-2 Block diagram of the proposed model

4.3 Neural network applications in telecommunications

Having chosen to use neural networks in this way, it is worth reviewing their application to telecommunication networks, particularly to simulation of such networks to see if there is any similarity to the research of this thesis.

Neural networks have indeed been widely used in the telecommunication area, for instance there are applications for Connection Admission Control (CAC) in ATM. The book [Yuhas94] gives a good summary of the applications of neural networks in telecommunications.

Among papers in the literature are works on solving particular telecom problems that have some shortcomings when using traditional methods. However, only [Cassandras97] has any kind of similarity to this thesis. In that paper, a neural network was used as a 'substitution' of a simulation (because it is much faster) based on the previous simulation results.

In [Cassandras97], the authors proposed that for large complex systems, evaluating performance and exploring alternatives is a computationally slow process, and currently still beyond real-time applications. They pointed out three possible advances that might help to overcome these limitations:

(i) concurrent simulation based on developments in the theory of discrete event systems, enabling the extraction of information from a single simulation that would otherwise require multiple repeated simulations, and effectively providing simulation speedups of possibly orders of magnitude;

(ii) simulation for the purpose of obtaining a "metamodel" of the actual system, i.e. an approximate "surrogate" model that is computationally very fast, yet accurate. They specially discussed the use of neural networks as metamodelling devices, which may be trained through simulation. This is of relevance to this research because of the use of neural networks, and also because the HQ simulation may be thought of as a "metamodel", although in this case the metamodel is not obtained through simulation. The work in this area described in [Cassandras97] is discussed below;

(iii) hierarchical simulations provide another means for speed-up, a major challenge being the preservation of fidelity between hierarchical levels.

For the metamodelling method described in [Cassandras97], the authors used the "back-propagation algorithm" for their neural networks. The metamodelling procedure they followed is outlined in Figure 4-3. They start by running a large-scale simulation with possibly hundreds or thousands of inputs and hundreds of thousands of outputs. The neural network is a device that they have separately designed (completely independent of the simulator) and fed with exactly the same inputs and outputs as the simulator; it is not, however, part of the simulator. While the simulator is running, the neural network observes the inputs and outputs and "learns" from them, "training" the neural network. It is possible to visualise the neural network as an "entity" that is highly intelligent but has no knowledge of anything initially to apply its intelligence to. As it observes the simulation unfold, however, it learns from the basic cause-effect (i.e. input-output) relationships it observes. In fact, all the neural network does is to adjust its weights so as to emulate the behaviour it observes as closely as possible.

When the training is finished, the simulator can be removed. The neural network is now the surrogate model: it may be given input (as if the input were being given to the simulator) and it gives an output (as the simulator would, only faster). Therefore, it can be thought of as a "function" which responds to any input by providing some output, except that there is no explicit mathematical expression, just a device that acts as the model. A schematic diagram is shown in Figure 4-3.

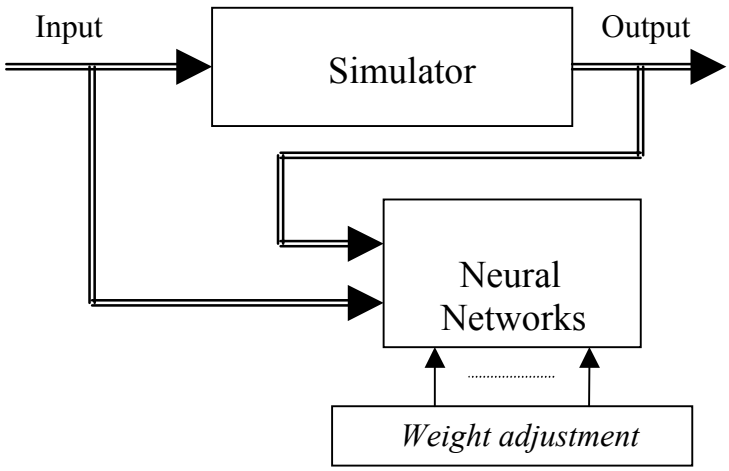


Figure 4-3 Neural network training through simulation to create a surrogate model

The neural network application in this thesis is not the same as in [Cassandras97]. In their work, a neural network is used as surrogate of the simulation that provides the simulation results once being well trained. Here the neural network is used to adjust the parameters of a highly abstracted model for simulation.

4.4 Why neural networks?

Why was a neural network chosen instead of some other mathematical approach, such as curve approximation? This section will answer that question.

Neural networks are not programmed; instead they are trained by being repeatedly shown large numbers of examples for the problem under consideration. As a result of this, they can provide a good result in relatively short timescales, which is exactly what is required for setting up the simulation. In [Tarassenko98], the author summarised some key attributes of neural networks. The attributes and how this application fits into them can be summarised as follows:

- *Learning from experience*: neural networks are particularly suited to problems whose solution is complex and difficult to specify, but which provide an abundance of data from which a response can be learnt. *The relation that needs to be found here is a function between the traffic input parameters and the difference between the simulation results, which is complex and difficult to specify.*

- *Generalising from examples*: a vital attribute of any practical self-learning system is the ability to interpolate from a previous learning "experience". *The experience here is that the output difference between the HQ and standard models is known, and then the analytical model within the HQ model can be adjusted accordingly. Having used the relationship between input parameters and the output differences to train the neural network, the neural network can give the correct response to data that it has not previously encountered.*
- *Developing solutions faster, and with less reliance on domain expertise*: neural networks learn by example, and as long as examples are available and an appropriate design is adopted, effective solutions can be constructed far more quickly than is possible using traditional approaches. *This is a very important factor in the method proposed here: as the overall objective is to speed up a simulation, it is obvious that it is undesirable to spend too much time on the model refining before actually running the simulation itself.*
- *Computational efficiency*: training a neural network is computationally intensive, but the computational requirements of a fully trained neural network when it is used on new data can be modest. For very large problems, speed can be gained through parallel processing, as neural networks are intrinsically parallel structures. *The data used to train the neural network here is not complicated and since only two inputs and one output are used, this is not a problem for this application.*
- *Non-linearity*: many other processing techniques are based on the theory of linear systems. In contrast, neural networks can be trained to generate non-linear mappings and this often gives them an advantage for dealing with complex, real-world problems. *The problem encountered here is non-linear and it is difficult to choose a function to represent this relation. For example, as depicted in Chapter 7, in the comparison test against the Linux workstation, two reasons are listed for the relatively large difference, and it is not easy to see how those differences might be eliminated; this is one of the main reasons for using neural networks.*

4.5 Establishing a neural network

An artificial neural network is an information-processing system and has been developed as a generalisation of mathematical models of human recognition of neural biology, based on the assumptions that:

- Information processing occurs at many simple elements called neurons.
- Signals are passed between neurons over connection links.
- Each connection link has an associated weight, which, in a typical neural network, multiplies the signal transmitted.
- Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

The building blocks of a neural network are: architecture, algorithm and activation functions.

4.5.1 Architecture

The architecture of a neural network is the pattern of connections between the neurons. It is often convenient to visualise the neurons as being arranged in layers. Within each layer, neurons usually have the same activation function and the same pattern of connections to other neurons. In many neural networks, the neurons within a layer are either fully interconnected or not connected at all. In the fully-interconnected case, if any neuron in a layer (layer A) is connected to a neuron in another layer (layer B), then every neuron in layer A is connected to every neuron in layer B.

Neural networks are often classified as single layer and multilayer. A single layer network has a number of important limitations in terms of the ranges of functions that it can represent [Bishop95]. To allow for more general mappings, successive transformations corresponding to networks having several layers of adaptive weights are widely used. In fact, networks with just two layers of weights are capable of approximating any continuous functional mapping [Bishop95].

A feed-forward network is where the outputs from a set of units (neurons) are connected only to the inputs of the units (neurons) in the next layer [Tarassenko98]. Feed-forward neural networks are regarded as providing a general framework for representing non-linear functional mappings between a set of input variables and a set of output variables [Bishop95].

Figure 4-4 shows a typical feedforward multilayer neural network. In these, the signals flow from the input units to the output units in a forward direction. Feed-forward networks have no cycle in their directed graphs, so the flow of information is non-recurrent. *The neural network models discussed in this research are limited to feed-forward neural.*

Often feed-forward neural networks are composed of layers of units that are stacked. There are no connections between units within a given layer, but all of the units in one layer are connected to the units in the layer above. As shown in Figure 4-4, without the intermediate layer, the structure is called single layer neural network. The intermediate layer (Z) is normally called the "hidden layer" and, depending on the particular problems to be solved, there could be many hidden layers. $X_1, X_2, \ldots X_n$ is called the input layer; $\{X_i\}$ is the set of input units; the $Y_1, Y_2, \ldots Y_n$ is called the output layer, and $\{Y_i\}$ is the set of output units.

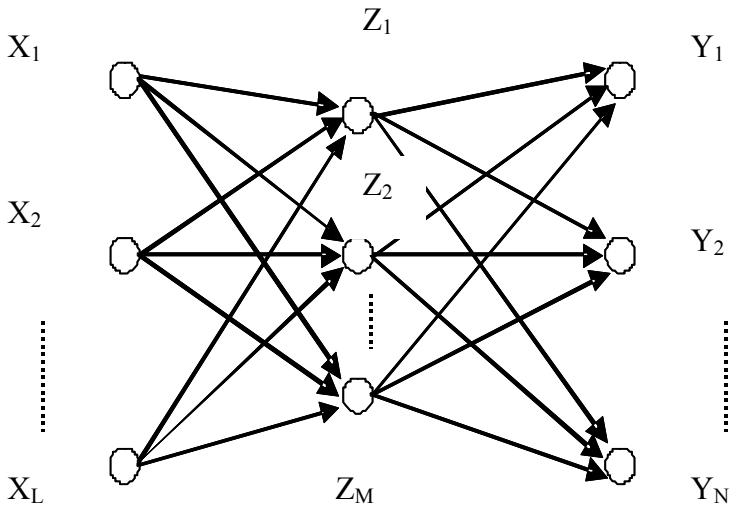


Figure 4-4 A multilayer neural network

The basic neural processing unit is shown in Figure 4-5. In most typical neural network settings, training is accomplished by presenting a sequence of training vectors, or patterns, each with an associated target output vector. Figure 4-5 is used to illustrate the basic concept, the inputs may be the inputs discussed in the previous section, or inputs to the neurons in the next layer, and the same applies to the outputs. When a sample vector X is presented, a sum computation ($w_0 + w_1 x_1 + ... + w_n x_n$, where w_0 is the initial weight, $w_1,...,w_n$ are the weights from each input to this neuron) is made at each neuron (only one neuron is shown in the Figure), which then applies its activation function *f(x)* (described in 4.5.3) to compute its output signal (here only one output is shown). The number depends on the application. The weights are then adjusted according to a training algorithm (described in Appendix A). This process is known as supervised training (which is the one used in this research).

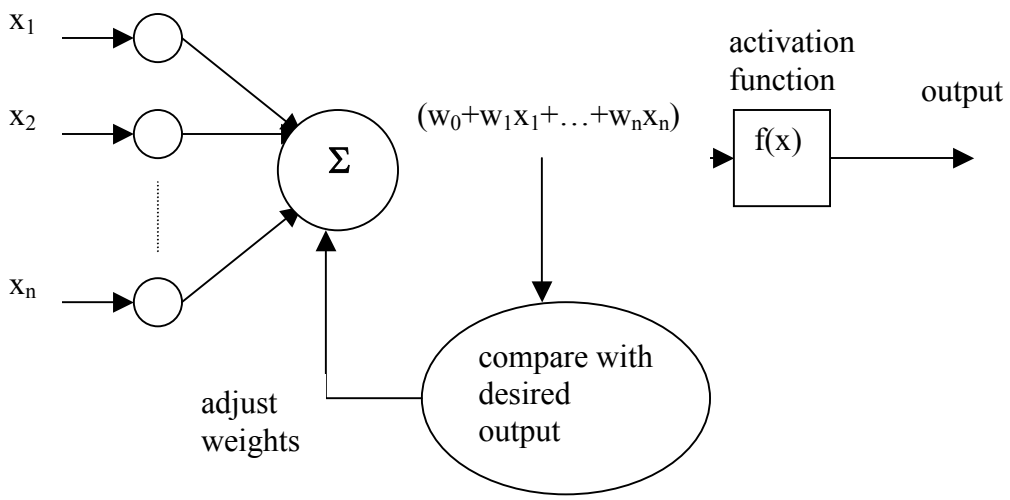


Figure 4-5 Neuron processing unit

4.5.2 Training algorithm

An algorithm is a method of determining the values of the weights. This process is also called training and is an important distinguishing characteristic of different neural networks. There are two types of training: supervised and unsupervised.

The fundamental principle underlying the perceptron learning[4] algorithm is to modify weights in such a manner as to reduce the number of misclassifications. For example, Root Mean Square (RMS) is used to verify the number of misclassified samples.

Neural networks can be trained to perform a non-linear mapping from an *n*-dimensional space of input vectors (*n*-tuples) to an *m*-dimensional output space. In the case of unsupervised training, a sequence of input vectors is provided, but no target vectors are specified. The neural network modifies the weights so that the most similar input vectors are assigned to the same output (or cluster) unit. The neural network will produce an exemplar (representative) vector for each cluster formed.

[4] The perceptron is a program that learn concepts, i.e., it can learn to respond with True (1) or False (0) for inputs that present to it, by repeatedly "studying" examples presented to it.

A "backpropagation" training algorithm is chosen in this research; and supervised training is used. The backpropagation algorithm is explained in detail in Appendix A

A major issue in the design of backpropagation architecture is in choosing the number of hidden layers/hidden neurons. Since there is no parametric/theoretic guidance available, the design has to be based on a heuristic approach [Xuan98]. In the applications addressed in the later chapters, the optimum number of layers/hidden neurons was determined by comparative cross-validation amongst several neural networks.

4.5.3 Activation function

As mentioned earlier, the basic operation of an artificial neuron involves summing its weighted input signal and applying an output, or activation function. For the input units, this function is the "identity" function. In most cases, a nonlinear activation function is used. In order to achieve the advantages of multilayer nets, compared with the limited capabilities of single-layer nets, nonlinear functions are required. (The result of feeding a signal through two or more layers of linear processing elements is no different from what can be obtained using a single layer.)

The activation function for a backpropagation network should have several important characteristics [Fausett94]: it should be (i) continuous, (ii) differentiable, and (iii) monotonically non-decreasing. Furthermore, for computational efficiency, it is desirable that its derivative be easy to compute. One of the most typical activation functions that a backpropagation neural network uses is called the binary sigmoid function, defined as:

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{with} \quad f'(x) = f(x)[1 - f(x)]$$

The function is illustrated in Figure 4-6.

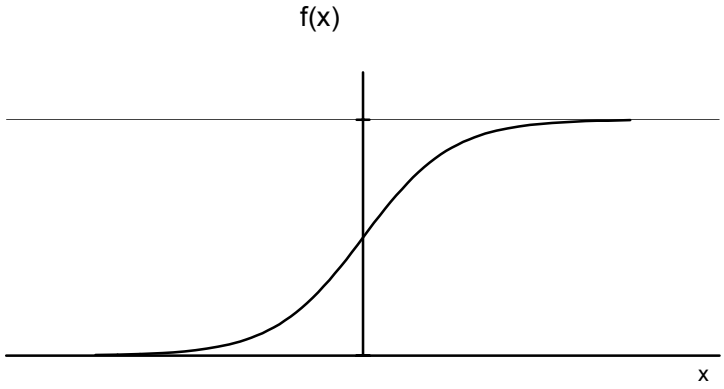


Figure 4-6 Binary sigmoid, range (0,1)

4.5.4 Data pre-processing

Neural networks can perform essentially arbitrary non-linear functional mappings between sets of variables. For most applications, it is necessary first to transform the data into some new representation before training a neural network. For most practical applications, data pre-processing is often one of the most important stages in the development of a solution, and the choice of pre-processing steps can often have a significant effect on generalisation performance [Bishop95]. It generally is convenient to process the whole training set using the pre-processing transformations, and then use this transformed data set to train the network.

A simple reason for the need for pre-processing data is that often different input variables have typical values that differ significantly. In a system monitoring a chemical plant, for instance, two of the inputs might represent temperature and pressure respectively. Depending on the units in which each of these is expressed, they may have values that differ by several orders of magnitude. Therefore, without pre-processing, just using the raw data, the neural network may not be trained. The application of a neural network in this thesis happens to be such a case: the two input values differ by at least 2 magnitudes. Hence, all the application of neural networks in Chapter 5, 6 and 7 have been pre-processed before training takes place.

There are several approaches to pre-processing [Bishop95]. Considering the data that have been used, a simple normalisation is used in this research, i.e. a sequence of any

input, such as cell arrival rate, (this is described in detail in 5.3 and 6.3) is divided by its maximal value. Thus the transformed data falls into (0,1][5]. Through this simple normalisation, all the inputs fall into the range of (0 1] and are of the same magnitude.

4.6 Applying neural network in the HQ model

General speaking, the application is divided into two parts: (i) collecting the training data, which comprises several simulation runs for both the HQ and the standard simulator, and (ii) adjusting the model with the results from the neural net in order to undertake production simulation runs.

4.6.1 Training the neural network

Figure 4-7 shows the arrangement for getting training data from a real piece of hardware.[6] In the diagram, the [*X*] represent the features of the input traffic, such as the mean duration of the ON periods and the traffic load. The y is the mean inter-exit-time from a run in the router lab; the Y is the mean inter-exit-time of the results from the model. δ is the difference between y and Y. The $[X]_{1\text{-}N}$ and the δ were used as one training set for the neural network. The HQ model and the router need to be run several times to get enough training data covering a wide range of input traffic patterns. Instead of using router hardware, it would be possible to use a standard and detailed simulation to represent the "real" model.

[5] (a, b] denotes range a – b, excluding a but including b.

[6] This experimental work was performed by the author in Nortel Networks' router lab at the Harlow site.

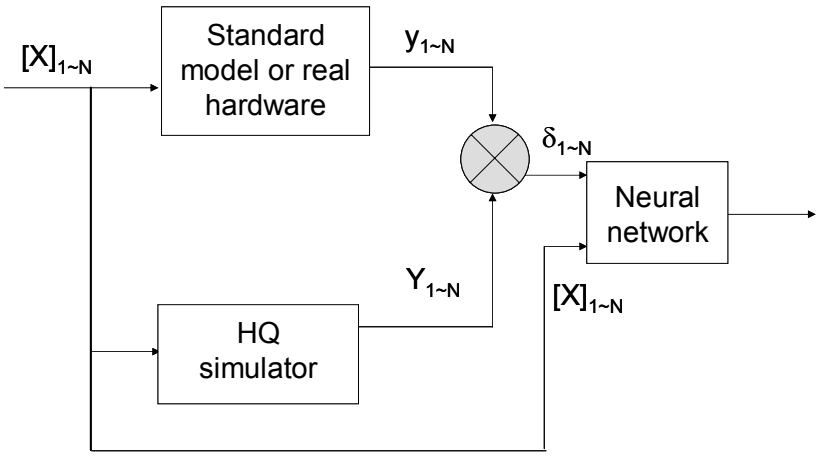


Figure 4-7 Training the neural network

4.6.2 Proposed HQ model

Once the neural network is trained, a new input traffic set can be simulated and the neural network will predict the inherent difference of the inter-exit time between the real network and the HQ model. The new traffic set must be within the range of the training data. For example, the training data covered the offered load from 0.1 to 0.9, and the new traffic parameters are chosen within this range. Using this predicted difference to fine-tune the service time of the buffer provides maximum accuracy. Figure 4-8 shows the overall structure.

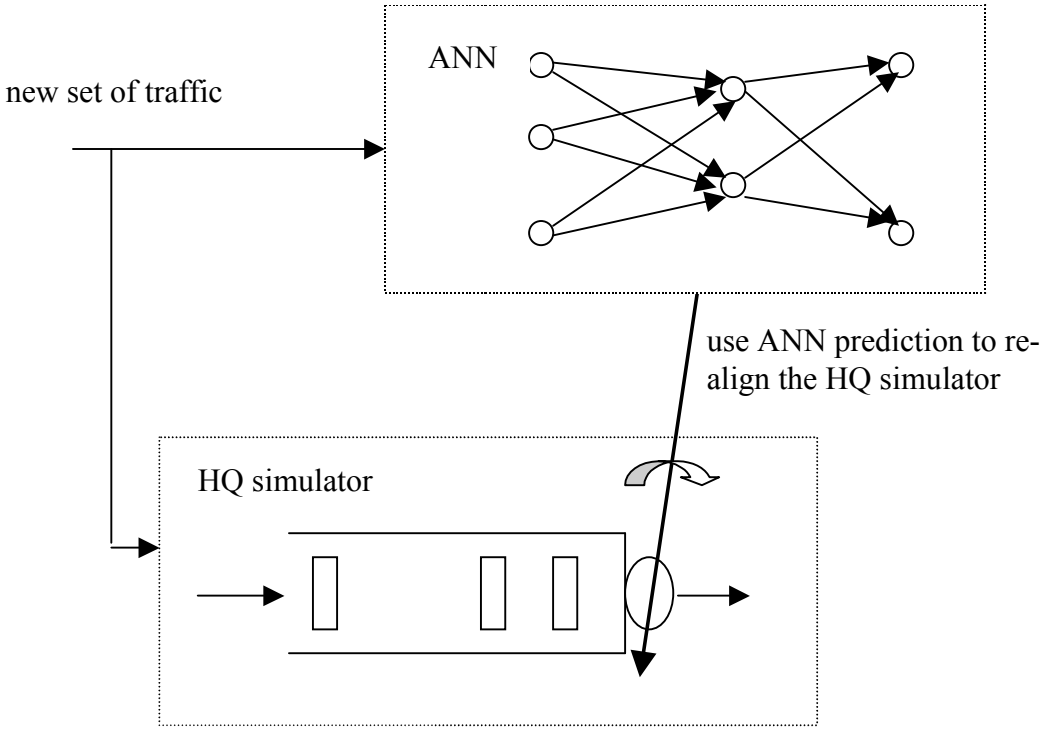


Figure 4-8 Structure of the HQ model

Using neural networks to predict the difference will resolve any inaccuracies of the service time from the formula ((5.8), (5.12) and (6.15)) and will take into account other lower-layer effects, such as the effect of the MAC layer. This is illustrated later.

5. HQ approach used in cell-scale queuing

In this chapter, the complete HQ approach (the speed-up and the re-alignment) is applied to cell scale queuing. A service rate formula for a FIFO queue is considered first, and then for a priority queue.

5.1 Evaluation of the service rate in a single queue

The queuing model comprises a traffic input, a buffer and a server. In a normal ATM simulation, the traffic input is assumed to consist of N Poisson sources; here, one of the sources is assumed to be the foreground traffic and the rest are the background traffic. The service time for each cell is fixed so that the actual queuing model is M/D/1, which is treated as a special case of the M/G/1 queue. In the proposed abstracted model, as described before, only the foreground traffic is simulated, but the service time is no longer fixed, so the queuing model is assumed to be M/M/1. This assumption is a simple approximation, and it proved to be accurate in the validations later in this chapter. The basic idea is to use the abstracted model to represent the real cell stream in an ATM switch. Therefore, what is needed is to create an equivalence between the two queuing systems by parameterisation as appropriate.

5.1.1 Analysis of the queueing models

5.1.1.1 Decay rate for M/D/1 queue

Based on the excess-rate (ER) method, which was introduced in [Schormans94], and briefly described in Chapter 3, the analysis depends on observations of how the queue changes in size: for every excess-rate cell, the queue increases by one; a single cell arrival when one cell is being served causes no change in the queue state. The queue decreases when there are no cells arriving in any time slot.

For each simulation buffer, define:

X = buffer length in both accelerated and original systems

η_{MD1} = decay rate of the state probabilities for M/D/1

η_{MM1} = decay rate of the state probabilities for M/M/1

λ = average arrival rate of the input to the queue, in arrivals per time slot

ρ = utilisation of the queuing system

a_k = Pr(k arrivals in a service time)

λ_f = average arrival rate for the foreground traffic to the queue

S = mean service time

[Schormans97] made use of the ER concept and altered the Poisson arrival process to produce a geometric series for the tail of the distribution, giving a geometrically distributed approximate Poisson process (GAPP) [Schormans96]. By analysing the number of cell decreases in the queue between ER arrivals, [Schormans97] gave a general formula of decay rate for any M/G/1 queue as:

$$\eta_{MG1} = \frac{a_1(1-\rho)+\rho-1+a_0^2}{a_0(\rho-1+a_0)} \tag{5.1}$$

and in addition, if the service time is fixed, which is the case with M/D/1 and it is a special case of M/G/1, the items a_0 and a_1 are:

$$a_0 = e^{-\rho} \tag{5.2}$$

$$a_1 = \rho e^{-\rho} \tag{5.3}$$

Substituting (5.2) and (5.3) into (5.1), then the decay rate for M/D/1is:

$$\eta_{MD1} = \frac{\rho \cdot e^{-\rho}(1-\rho)+\rho-1+e^{-2\rho}}{e^{-\rho}(\rho-1+e^{-\rho})} \tag{5.4}$$

5.1.1.2 Derivation for the service rate

The equivalence between the original and accelerated queuing systems is achieved by forcing equivalence between the decay rates of the state probabilities. For the M/M/1 queue, from [Pitts01],

Pr{system size > x} = ρ^{x+1}

From [Schormans99B], using decay rate,

Pr{cell exceeds x} = η^{x+1} , where η is the decay rate

Therefore,

$$\eta_{MM1} = \rho_{MM1} \tag{5.5}$$

Equating with the original M/D/1 queue leads to:

$$\rho_{MM1} = \eta_{MD1} = \frac{\lambda \cdot e^{-\lambda} \cdot (1-\lambda) + \lambda - 1 + e^{-2\lambda}}{e^{-\lambda}(\lambda - 1 + e^{-\lambda})} \tag{5.6}$$

So to parameterise the service time distribution, it is simply necessary to determine the mean service time:

$$\rho_{MM1} = \lambda_f.S \tag{5.7}$$

Therefore, the formula for the average service time is:

$$S = \frac{\lambda \cdot e^{-\lambda} \cdot (1-\lambda) + \lambda - 1 + e^{-2\lambda}}{\lambda_f \cdot e^{-\lambda} \cdot (\lambda - 1 + e^{-\lambda})} \tag{5.8}$$

The S will be used as an average service time in simulating the foreground traffic only, with respect to the effect of the background traffic on the foreground stream. λ is the total traffic arrival rate, and λ_f is the foreground traffic arrival rate.

5.1.2 Validation of the formula

Validation is done in two parts: (i) using cell loss probability (CLP) as a comparison criterion and comparing with the results of the M/D/1 queue; (ii) using inter-exit time (IET) as a comparison criterion against a conventional detailed simulation model,

which simulates so-called background as well as the foreground traffic. This simulation has already been validated.

5.1.2.1 Validation against the theory using cell loss probability

The accuracy of the HQ method is checked by plotting the state probabilities from both the HQ model and original model.

Three multiplexing cases are considered: (i) 234 voice sources ($\rho \approx 0.1$) (called original in the Figure), (ii) 1171 voice sources ($\rho \approx 0.5$), and (iii) 2109 voice sources ($\rho \approx 0.9$). The foreground traffic is a single 64kbit/s (167 cells/second) cell stream; the background traffic consists of the remaining voice sources. Figure 5-1 gives the comparison results. The accelerate models are models using HQ approach.

It is clear that the accelerated simulation that using HQ approach is an extremely good approximation.

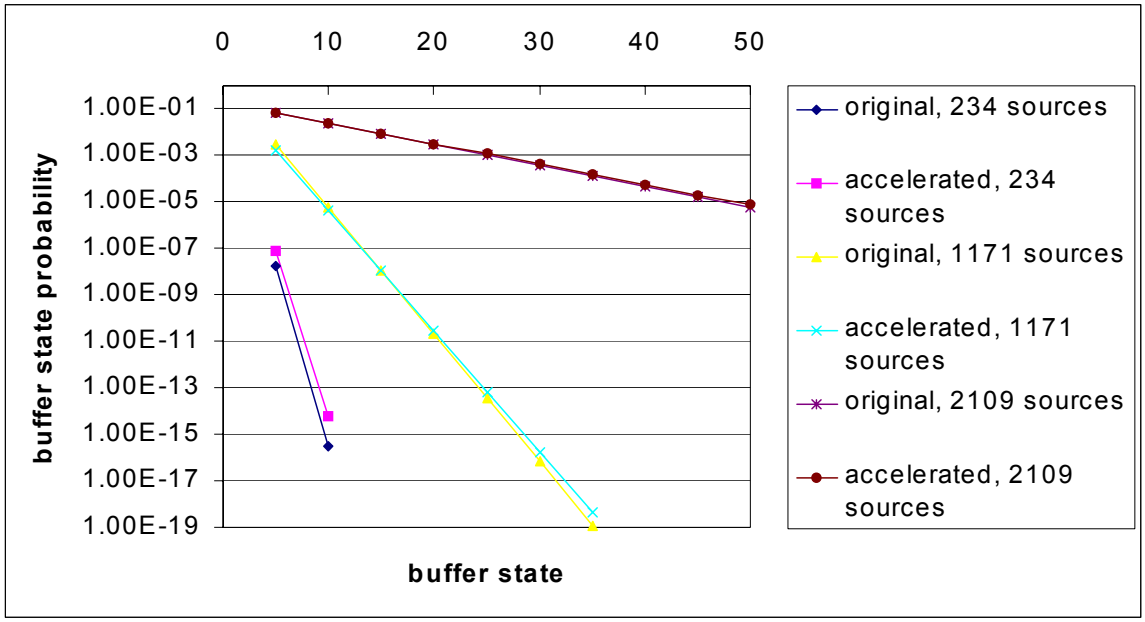


Figure 5-1 Comparison of original buffer model with the HQ model

5.1.2.2 Validation against conventional model

While the previous section validated the *approach* against the theory, it is more significant to see how well the simulation works when compared with a real network, or network element.

To do this, a software testbed is used. This is also a simulation, called the "BBS model". It is a detailed ATM switch model that emulates a Nortel Networks product. The BBS model is a Nortel internal research product and has been used to model specific Nortel equipment in a Lab environment. It was already fully tested and validated. Because this research was supported under the TCS scheme, the author has permission to use the BBS model. To allow a fair comparison, both simulations are built in a commercial simulation tool called OPNET.

OPNET is a commercial product that has been widely used in the network level simulation. It is a discrete event driven simulator that allows the user to specify the system through a graphical interface in terms of decreasing levels of abstraction from the network level down to the individual process level. The top level (it depends on the application; here in this research, it is the network level in OPNET's structure) is where the topology of the simulated network model is defined; the interconnection within the network (e.g. ATM links) is also defined at this level. The next level below the network level is called the node level, where the elements that make up the network nodes and their interconnections (e.g. Ethernet links, or PPP links) within the network are defined. At the next level, these elements are defined in detail by the user. The elements include queues, processes, sources, receivers and transmitters. The functionality of each process or queue element is defined in terms of a finite state diagram and the transition between states. The lowest level is where the processing in each of the states in the finite diagram is coded in so-called Proto-C.

The configuration of the test for a single queue is shown in Figure 5-2. Basically, there are three parts: (i) the traffic generator, where there are N traffic sources of Poisson traffic being generated; (ii) the queue (which includes the buffer and a server)

and in this test, there is only one queue, so that priority is not considered; (iii) the data collector, which collects the data (in this case, it is the inter-exit times for the cells) and then discards the cells.

In the case of the HQ model, as described previously, only the foreground traffic is generated in the source model, for example, source *j*; in the case of the testbed, which is a full simulation, all the foreground and background traffic are generated and simulated, so the number of traffic sources is *N*.

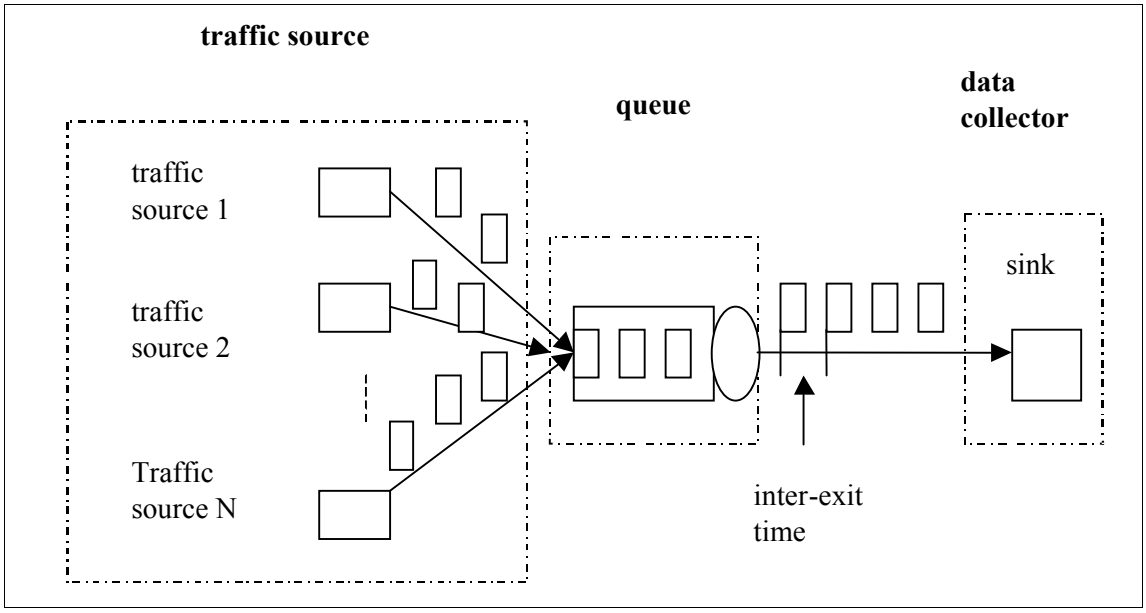


Figure 5-2 Test configuration for the single queue scenario

The inter-exit-time was used as the basis of comparison between the two simulations. The reason for choosing IET is that it represents the behaviour of the network better than a single parameter such as Cell Loss Probability. [Mitchell00] also used this parameter in their paper, and they called it inter-departure time. Later in this test, the results of the IET distribution will be given to show that it is acceptable to choose the *mean IET* as a comparison criterion.

Table 5.1gives the traffic parameter for the test, and the results are shown in Figure 5-3. In the tests, the link capacity is 155Mbit/s, the cell size here is chosen to be 424

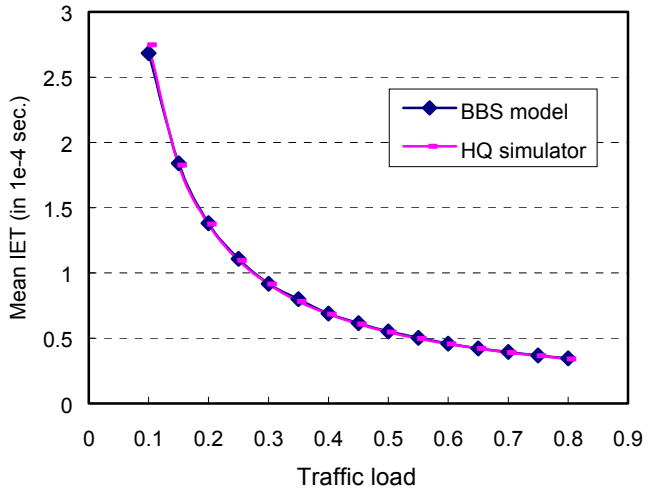
bits/cell. There are two cases with different proportions of foreground traffic. The traffic is VBR.

Table 5.1 Traffic parameters of single queue

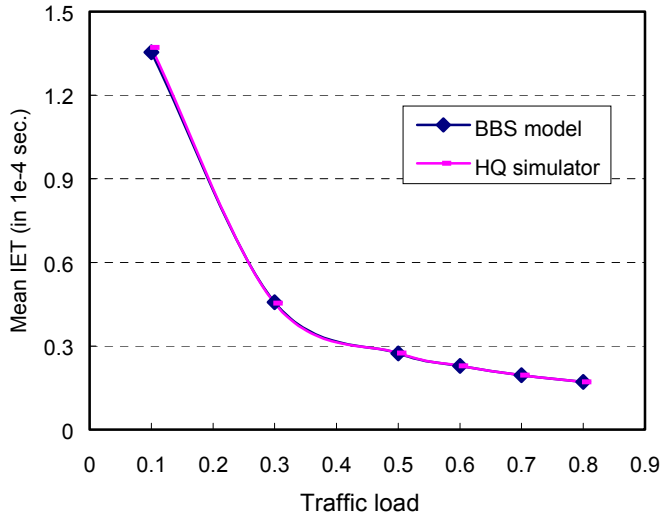
proportion of foreground traffic	10% of the total traffic		50% of the total traffic	
	min	max	min	max
utilisation	0.1	0.8	0.1	0.8
arrival rate for the foreground (cells/s)	3656	29245	18278	146226
arrival rate for the background (cells/s)	32901	263208	18278	146226

In the HQ, the service time for the foreground traffic is altered by the formula (5.8) described previously in this Chapter.

Figure 5-3 shows that the proposed model works well with different proportions of foreground traffic load while increasing the total traffic load up to 0.8.



a) Foreground load = 10% of the total load



b) Foreground load = 50% of the total load

Figure 5-3 Mean inter-exit time for both HQ and BBS models

It might be argued that using only the mean inter-exit time is not sufficient for the comparison. Other results, such as Cumulative Density Function (CDF) and Occurrence Histogram (OH) of the inter-exit times from each simulation run, have also been used for comparison. Figure 5-4 and Figure 5-5, shows that the CDF curves of our simulator and the testbed are similar. Note that these curves have been obtained by digitising picture output from OPNET directly into a spreadsheet, so that there are slight errors from the digitising process, but no more than the thickness of the line on the graph.

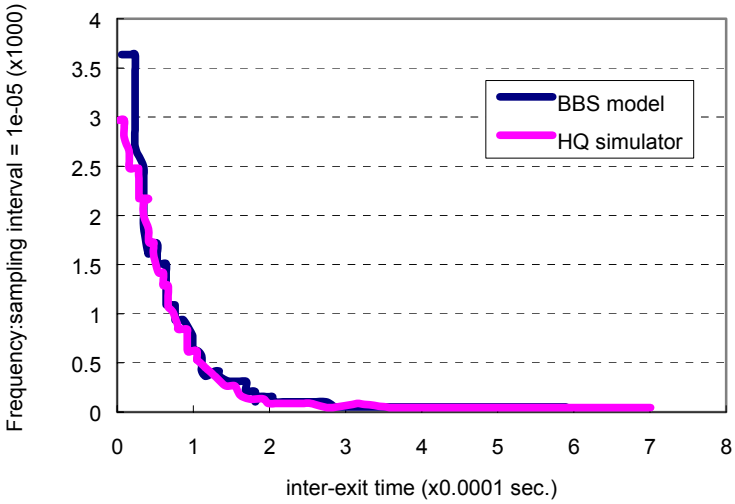


Figure 5-4 OH of BBS and HQ model in a single queue comparison

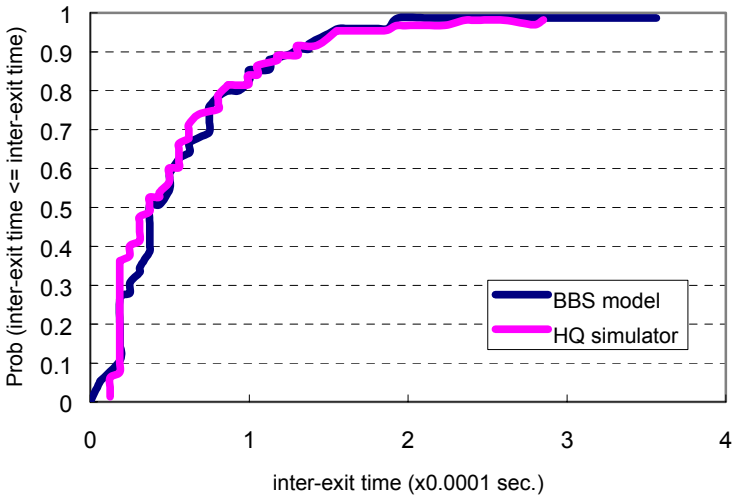


Figure 5-5 CDF of BBS and HQ model in a single comparison

These results are encouraging. The accelerated simulation does give good results when compared with the detailed model and is about 30 times faster. The work will now be extended to determine the service time for multiple queues so that it is possible to model an ATM switch where each queue represents a different Quality of Service (QoS).

5.2 Evaluation of the service rate in priority queues

5.2.1 Background

Traffic management within ATM networks has two main goals: to provide specified and guaranteed levels of Quality of Service (QoS) while efficiently using available network resources. In ATM, for each virtual connection a specific QoS can be chosen, which influences the ATM transfer performance parameters such as cell loss, cell delay and cell variation.

A definition of QoS is given in CCITT Recommendation I.350. "*Quality of Service is defined as the collective effect of service performances which determine the degree of satisfaction of a user of the specific service*". ATM traffic can be described by a three-level hierarchical model with different time scales:

(i) Connection level in which the QoS control is achieved by *connection admission control* (CAC), which decides whether a connection can be admitted on to the network before a connection attempt is made to the destination.

(ii) Burst level, this can be controlled by fast resource reservation mechanisms and adaptive flow control protocols like ABR. It determines the large buffers needed for non-real-time connections.

(iii) Cell level, which can be controlled by mechanisms such as policing, priority control or traffic shaping. At multiplexing points, the traffic pattern at the cell level determines the buffer size required for real-time connections.

Introducing priorities to different classes of packets is one way to maintain Quality of Service. [Shaikh89] [Shaikh90]. Some switch designs, for example the ATOM switch [Suzuki89], used priority scheme to support integrated services.

[Yang89] considered an integrated packet switched system with infinite buffer serving two priority classes, with high priority class packets modelled by a two-state Markov

process and low priority class modelled by a Poisson process. [Chen92] used two priority buffers in the delay and packet loss control of a packet switch with output buffering. The two types of services under consideration have different requirements on delay and loss probability: real-time packets require shorter delay while non-real-time (data) packets require smaller loss probability; hence they are given different priority buffers and two types of traffic are queued in separate buffers. The queueing discipline used there is the same as used in this thesis.

ATM networks can feature two forms of priority mechanism: (i) space and (ii) time. Both forms relate to how an ATM buffer operates. *Space priority* addresses whether or not a cell is admitted into the finite waiting area of the buffer. *Time priority* deals with the order in which cells leave the waiting area and enter the server for onward transmission. Thus, the main focus of the space priority mechanism is to distinguish different levels of cell loss performance, whereas for time priority the focus is on the delay performance. For both forms of the priority, the waiting area can be organised in different ways, depending on the specific priority algorithm being implemented.

In ATM networks, control packets that carry vital instructions for network operations are usually transmitted with higher priority than data packets. Another example would be the multi-media system in which voice and data are carried in the same network, the voice packets may be accorded a higher priority than that of the data packets owing to real-time requirements.

The focus is now on applying the HQ approach in a time priority queuing environment. For simplicity, it is assumed that there are j waiting areas, in other words, j queues, but only one server. Each queue corresponds to a priority and hence a different QoS level. The main task is still to work out the service time for the foreground traffic considering the affect of the background traffic. However, in this new scenario, there is another issue that needs to be solved: the queuing discipline.

5.2.2 Queueing discipline

There are two basic queueing disciplines for priority systems[Ng97]; namely preemptive and non-preemptive. In a preemptive priority queueing system, the service of a customer is interrupted when a customer of higher priority class arrives. This scheme is further divided into two categories: preemptive-resume system and preemptive-repeat system. If the customer whose service was interrupted begins service from the point of interruption once all customers of higher priority have been served, it is a preemptive-resume system; if the customer repeats his entire service then it is a preemptive-repeat system. In the case a non-preemptive priority system, a customer's service is never interrupted even if a customer of higher priority class arrives in the meantime. This is the way multiple queues are handled in ATM.

As shown in Figure 5-6, suppose that there are k (k = 1,2,…, n) priorities with arrivals according to a Poisson process and are served by the same server. Class 1 has the highest priority and n has the lowest. The arrival processes of each class are assumed to be independent of each other and the service process. Within each class, customers are served by their order of arrival. For simplicity, each priority corresponds to its own queue, numbered 1, 2, …, k as well. When the server is idle, it starts from the highest priority; if this queue is empty it then looks at the second highest priority queue, and continues doing this until it finds a cell in the instantaneous 'highest' priority queue. Once the 'highest' cell is found, it is served regardless of any new cell arrivals at higher priority queues while the server is busy. In other words, the server always serves the cell in the instantaneous highest priority queue.

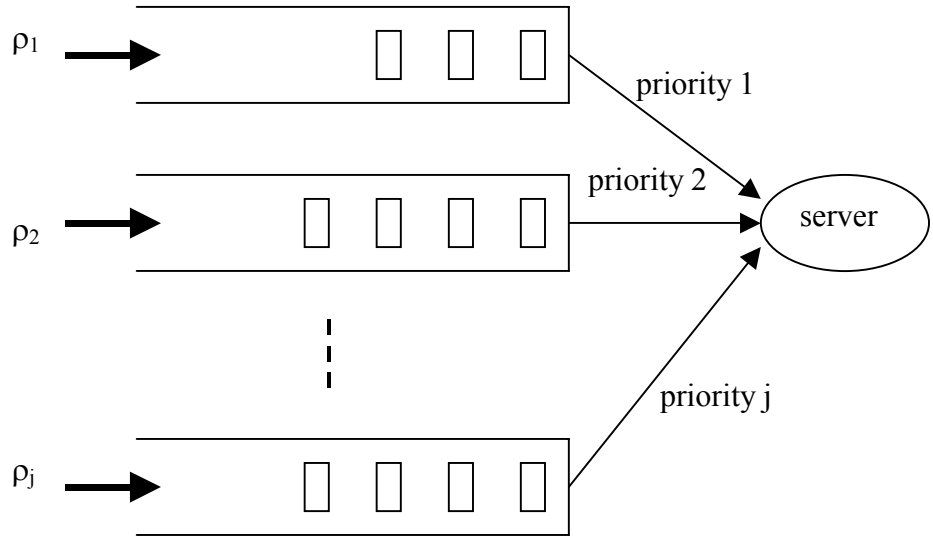


Figure 5-6 Priority scheduling

5.2.3 Service time formula for M/D/1 with priorities

Given the assumption that the traffic input processes to the buffer are all Poisson, and defining:

ρ_a = load applied by all the traffic of higher priority than *j*
ρ_j = load applied by traffic of priority *j*
λ_j = arrival rate of the priority *j*
λ_f = arrival rate of the foreground traffic with priority *j*

The M/D/1 buffer has a set service rate, which is referred to as: "original_service_rate_of_the_buffer". It is possible to approximate the non-pre-emptive priorities scheduling by a single buffer with *random* (exponential) service times (rather than fixed service time) where the randomness arises from the scheduling of the higher priority cells.

Service time S_{effj} affected by the higher priority is:

$S_{effj} = 1 / (1- \rho_1)$. (*original_service_time*) = (1 / (1- ρ_a)) / (*original_service_rate_of_the_buffer*) (5.9)

where

ρ_a = (sum of arrival rate of priority *1~j-1*) / (*original_service_rate_of_the_buffer*)

(5.10)

To parameterise the service time distribution, it is only necessary to re-evaluate the mean of the distribution, thus the formula for the average service time is:

$$S = \frac{(\lambda_j /(1-\rho_a)) \cdot (original\_service\_time)}{\lambda_j} \quad (5.11)$$

$$S = \frac{1/(1-\rho_j)}{original\_service\_rate\_of\_the\_buffer} \cdot \frac{\lambda_j}{\lambda_j} \quad (5.12)$$

where:

ρ_j = (priority j arrival rate) / (original_service_rate_of_the_buffer)

The original_service_rate_of_the_buffer is set to 155Mbit/s in the experiments.

5.2.4 Validation against conventional model

The HQ simulator using this derivation of service time to handle priorities was validated against the conventional BBS simulator, using 2 priorities. The reason for only two priorities being used is the lower priority queues only see the queues with higher priority as *one* queue, and do not care about the queues that have lower priority. Therefore, there appear to be only two queues: higher priority queue and itself. Hence, two priorities are sufficient to demonstrate the validity. The configuration is shown in Figure 5-7.

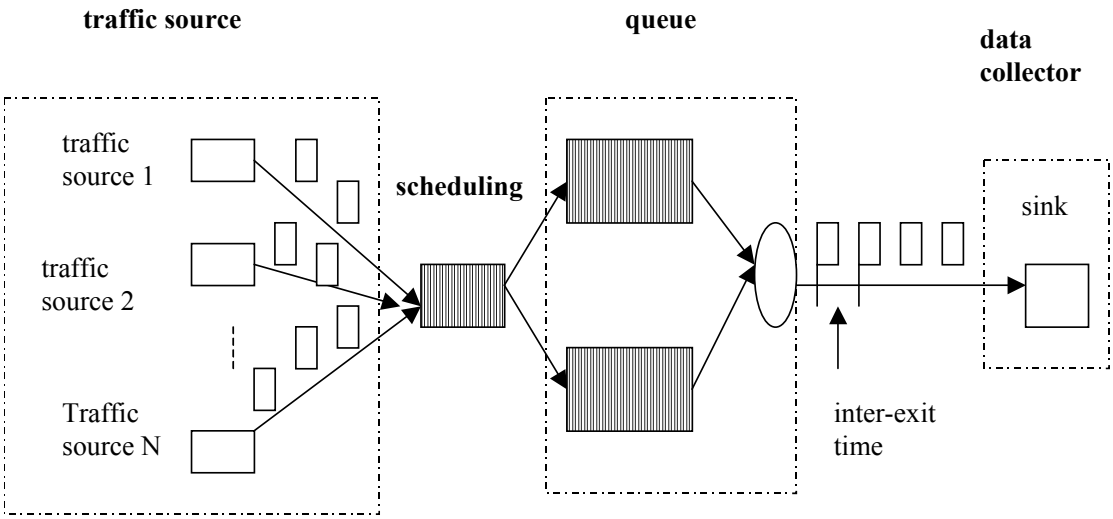


Figure 5-7 Configuration of the test for the multiple queues

In the case of the HQ model, there are two types of foreground traffic, representing the high priority and low priority traffic.

There are four parts to the configuration of the conventional model:

(i) The traffic sources generate a Poisson process of cells; this is the same as in the previous test, but the difference now is that there is high priority and low priority traffic in both models.

(ii) There is now a part called "scheduling", which is used to schedule the input cells based on their QoS value in the header of the ATM packet and put the cell in the particular queue.

(iii) The queue part now includes two queues, which represent the high priority and low priority, respectively. As with the scheduling approach, non-pre-emptive queuing is used.

(iv) The last part is the data collector, which works in the same way as the previous test: inter-exit times are collected.

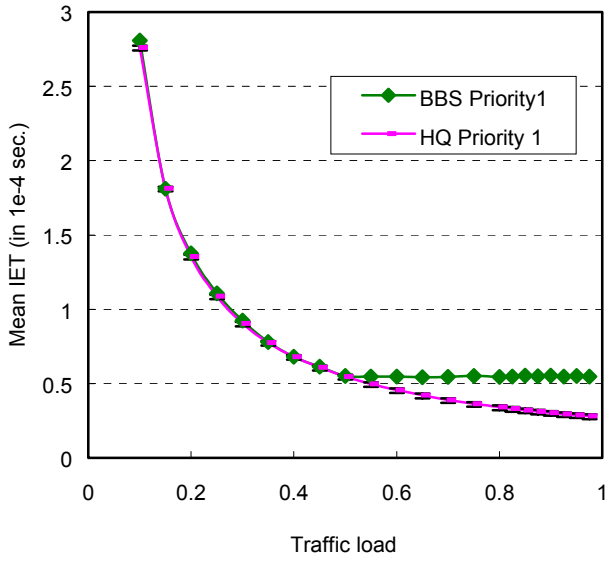
The value of the traffic parameters is given in Table 5.2. The high priority and the low priority queue use the same parameters. Figure 5-8 gives the results of the priorities 1 (high) and 2 (low), respectively. For the high priority queue, the same formula as in the single queue is used, and for the low priority queue, formula (5.12) is used.

Table 5.2 Traffic parameters of the priority queue

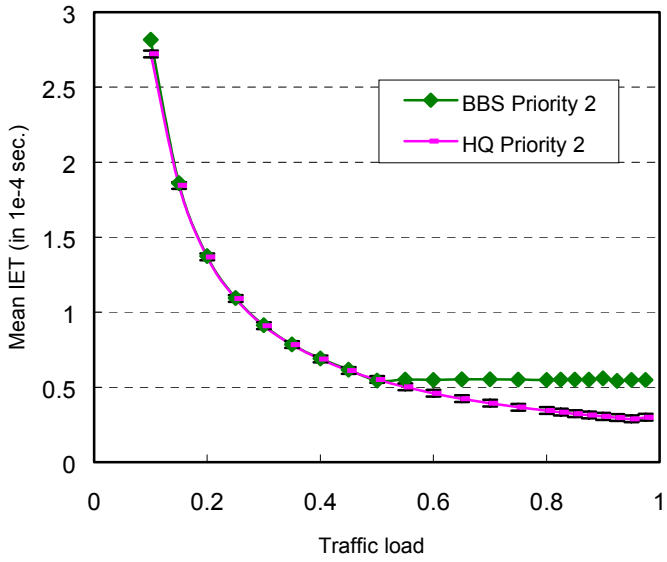
proportion of foreground traffic	Priority 1 10% of the total traffic		Priority 2 10% of the total traffic	
	min	max	min	max
utilisation	0.05	0.4	0.05	0.4
arrival rate for the foreground (cells/s)	1828	14622	1828	14622
arrival rate for the background (cells/s)	18450	131604	18450	131604

The total offered load of high and low priority in this test is chosen to be between 0.1 and 0.975. The foreground traffic in both high and low priority source is 10% of the total traffic of that particular priority.

The confidence interval with 95% confidence for the HQ model is also shown in the figures.



(a) High priority traffic



(b) Low priority traffic

Figure 5-8 Comparison results for the two priority queues

From Figure 5-8 the conclusion can be drawn that the formula for the multi-priority queues work well when traffic loads are less than 0.5, but when traffic load increases, there are some differences between the HQ model and the BBS model.

It also shows that IET of the BBS model reach a minimum value when the offered load is greater than 0.6. At first sight this appears to be wrong: from the queuing

model it would be expected that the IET values would continue to decrease, as with the results for the HQ model. However, on reflection it must be realised that the BBS model is a *model of real hardware*, not a queuing model. It is not unreasonable to expect within the hardware there to be some of spacing (or different queuing discipline) that limits the rate at which cells can be switched internally, or appear at an output. The exact process is unknown (since the internal details of the Nortel product were proprietary) but that does not matter here: what is important is that the BBS model represents the behaviour of some physical device and the whole aim of network simulation is to model a collection of such devices.

Figure 5-9 and Figure 5-10 show the OH graphs of both high and low priority of HQ model and BBS model.

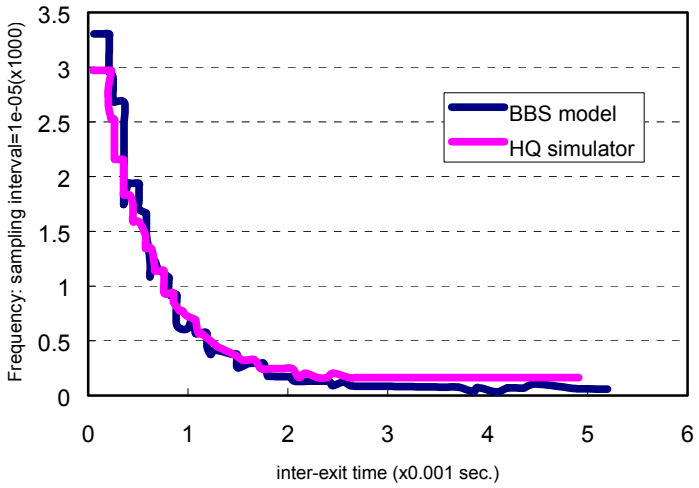


Figure 5-9 OH of the high priority queue in BBS and HQ model

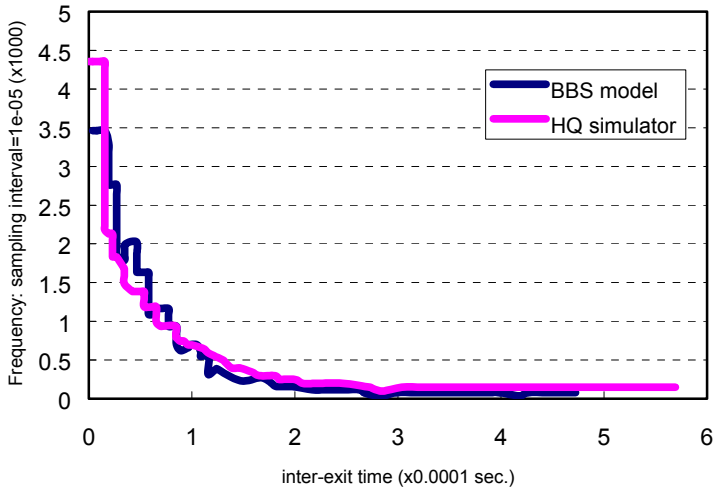


Figure 5-10 OH of the low priority queue in BBS and HQ model

Figure 5-11 and Figure 5-12 show CDF graphs of high and low priority queue of both BBS and HQ models.

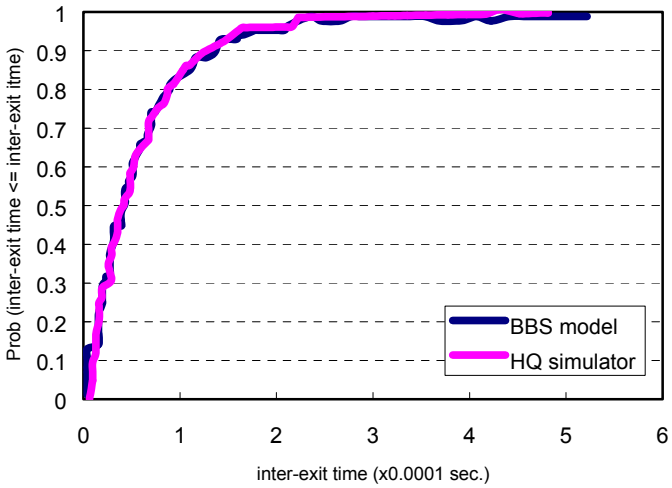


Figure 5-11 CDF of the high priority queue in BBS and HQ model

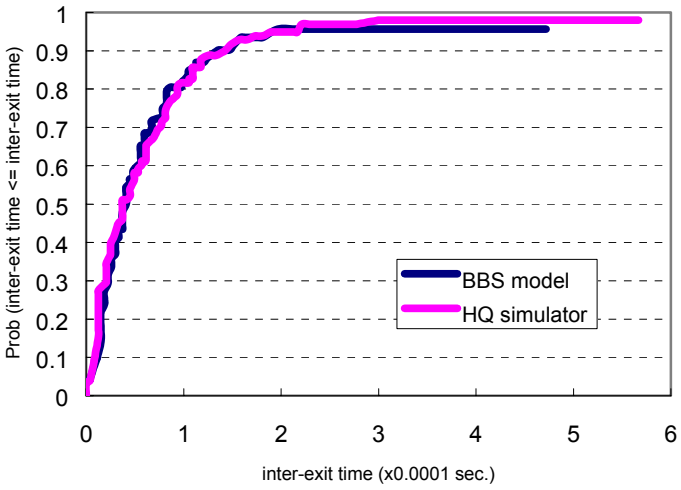


Figure 5-12 CDF of the low priority queue in BBS and HQ model

5.3 Realignment of the multi-priority model

As explained in Chapter 4, a neural network can be used to tune the parameters of the service-time in order that the HQ model better fits the situation being modelled. In this case, the intention is to see whether the neural network can make the HQ simulator fit the limiting behaviour on IET that is shown by the BBS model.

Firstly, the HQ model and BBS model ran separately in order to get training data. The inputs are offered traffic load and cell arrival rate. The output is the inter-exit time difference between the HQ and BBS model. Since there are no rules on how to build a neural network, (i.e. the number of inputs/outputs and hidden layers), this was done experimentally. In this test, the number of neurons at input and output was determined by a number of trials (of the order of 5-8 trials). This 2-8-1 network structure is found the best in performance in this particular case, fast in convergence and small in RMS, whilst the rest tried structures are not good in either convergence or accuracy. Some of them even could not converge with a given RMS.

The activation function is chosen to be the binary sigmoid function.

Figure 5-13 displays a plot of the epoch versus RMS training error. The graph is useful in determining if the neural network is learning properly, and when learning is completed. The epoch is equivalent to one complete sweep of the training data set through the network. RMS is calculates by $\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$, and is used to show output error between the currently trained neural network and the training sets. As can be seen, the neural network output error rapidly converges to the desired level, which is a pre-defined RMS error margin. The training stopped when the prior defined RMS error is reached.

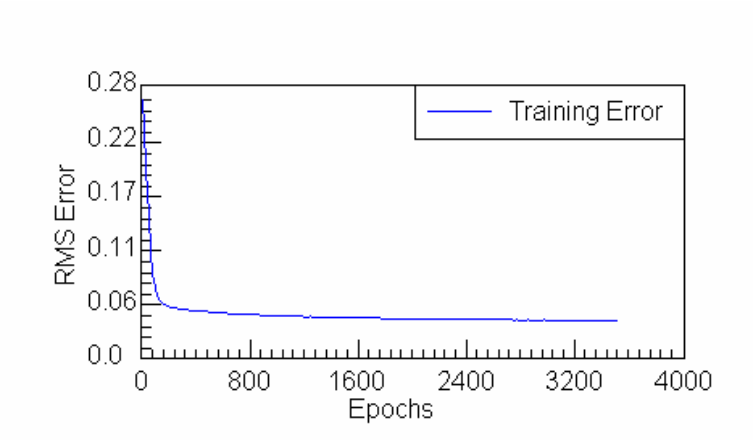
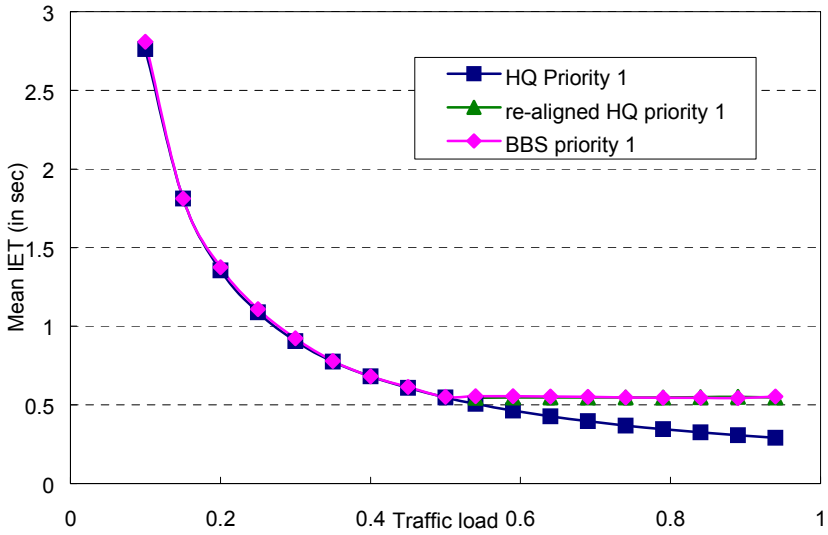
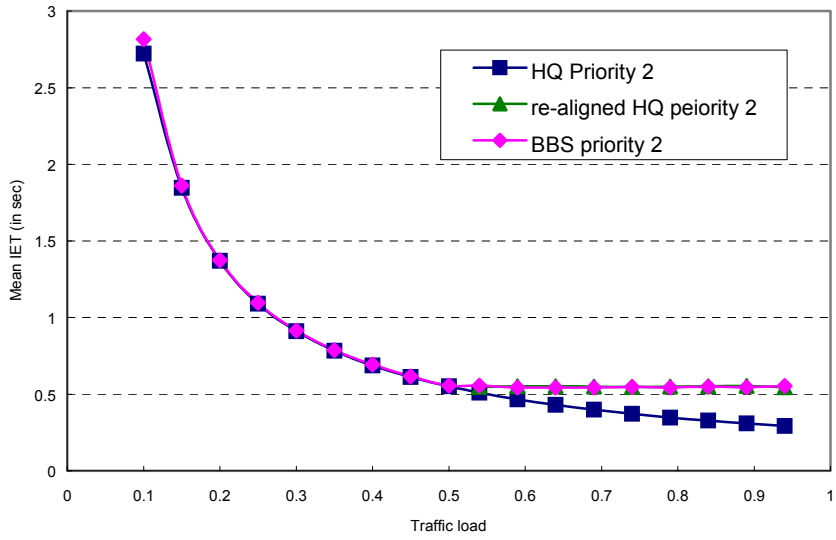


Figure 5-13. RMS error vs. training time

Figure 5-14 shows a comparison of mean inter-exit time obtained from the BBS model, with the neural network-based adjusted HQ model and with the original HQ model for the ATM node simulation. Figure 5-15 gives the enlarged figures. Based on the neural prediction, the adjustment of the HQ model is made by re-aligning the approximate service time associated with the particular traffic input. It is clear that the neural network has learnt how to adjust the model parameters so that it now acts the same as the BBS model, simulating the saturation behaviour of the real switch at high load. *This means that the HQ model has learnt how tore-align the queueing behaviour of this particular Nortel switch*.

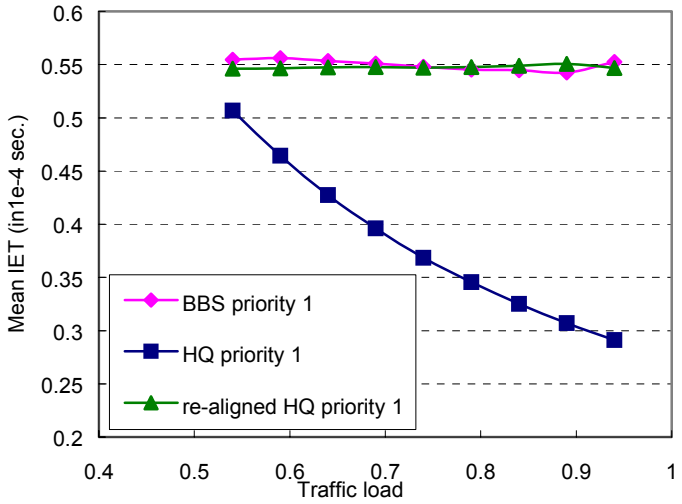


(a) High priority traffic

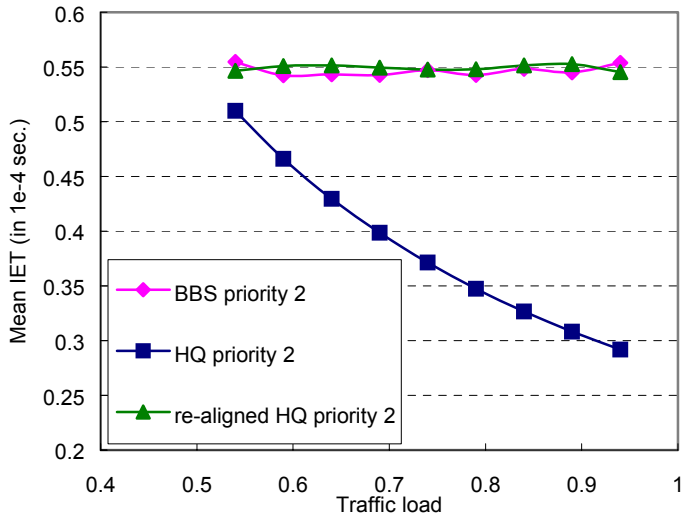


(b) Low priority traffic

Figure 5-14 Re-aligned HQ model



(a) High priority traffic



(b) Low priority traffic

Figure 5-15. Adjusted HQ model – Enlarged Figure

6. HQ approach used in burst-scale queuing

In this chapter, the HQ approach is applied for simulation using burst-scale queuing. The input traffic model is assumed to be ON-OFF in order to model the burstiness of the traffic. The service time formula is derived, and a simulation model is built and adjusted using neural network.

6.1 Bursty traffic

As data traffic becomes dominant in networks, the overall traffic appears burstier than it was in the past. This chapter considers how the accelerated simulation can be applied to such bursty data traffic.

It should be noted that some authors [Gallardo99], aiming at accelerating the simulation of long-range dependent (LRD) traffic in ATM networks, employed the fact that a large number of sources aggregates into a particular different kind, i.e. overlapping Pareto processes become Fractional Brownian Motion (FBM) if the number of sources is large enough. It is a particular advantage of the method described here that it does not rely on such aggregation of a large number of sources and can apply equally well to a small number of sources.

In this chapter, an approximation of two state ON-OFF sources is used to model the burst scale queueing, so as in [Schwartz96] and [Pitts01]. The HQ approach is applied to N ON-OFF sources that are used to model the burstiness of the traffic. Section 6.2 explains the assumptions, and 6.3 derives the service time formula. Section 6.4 validates the approach in several scenarios.

6.2 Analysis for burst scale traffic

A bursty source is usually characterised by alternating periods of inactivity and activity. By way of illustration, Figure 6-1 illustrates the burstiness of some B-ISDN applications and their peak rate [Onvural93].

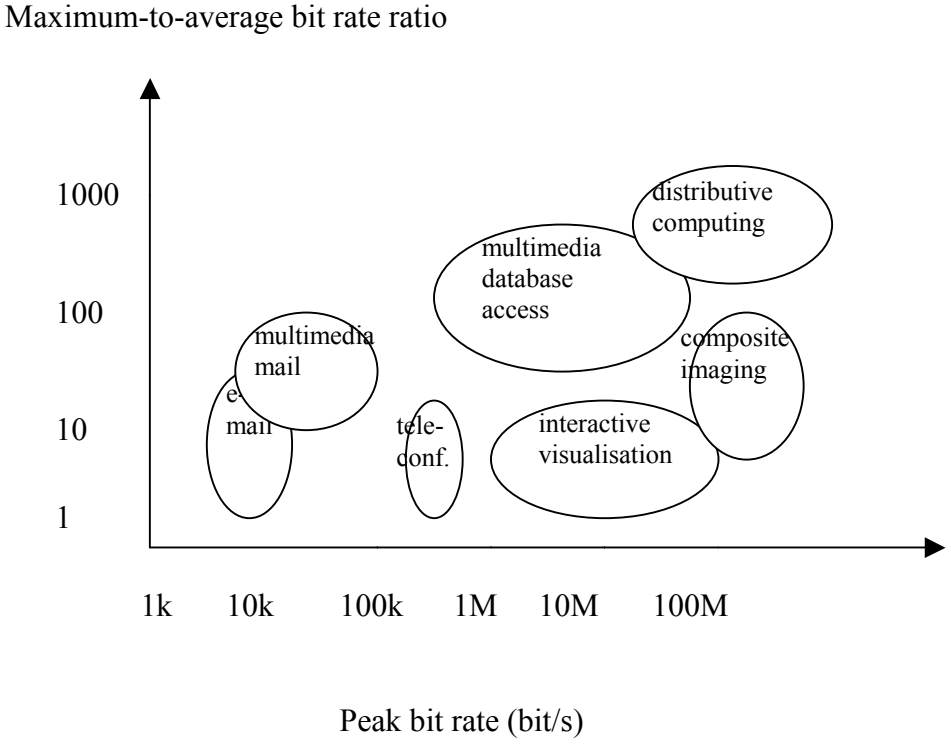


Figure 6-1 Busrstiness of some B-ISDN services (adapted from [Onvural93])

The specific Markov model of bursty traffic is normally chosen to be ON-OFF [Schwartz96], and in this thesis, it is assumed that *N* independent but statistically identical IP sources feed into a router, of which each has a *constant* packet rate of '*r*' during ON periods, and a rate of zero during OFF, as shown in Figure 6-2. Such a traffic pattern basically matches the CBR traffic stream of services that have gaps in activity (e.g. telephony with silence detection). The ON and OFF periods have exponential distributions, so do the service times.

The ON-OFF sources are characterised by (i) the peak bit rate (ii) the activity factor, and (iii) the mean ON duration. The activity factor is defined as the ratio of the mean ON duration to sum of the average ON and OFF duration.

6.3 Derivation for the formula of the increased service time

Decay rate is again being used to force equivalence of the HQ model and the conventional model. We start with finding the decay rate for the aggregated traffic.

6.3.1 Decay rate of the HQ model

6.3.1.1 HQ model

When *N* ON-OFF sources are aggregated into a buffer, the cells from different individual sources interleave, but only the cells from the foreground source are of interest. For simulation speedup purposes, this source is again created as the unique source fed into the buffer, which is equivalently served with an exponentially distributed service time. This unique traffic is again treated as foreground traffic and in the HQ method, it is the only traffic being simulated. The service rate of this server is reduced in order to mimic the effect of the background traffic, i.e. the effect of the background traffic is accommodated via an approximate formula. The approximation is therefore equivalent to simulating only one of the *N* input sources and increasing the mean packet service time (decreasing the service rate) accordingly, while forcing the cell service time to become exponential.

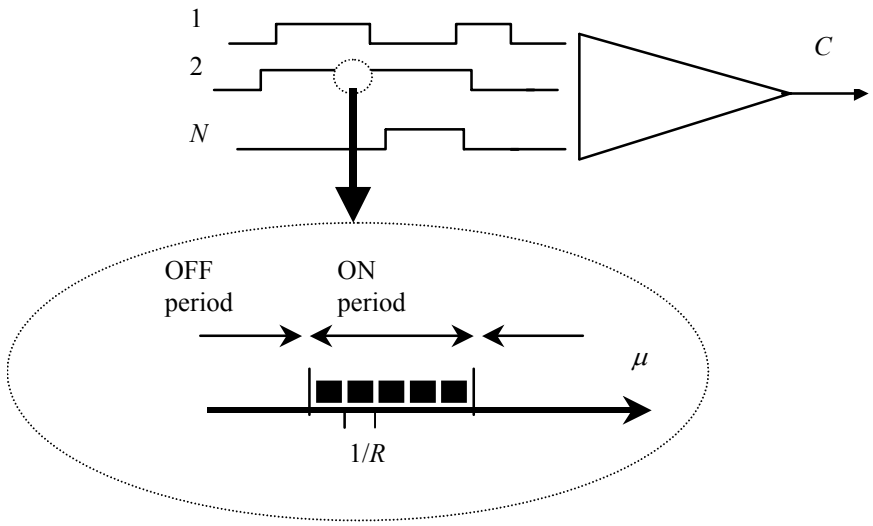


Figure 6-2. *N* ON-OFF sources aggregating into a buffer

It is assumed here that the arrival rate during the ON periods is constant; with inter arrival time equal to 1/R. The foreground traffic can be represented as a G/M/1 queue model equivalently, where G indicates an input process with an arbitrary interarrival time.

6.3.1.2 Derivation of the formula for the increased service time

Define:

X = the buffer length

ρ_f = load of the foreground traffic

ρ_b = load of the background traffic

ρ = the total traffic load, and equals $\rho_f + \rho_b$

T_{ON} = mean duration of the OFF time periods

T_{OFF} = mean duration of the ON time periods

μ = reduced service rate of the buffer for foreground traffic in packets per time unit

R = cell arrival rate of the single ON-OFF source during the ON periods

C = link capacity

For G/M/1 queue, [Schormans99A] presented a generic formula to calculate the decay rate:

$$\eta_{GM1} = \frac{D_0}{1 - D_0 - D_1 + qD_0} \tag{6.1}$$

where D_k = Pr {'k' departures between arrivals} and q is given by:

$$q = 1 - \frac{1 - D_0 - D_1}{1/\rho - 1 + D_0} \tag{6.2}$$

Now, it is only necessary to get D_0 and D_1 in order to complete the formula.

An ON-OFF source fed into a buffer can be illustrated in Figure 6-3.

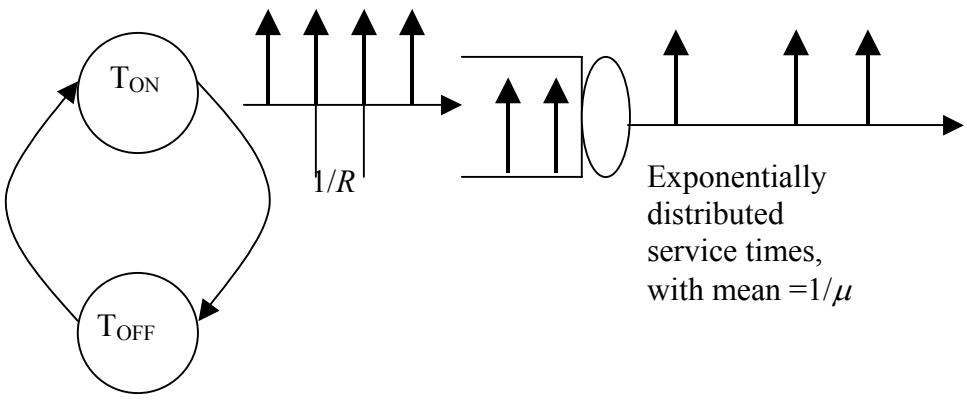


Figure 6-3 Source model and buffer diagram

To take a closer look at the traffic arrivals, consider Figure 6-4.

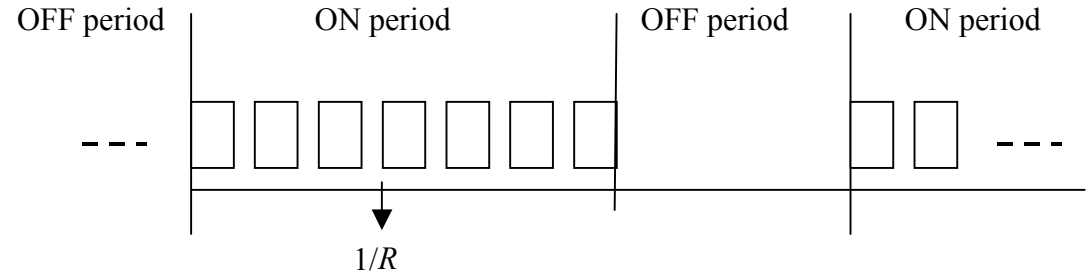


Figure 6-4 ON-OFF series

Define $P[k]$ = Pr {inter arrival time between packets = k time units}

As shown in Figure 6-4, the packet interval time during the ON period is $1/R$. Only one packet interval is different, which is the interval between the last packet of the previous ON period and the first packet in the current ON period. It actually is the interval of a particular OFF period. Therefore, $P[k]$ distribution is in two parts: a probability mass at $P[k=1/R]$, which dominates the packet intervals in the ON period; and a weighted exponential deriving from the exponentially distributed length of the OFF periods. Thus:

$$P[k=1/R] = 1 - \frac{1}{T_{ON} \cdot R}$$

$$P[k \text{ is exponentially distributed with mean } T_{OFF}] = \frac{1}{T_{ON} \cdot R}$$

Using the Law of Total Probability,

$$D_0 = (1 - \frac{1}{T_{ON} \cdot R}) \Pr\{\text{zero departures between arrivals in the ON period}\} +$$
$$\frac{1}{T_{ON} \cdot R} \Pr\{\text{zero departures between arrivals in the OFF period}\}$$

and in this formula for D_0,

$\frac{1}{T_{ON} \cdot R} \to 0$, leaving:

$D_0 = \Pr\{\text{zero departures between arrivals in the ON period}\}$, then,

Using the queue model G/M/1,

$$D_0 = e^{-\frac{\mu}{R}} \tag{6.3}$$

Equally for D_1,

$$D_1 = (1 - \frac{1}{T_{ON} \cdot R}) \Pr\{\text{one departure between arrivals in the ON period}\}$$
$$+ \frac{1}{T_{ON} \cdot R} \Pr\{\text{one departure between arrivals in the OFF period}\}$$

again, in this formula for D_1,

$\frac{1}{T_{ON} \cdot R} \to 0$, leaving:

$D_1 = \Pr\{\text{one departures between arrivals in the ON period}\}$,

$$D_1 = \frac{\mu}{R} e^{-\frac{\mu}{R}} \tag{6.4}$$

Then, substitute (6.2), (6.3) and (6.4) into (6.1),

$$\eta_{GM1} = \frac{e^{-\frac{\mu}{R}}}{1-e^{-\frac{\mu}{R}} - \frac{\mu}{R}e^{-\frac{\mu}{R}} + (1 - \frac{1-e^{-\frac{\mu}{R}} - \frac{\mu}{R}e^{-\frac{\mu}{R}}}{1/\rho - 1 + e^{-\frac{\mu}{R}}}) \cdot e^{-\frac{\mu}{R}}} \tag{6.5}$$

Rearrange to get:

$$\eta_{GM1} = \frac{1/\rho - 1 + e^{-\frac{\mu}{R}}}{e^{\frac{\mu}{R}}(1/\rho - 1) + (\mu/R)(1 - 1/\rho) + e^{-\frac{\mu}{R}}} \tag{6.6}$$

6.3.2 Decay rate of the original traffic

The original multiplex of *N* ON-OFF sources with fixed length service time is shown in Figure 6-5.

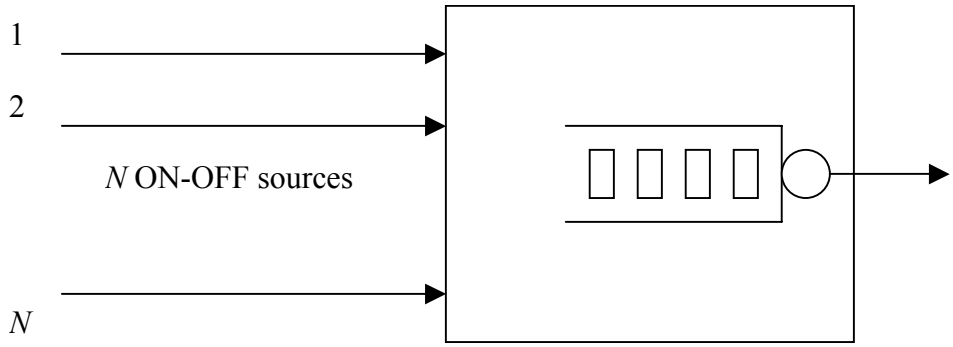


Figure 6-5 Access multiplexor

In [Nyong98], it was noted from data collected via studies of a real network, that these ON-OFF models are accurate as well as simple. To parameterise the simulation model for multiplexed ON-OFF sources, from [Schormans99B], define:

$T_{(ON)}$ = mean time the aggregate input process spends in ON state

$T_{(OFF)}$ = mean time the aggregate input process spends in OFF state

C = the output rate of the queue in cells/second

R_{ON} = mean input rate into the queue when the aggregate input process is in the ON state

R_{OFF} = mean input rate into the queue when the aggregate input process is in the OFF state,

Using briefly described discrete fluid-flow method, [Schormans99B] gave an equation for the decay rate of N ON-OFF aggregated traffic:

$$\eta_{N(ON-OFF)D1} = \frac{a}{s}, \tag{6.7}$$

where:

$$a = 1 - \frac{1}{T_{(ON)}(R_{ON} - C)}, \tag{6.8}$$

$$s = 1 - \frac{1}{T_{(OFF)}(C - R_{OFF})} \tag{6.9}$$

Substitute (6.8) and (6.9) into (6.7),

$$\eta_{N(ON-OFF)D1} = \frac{1 - \frac{1}{T_{(ON)}(R_{ON} - C)}}{1 - \frac{1}{T_{(OFF)}(C - R_{OFF})}} \tag{6.10}$$

The parameterisation of equation (6.12) can be found in [Schormans99B] and the formulas are listed in Table 6.2.

6.3.3 Service rate formula

When forcing equivalence of the decay rate of the aggregated model in (6.6) and the original model in (6.10) the reduced service rate, μ, can easily be derived.

$$\eta_{N(ON-OFF)D1} = \eta_{GM1} = \frac{1/\rho - 1 + e^{-\frac{\mu}{C}}}{e^{\frac{\mu}{C}}(1/\rho - 1) + \frac{\mu}{C}(1 - 1/\rho) + e^{-\frac{\mu}{C}}} \tag{6.11}$$

Therefore,

$$\eta_{N(ON-OFF)D1} = \frac{1/\rho - 1 + e^{-\frac{\mu}{C}}}{e^{\frac{\mu}{C}}(1/\rho - 1) + \frac{\mu}{C}(1 - 1/\rho) + e^{-\frac{\mu}{C}}} \tag{6.12}$$

Using a power series expansion, and only taking the first three terms,

$$e^{-\frac{\mu}{C}} = 1 - \frac{\mu}{C} + \frac{(\frac{\mu}{C})^2}{2!}, \tag{6.13}$$

$$e^{\frac{\mu}{C}} = 1 + \frac{\mu}{C} + \frac{(\frac{\mu}{C})^2}{2!} \tag{6.14}$$

Substitute (6.13) and (6.14) into (6.12),

$$\eta_{N(ON-OFF)D1} = \frac{\frac{1}{\rho} - 1 + 1 - \frac{\mu}{C} + \frac{(\frac{\mu}{C})^2}{2}}{(1 + \frac{\mu}{C} + \frac{(\frac{\mu}{C})^2}{2})(\frac{1}{\rho} - 1) + \frac{\mu}{C}(1 - \frac{1}{\rho}) + 1 - \frac{\mu}{C} + \frac{(\frac{\mu}{C})^2}{2}}$$

then,

$$(\eta_{N(ON-OFF)D1}/\rho - 1)(\frac{\mu}{C})^2 - (2\eta_{N(ON-OFf)D1} - 2)\frac{\mu}{C} + 2\frac{\eta_{N(ON-OFF)D1}}{\rho} - \frac{2}{\rho} = 0$$

then,

$$\frac{\mu}{C}=\frac{2\eta_{N(ON-Off)D1}-2\pm\sqrt{(2\eta_{N(ON-OFF)D1}-2)^2-4(\frac{\eta_{N(ON-OFF)D1}}{\rho}-1)(2\cdot\frac{\eta_{N(ON-OFF)D1}}{\rho}-\frac{2}{\rho})}}{2(\frac{\eta_{N(ON-OFF)D1}}{\rho}-1)}$$

Finally,

$$\mu/C=\frac{1-\eta_{N(ON-OFF)D1}\pm\sqrt{(\eta_{N(ON-OFF)D1}-1)^2-2\cdot(1-\eta_{N(ON-OFF)D1}/\rho)\cdot[(1-\eta_{N(ON-OFF)D1})/\rho]}}{1-\eta_{N(ON-OFF)D1}/\rho} \quad (6.15)$$

From (6.15), the mean service rate μ of the foreground traffic packets can be calculated taking into account the effect of the background traffic. $\eta_{N(ON-OFF)D1}$ can be obtained from (6.10).

Based on the abstracted method described, a simulation model was built using the method with one foreground source feeding into a queue model. This was done using OPNET.

6.4 Validation of burst-scale model

6.4.1 HQ model

The HQ model comprises a single ON-OFF source, which represents the foreground out of 100 sources, a FIFO queuing model and a data collector. The sample parameters (when utilisation = 0.8) are shown in Table 6.1.

Table 6.1 Parameters of the burst-scale experiment for one source

Utilisation of the queueing system, ρ	= 0.8
Input rate of the individual sources, h	= 345 cells per second
Mean ON time of the individual source, T_{ON}	= 0.35 second
Mean OFF time of the individual source, T_{OFF}	= 0.65 second

Number of sources	= 1
Output link capacity of the buffer, C	=15100 cells per second

The major effort is to calculate the new service rate for the HQ model. Considering formula (6.15), it becomes the calculation of $\eta_{N(ON-OFF)D1}$, which is given in (6. 7). The full details of deriving (6.7) can be found in [Schormans99B]. Here is only a brief explanation.

Considering *N* ON-OFF traffic aggregated together into a 'new' ON-OFF traffic, in both ON and OFF periods, there are cell arrivals. This is different from the normal ON-OFF model in which there are no cell arrivals during the OFF period. It is because here, this ON-OFF is actually the aggregation of a large number of sources, and it is more reasonable that there are still arrivals during the OFF period. The arrival rate in both ON and OFF period are denoted as R_{ON} and R_{OFF}. The mean duration of the ON and OFF period are $T_{(ON)}$ and $T_{(OFF)}$. The calculation of these parameters in the aggregated mode can be found in [Schormans99B] , and the formulae are listed in Table 6.2.

Table 6.2. Formula used in to get $\eta_{N(ON-OFF)D1}$

$T_{(ON)} = \frac{h \cdot T_{ON}}{C - A_p}$
$T_{(OFF)} = T_{(ON)} \cdot \frac{1-D}{D}$
$R_{ON} = C + h \cdot \frac{A_p}{C - A_p}$
$R_{OFF} = \frac{A_p - D \cdot R_{ON}}{1-D}$
$A_p = F \cdot T_{ON} \cdot h$
$F = \frac{N}{T_{ON} + T_{OFF}}$
$D = \frac{N_0 \cdot B}{N_0 - A + A \cdot B}$
$A = F \cdot T_{ON}$

$N_0 = \frac{C}{h}$
$B = \frac{A^N}{N!} \Big/ \left(1 + \frac{A^1}{1!} + \frac{A^2}{2!} + \cdots + \frac{A^N}{N!}\right)$

6.4.2 Conventional model

In the test, the conventional model comprises 100 independent but statistically identical ON-OFF sources, a FIFO queueing model, and a data collector. The sample input parameters (when utilisation is 0.8) are shown in Table 6.3.

Table 6.3. Parameters of the burst-scale experiment for 100 sources

Output link capacity of the buffer C	= 15100 cells per second
Utilisation of the queueing system, ρ	= 0.8
Input rate of the individual sources, h	= 345 cells per second
Mean ON time of the individual source, $T_{(ON)}$	= 0.35 second
Mean OFF time of the individual source, $T_{(OFF)}$	= 0.65 second
Number of sources	= 100

6.4.3 Simulation results comparison

Figure 6-6 shows the comparison of results of the conventional model and the HQ model. The mean Inter-Exit time is again used as a comparison criterion. A CDF distribution shown is in Figure 6-7 and it provides additional support for this comparison.

It is clear that the magnitudes of the results of IET are the same. This validation proves that the statistical model as described in section 6.3 works well.

The comparison starts only when the offered load for each source is greater than 0.5. This is because the total input rate of simultaneous bursts cannot exceed the output rate of the buffer for lower loads, which means that the burst-scale queueing is not formed. Using the parameters shown in Table 6.1 and Table 6.3, the formulas in Table 6.2 do not apply when the loads are less than 0.5.

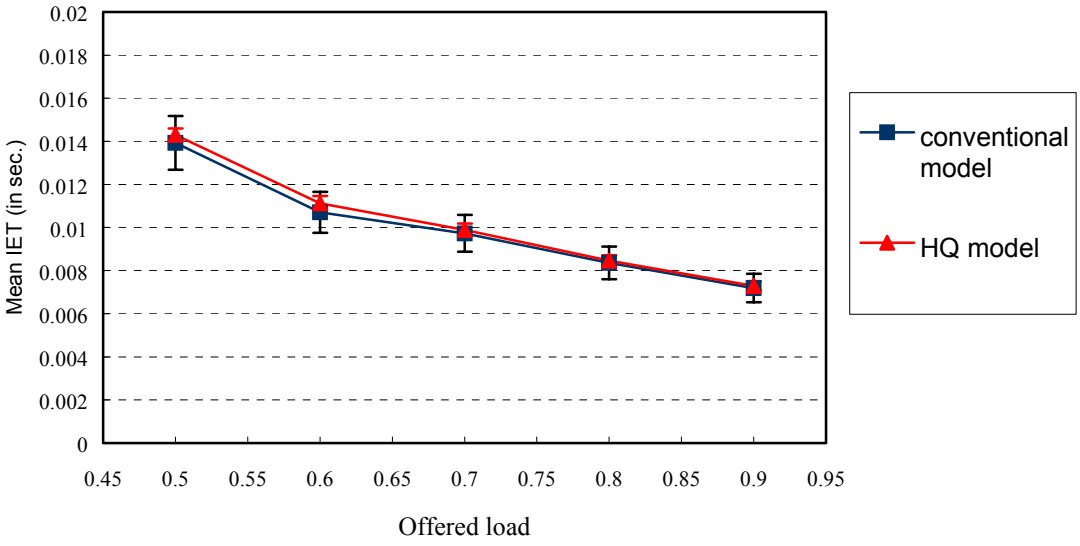


Figure 6-6 Comparison between the conventional and HQ model at burst-scale queueing

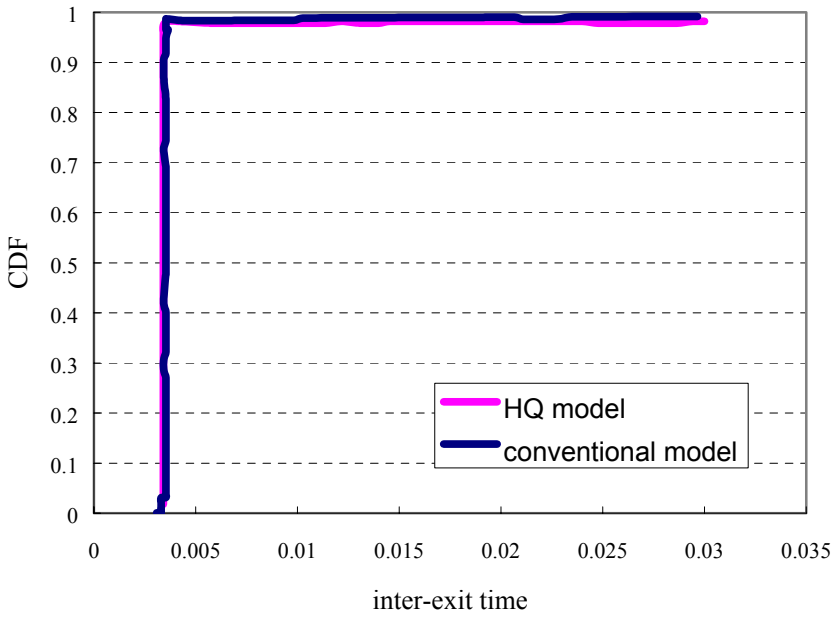


Figure 6-7 CDF comparison of the burst-scale test (offered load = 0.6)

The HQ model shown here works well when compared to the traditional model, therefore, there is no need to apply the neural network to adjust the HQ model.

7. Simulating an IP router

Chapter 5 and 6 showed that the HQ approach works well in both cell scale and burst scale simulation. It substantially speeds up the simulation in both cases, and also achieves the accuracy when a neural network is embedded in the simulation models. However, that was for ATM networks. Now TCP/IP traffic is becoming more and more important on telecommunication networks and hence IP routers are becoming more and more common. In this chapter, the HQ approach is applied to simulating a router, and a simulation model based on HQ is built as an elemental block in a network simulation.

It is known [Schwartz96][Pitts01] that the traffic aggregating from several virtual circuits at the output of a switch/multiplexer tends to be bursty so that the ON-OFF model of Chapter 6 is a natural choice for the router simulation. However, TCP/IP traffic does show other attributes and the intention here is not to explore and model TCP/IP in depth using the HQ approach, but to show its general applicability in modelling a router.

7.1 Test against a 'real' router

The new characteristics of the traffic also affect the network performance analysis. The traffic aggregating from several virtual circuits at the output of a switch/multiplexor tends to cluster in bursts alternating with idle periods. An ON-OFF burst-based pattern is a natural choice for the router simulation. Figure 7-1 [Winstanley98] shows work carried out in the European ACTS project EXPERT (AC094) and shows an illustration of an ATM traffic trace from a Network Interface Card (NIC) that uses a TCP/IP protocol stack.

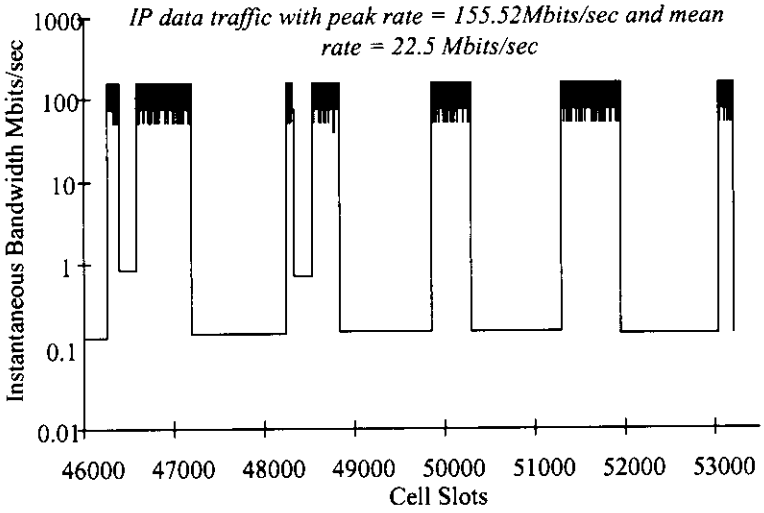


Figure 7-1 IP data traffic

In this part of the research, the HQ simulation was compared to actual hardware, rather than to a conventional simulation[7]. All the *hardware* results detailed in this chapter are based upon a single Linux router with 20 ON-OFF sources feeding into a source node, which was then connected to the router. The generation of packets in ON periods is exponential distributed. In the HQ simulation in OPNET, the same type of traffic and load are modelled.

In the accelerated simulation, the service time for the ON-OFF source representing the foreground traffic is calculated by the formula described in the previous chapter. Figure 7-2 shows the mean inter-exit time of the HQ model and testbed. To show the accuracy of the simulation model the confidence interval with the confidence of 95% is also shown in the graph.

In these tests, the parameters were specified slightly different in order to fit the capabilities of the hardware. The mean number of packets is now the mean number in the ON period and also T_{off} is fixed at a constant value. Details of the parameters are given in Table 7.1.

[7] The router research lab at Nortel Networks' (UK) Harlow Lab was used for these results

Table 7.1. Parameters for validation against the RRL

utilisation	0.1~0.8
link capacity (*C*)	10Mbit/sec
no. of ON/OFF sources (*N*)	20
mean no. of pk during ON	10~83
***Ton* (sec)**	0.032~0.2653
***Toff* (sec)**	6.368

Figure 7-2 and Figure 7-3 show the results of the HQ model and the testbed. There is quite a large difference between them. In the testbed, Ethernet is used for the LAN emulation. Ethernet operates at 10Mbit/s over a bus topology using Carrier Sense Multiple Access with Collision Detection (CSMA/CD) medium access control protocol. In the testbed environment, there are several routers connected to the LAN. Because of the CSMA/CD protocols, even if link capacity is chosen to be 10Mbit/s, the actual service rate cannot reach 10Mbit/s. In contrast, in the case of the HQ model, which is implemented in OPNET, the service rate can reach 10Mbit/s; therefore, the output difference occurs. However, later, a neural network is introduced to learn this difference and the re-alignment makes the HQ model behave more like the testbed.

In order for the HQ model to better fit to the real equipment, the analytical model within the HQ model can be re-aligned as before with the help of the neural network.

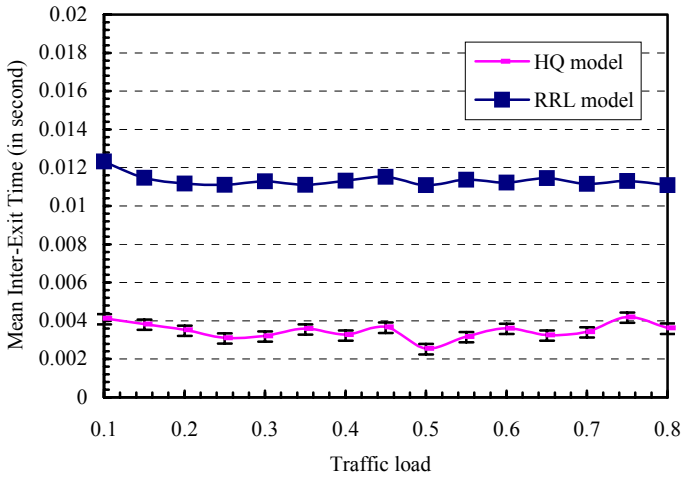


Figure 7-2 Results in the IP networks

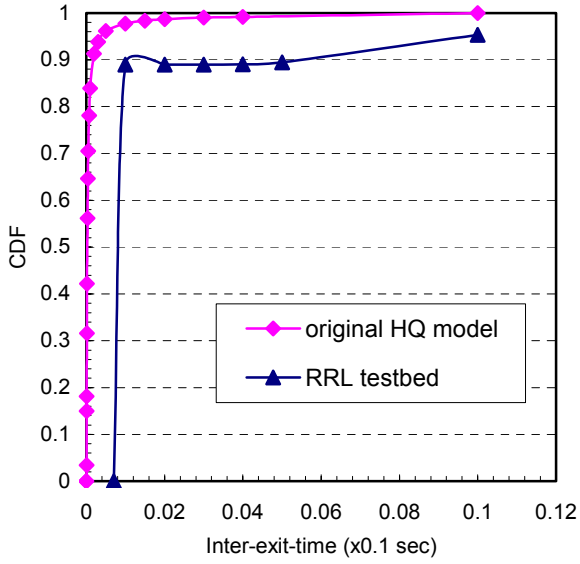


Figure 7-3. The CDF results of the IP networks

7.2 Re-align the HQ model according to the 'real' router

The neural network was trained with an ON-OFF traffic input pattern, and the traffic load ρ takes value in the range of $0.1 \le \rho \le 0.9$. Mean number of packets generated during the ON period is from 10 to 83. The mean inter-exit time is used as the comparison criterion against the testbed because, once again, the outputs of the HQ simulator and the testbed have similar CDF distributions of Inter-Exit Time.

In this case it was found that 2-8-1 neural networks structure provide the best performance for an IP node.

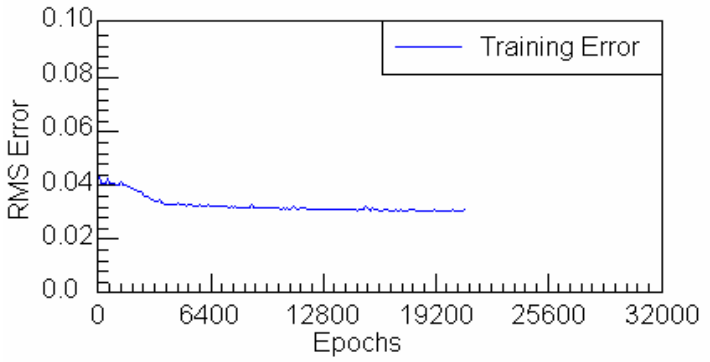


Figure 7-4 RMS vs. training time

Referring to Figure 7-5 and Figure 7-6, the results from the adjusted HQ model show that they now match more closely the Router Lab results. In the figure shown below, the neural network re-aligned HQ model does not work as well as in the previous tests and gives a bigger difference, compared to the testbed result. This is because the neural network re-alignment did not take into account the product dependent features of the real router plus the physical layer effects of the Ethernet connection.

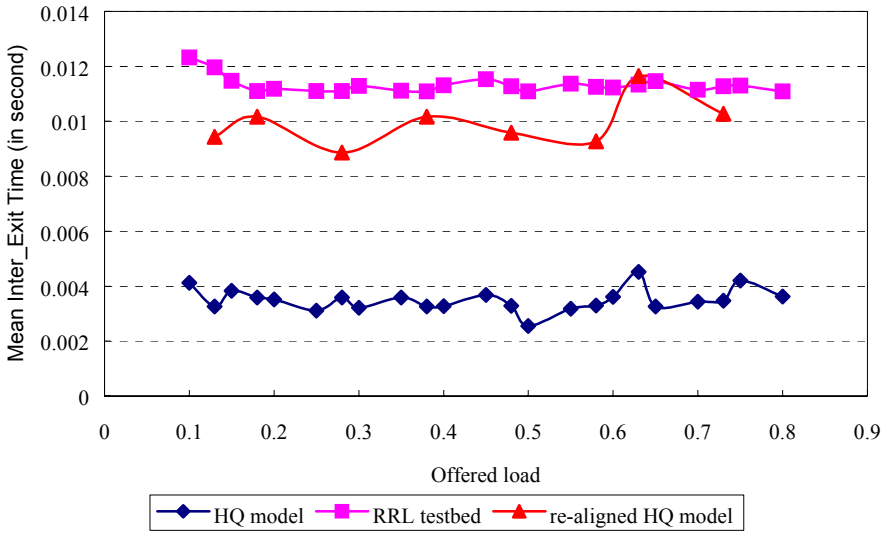


Figure 7-5. Test of the NN prediction in IP node

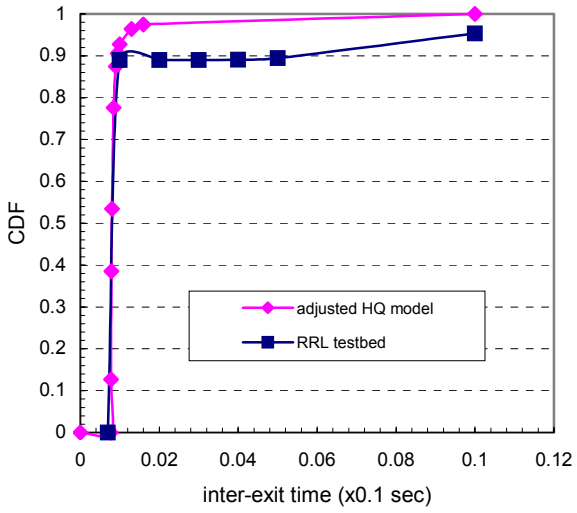


Figure 7-6. Results of the adjusted HQ models

Figure 7-3 indicates that the largest modelling errors are because the overall IET rate is different: this corresponds to the vertical line in that figure. The neural network successfully corrects this, as shown in Figure 7-6

This again proves that the HQ model not only works well when compared to a traditional simulation model, but also works well in the case when the testbed is hardware with product constraints.

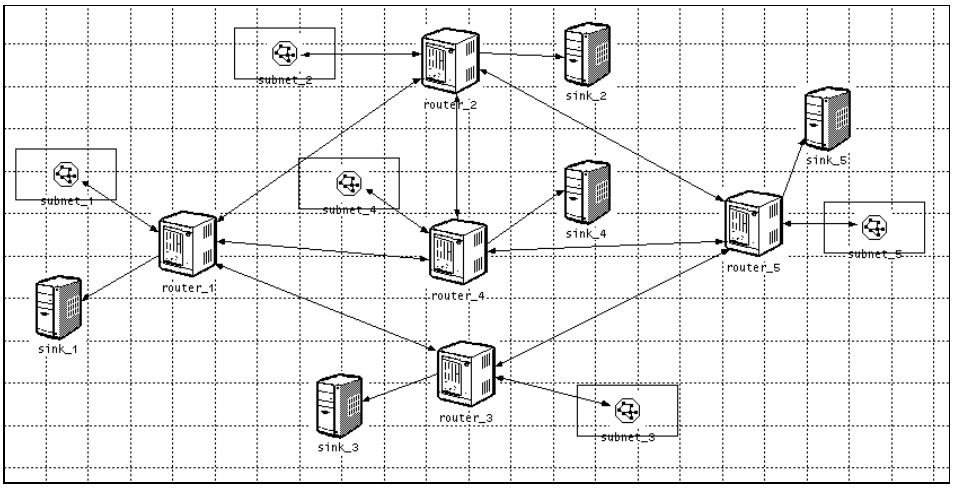
7.3 Validation in the network environment

As a final test, the HQ model is used within a network simulation.

Figure 7-7 shows the network topology used for this investigation. The HQ method is compared against an equivalent conventional simulation.

The network structure is a 10-router network model. The upper part of Figure 7-7 shows the backbone network and each of the backbone router is connected to a local subnet router, as shown in the bottom part of the figure. The subnet routers then connect to 6 workstations (there are 6 workstations only in subnet_1, in the rest of the subnets there are 5 workstations). Each workstation generates a traffic stream that

terminates at each of the backbone routers, but one workstation of one subnet is selected to generate the foreground traffic stream. In the HQ network, only this foreground traffic component is actually generated, the other traffic sources being disabled.



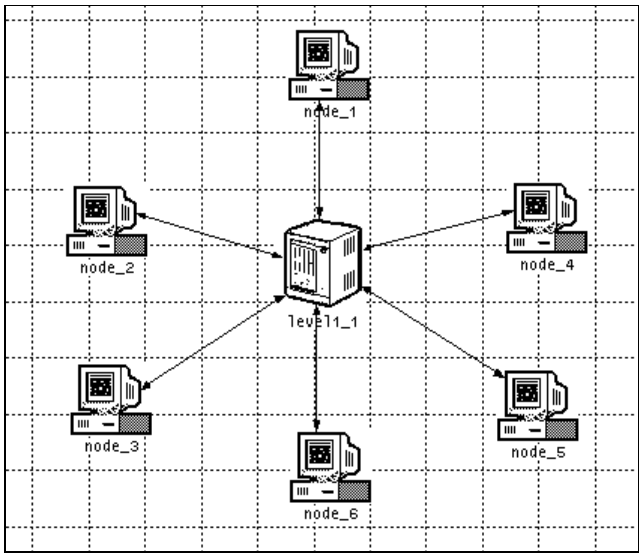


Figure 7-7. Network structure of for the comparison

Figure 7-8 and Figure 7-9 show the probability distributions of the comparison between the HQ model and the testbed. In the test, fixed routing is used, and the foreground stream is generated by the node_1 in subnet 1. It travels through the backbone router 1, 2 and 5, and terminates at sink_5, for data collection.

The comparison parameters used here between the two models are *inter-arrival time* plus *end-to-end delay* as the latter is perhaps a more useful measure of network performance. In both figures, the HQ model provides results that are in close agreement with the testbed.

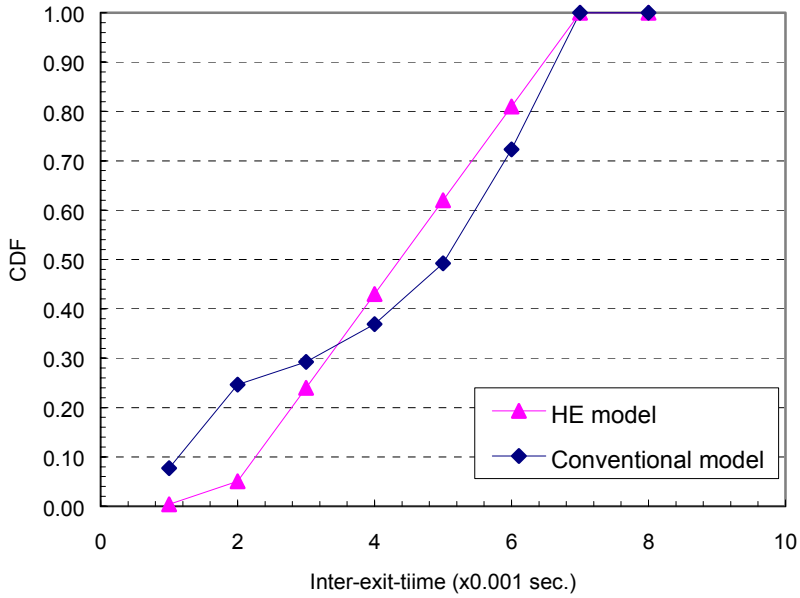


Figure 7-8. Probability distribution of inter-exit time for HQ model and the conventional model

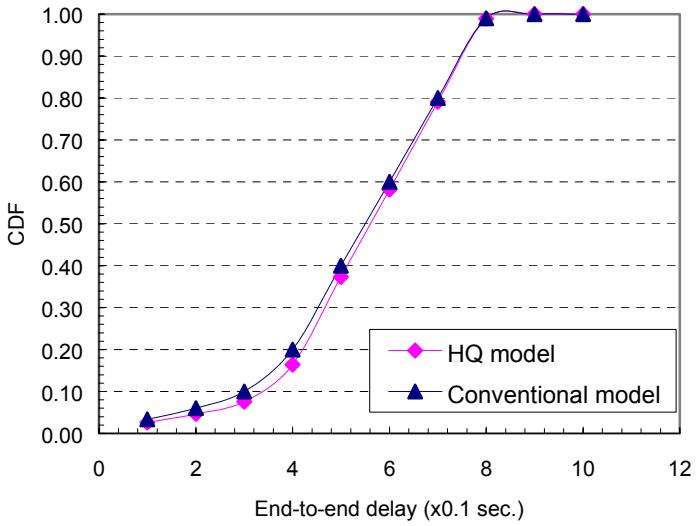


Figure 7-9. Probability distribution of end-to-end delay of the HQ model and the conventional model

The results clearly show that the proposed HQ model can be used successfully within a network environment, providing an accurate representation of queueing system(s) whilst providing significant speedup relative to conventional models. In this particular case, the HQ model is more than 10 times faster than the conventional model.

8. Conclusions and future work

Network simulation has always been an important method for studying network performance, allowing the network designer to draw important conclusions before major capital investments are made. Theoretical and mathematical analysis serve the same purpose as simulation, but when the object of study is too complex, analysis tends to be unmanageable, leaving simulation as the only tractable method. Furthermore, given the large number of nodes and the complexity of the networks typically under investigation, speedup is desirable. The HQ method proposed in this thesis achieves speedup with a novel approach that sustains accuracy while accelerating the simulation by factors of between 10 to10^3, depending upon the network conditions being modelled.

8.1 Conclusions

As described in the body of this thesis, HQ simulation modelling comprises two steps: analytical abstraction and neural network realignment. In analytical abstraction, only the traffic flow of interest is simulated, the background traffic being represented by a modification to the queuing model. This modification is the major analytical modelling exercise as the model may not necessarily be the same in every type of network and under every type of traffic condition (see "Further Work"). The ER queuing concept has been used in the derivation and parameterisation of the service time formulae.

The foreground traffic is defined by the user and usually represents a very small portion of the total traffic: hence, the method proposed here speeds up the simulation significantly. Traditionally, background traffic is explicitly simulated, taking a lot of processing time in the simulation.

In the second phase, a neural network is used for realignment. A trained neural network is embedded into the simulation model in order to improve the accuracy of simulation, which may be hitherto slightly affected due to the assumptions made in

the derivation of the service time formulae, or because of physical effects in real hardware. By comparing the HQ simulation model against a testbed, which can be a traditional simulation model or a hardware testbed, the difference between the HQ model and a testbed can be obtained. This difference is used to train a neural network. The feed-forward backpropagation algorithm was adopted for training a neural network in this application. The trained neural network is built into the HQ simulation, and used to compensate for the difference of the HQ modelling under any inputs without the testbed being run at all. The overhead added into the HQ simulation by the embedded neural network is insignificant as the time consumed on the computation of the neural network is much less than the simulation itself. Thus the realigned HQ simulation model both achieves the simulation acceleration and sustains the simulation accuracy.

The HQ simulation has been applied at both cell and burst scale, these two levels being widely used in ATM switching analysis [Roberts91] [Norros91]. For burst scale the HQ model has been compared against a conventional simulation and for cell scale the HQ model has been compared with a theoretical model and a detailed ATM switch model. All the comparisons showed that the HQ simulation model matches the testbed model well without needing the embedded neural network, apart from where physical effects (modelled in the detailed ATM switch model) affect the performance.

It has been shown that the HQ approach is also applicable to the simulation of IP traffic. In an IP scenario, the input sources have been assumed to be ON-OFF, and the aggregated traffic has been also assumed ON-OFF traffic. In this case, the HQ simulation model was compared against a hardware IP router. The results showed that the HQ simulation model works well in this scenario, especially with the embedded neural network.

Additionally, the HQ simulation has been used in a network environment, and worked well, as shown by the results given in Chapter 7.

The HQ approach requires that one has to have a clear understanding of the traffic behaviour in order to analytically model the background traffic. However, while this may be difficult in some cases, the ON-OFF model can represent a wide range of traffic conditions, so this constraint is unlikely to be too restrictive in practice.

HQ simulation is a brand new approach for accelerating simulations, and it could work in addition to any existing methods of speeding up simulation like *Importance Sampling*. Some ideas for further work on the HQ approach are discussed in the next section.

[Law99] gives several ways of validating a simulation model. In this research, the validations were made against *existing theory* and *relevant results from similar simulation studies*. Several comparison tests were made to prove the effectiveness and efficiency of the HQ simulation approach. The comparison was made between the HQ model and a conventional simulation model that explicitly simulates both foreground and background traffic. In most of the cases in this thesis, the conventional simulation model was prepared by a "trusted source", such as Nortel (for the detailed switch simulator) or Mil3 (for the OPNET library model). In addition, comparisons were made against real hardware.

Under all circumstances, the HQ simulation approach was considerably faster (i.e. several orders of magnitude) than its conventional equivalent, whilst retaining the accuracy of the conventional counterpart.

In addition, the scope of the HQ simulation scheme is limited only by the ability to represent the background behaviour analytically. This means that its applicability extends well beyond accelerated simulations of telecommunications networks, providing significant benefits when the proportion of background to foreground "traffic" is large.

8.2 Future work

The first possibility for future work is to use more than one variable to adjust the analytical HQ model to allow more degrees of freedom, for example the IET distribution could be adjusted.

The second trend of the future work is to model Long-Range Dependence (LRD) traffic. During the last few years, a great amount of research has focused on obtaining realistic models for the traffic generated by the users of new, multi-service telecommunications networks. LRD has proved to be an important feature of certain traffic, and several models of this type have been proposed with the objective of reflecting the real statistical behaviour of this type of traffic.

Therefore, using a traffic source model that has LRD features and modelling this as an abstracted queue, is one possibility for future work. [Gallardo99] gives a comparison of methods modelling LRD traffic and [Ma00] developed formulas that can be used to predict the buffer overflow probability of ON-OFF source with long-range-dependent characteristics, where both ON and OFF periods are Pareto distributed. The future work of HQ could use the derivation in this paper for the decay rate.

Thirdly, a library of analytical models that capture the behaviours of aggregated background traffic could be built up so that, by integrating the probe source (foreground traffic) data with an appropriate analytical library model, different HQ models could be selectively used to represent background traffic as network conditions vary. This is particularly relevant to the modelling of traffic with feedback behaviour, such as Transmission Control Protocol (TCP).

Finally, future work could focus on applying the HQ approach in more network environments. Although an example is provided in Chapter 7, this could also be extended by generating different models to represent different types of equipment in the network.

As described in [Paxson97], a basic question for a network simulation is what topology to use for the network being simulated. [Paxson97] also stated that there is no such thing as a ‘typical’ Internet topology; simulations exploring protocols that are sensitive to topological structure can at best hope to characterise how the protocol performs in a range of topologies. Future work could apply the HQ method to assess its performance under diverse topological circumstances.

Appendix A Backpropagation algorithm[8]

A multilayer neural network with input layer $X_1, \ldots, X_n$, one hidden units: $Z_1, \ldots, Z_p$, and output layer $Y_1, \ldots, Y_m$ is shown in Figure 1. The weights from input layer to the hidden layer is v_{ij}, and from hidden layer to the output layer is w_{jk}. The hidden layer may also have biases (not shown in the graph, but was used in the training algorithm). These bias terms act like weights on connections from units whose output is always 1. In the description of the algorithm, v_{0j} denotes the bias on the hidden layer Z_j and w_{0k} denotes the bias on the output layer Y_k. α is the learning rate, which is used to control the amount of weight adjustment at each step of training. It is a user defined value before the training takes place.

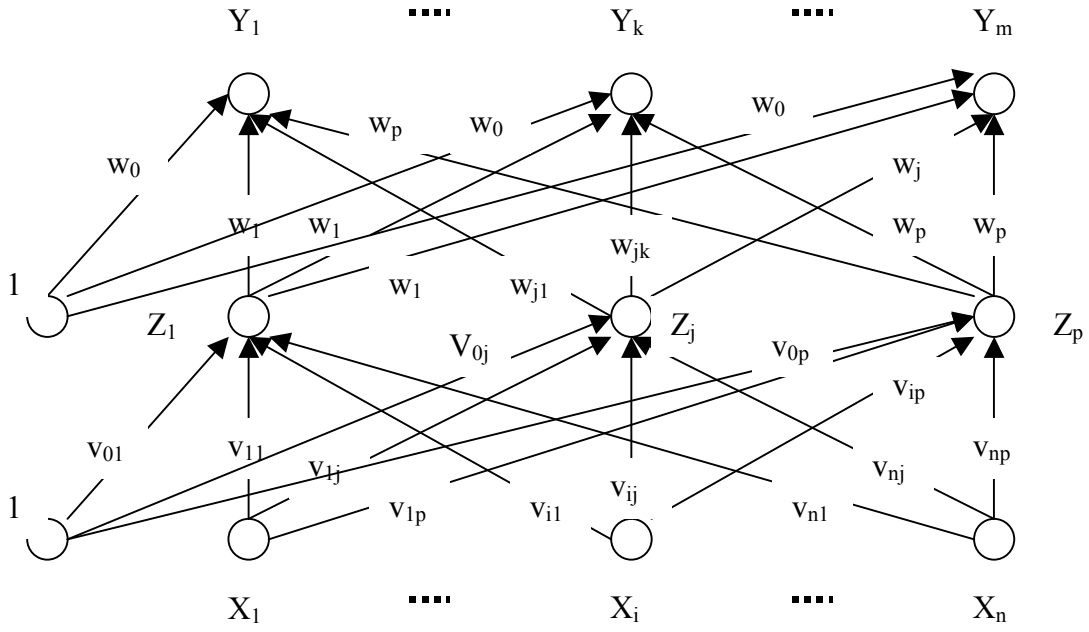


Figure 1 Backpopagation neural network with one hidden layer

Training a network by backpropagation involves three stages: the feedforward of the input training pattern, the backpropagation of the associated error, and the adjustment of the weights.

[8] This summarised description is extracted from [Fausett 94], where readers can find out more systematic context.

Step 0. Initialised weights randomly.

Step 1. While stopping condition is false, do Step 2-9.

Step 2. For each training pair, do Step 3-8.

Feedforward:

Step 3. Each input unit X_i $(i = 1,...,n)$ receives input signal x_i and broadcasts this signal to all units in the layer above (the hidden units).

Step 4. Each hidden units Z_j $(j = 1,...p)$ sums its weighted input signals,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

applies its activation function to compute its output signal,

$$z_j = f(z\_in_j),$$

and sends this signal to all units in the layer above (output units).

Step 5. Each output unit Y_k $(k = 1,...,m)$ sums its weighted input signals,

$$y\_in_k = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

and applies its activation function to compute its output signal,

$$y_k = f(y\_in_k).$$

Backpropagation error:

Step 6. Each output unit Y_k $(k = 1,...,m)$ receives a target pattern corresponding to the input training pattern, computes its error information term,

$$\delta_k = (t_k - y_k) f'(y\_in_k),$$

calculates its bias correction term (used to update w_{jk} later),

$$\Delta w_{jk} = \alpha \delta_k z_j,$$

calculates its bias correction term (used to update w_{0k} later),

$$\Delta w_{0k} = \alpha \delta_k,$$

and sends δ_k to units in the layer below.

Step 7. Each hidden unit Z_j $(j = 1,...p)$ sums its delta inputs (from units in the layer above),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk},$$

mulitplies by the derivative of its activation function to calculate its error information term,

$$\delta_j = \delta\_in_j f'(z\_in_j),$$

calculates its weight correstion term (used to update v_{ij} later),

$$\Delta v_{ij} = \alpha \delta_j x_i ,$$

and calculates its bias correction term (used to update v_{0j} later),

$$\Delta v_{0j} = \alpha \delta_j .$$

Update weights and biases:

Step 8. Each output unit Y_k $(k = 1,...,m)$ updates its bias and weights $(j = 0,...,p)$:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} .$$

Each hidden unit Z_j $(j = 1,...p)$ updates its bias and weights $(i = 0,...,n)$:

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} .$$

Step 9. Test stopping condition.

Publications

1) E Liu, L Cuthbert, J Schormans, G. Stoneley, "Fast and adaptive simulation method for IP networks", 8th International Conference on Telecommunication Systems, Nashville, TN, USA, March 2000
2) J. Schormans, E Liu L Cuthbert and G. Stoneley, "Analytic technique for accelerating simulation of generic network traffic", IEE Electronic Letters, Vol 35, Dec 1999
3) E. Liu, J. Schormans, L. Cuthbert, G. Stoneley and C.Phillips, "A novel accelerated simulation technique for WANs", Proceedings of Sixteenth UK Teletraffic Symposium on Management of Quality of Service, UK, May 2000
4) E Liu, J Schormans, L Cuthbert, G. Stoneley, "A novel fast simulation method for ATM networks", Proceedings of the IEEE Conference on High Performance Switching and Routing, Germany, June 2000
5) E Liu, L Cuthbert, J Schormans, G. Stoneley, "Neural network in fast simulation modeling", accepted by IEEE-INNS-ENNS International Joint Conference on Neural Networks, Italy, July 2000
6) John Schormans, Enjie Liu, Laurie Cuthbert and Jonathan Pitts 'A hybrid technique for the accelerated simulation of ATM networks and network elements', ACM TOMACS, 99- 38,
7) E. Liu, J. Schormans, L. Cuthbert, C.Phillips, G. Stoneley "Application of neural networks to accelerated simulation of Internet type WANs" 17th International Telecommunications Congress, Brazil, Dec. 2001
8) E. Liu, J. Schormans, L. Cuthbert, C. Phillips, G. Stoneley, "Fast hybrid simulation with neural network prediction", World Multiconference on Systemics, Cybernetics and Informatics, 23-26/07/00, Vol 7, Computer Science and Engineering Part 1, pp254-258

References

[Abate95] Abate J., Choudhury G., and Whitt W., "Exponential approximation for tail probabilities in queues: waiting time", Operations Research, Vol. 3, No. 5, Sept.- Oct., 1995

[Ahn96] J. S. Ahn and P. B. Danzig, "Packet network simulation: speedup and accuracy versus timing granularity", IEEE/ACM Transctions on Networking, vol. 4, pp. 743-757, October 1996

[Akyamac99] Akyamac Ahmet A., Haraszti Zsolt and Townsend J. Keith "Efficient rare event simulation using DPR for multidimensional parameter spaces", 16th International Teletraffic Congress, 1999, Edinburgh, UK

[Baiocchi91] Andrea Baiocchi, Nicola Blefari Melazzi, Aldo Roveri, and Roberto Winkler, "Loss performance analysis of an ATM multiplxer loaded with high-speed ON-OFF sources", IEEE Journal on Selected Areas in Communications, Vol. 9, No.3, April 1991

[Ben97] Ben Letaief K. Muhammad K. Sadowsky JS. "Fast simulation of DS/CDMA with and without coding in multipath fading channels", IEEE Journal on Selected Areas in Communications, Vol15, no.4, May 1997, pp626-639

[Bhatt98] Bhatt S. Fujimoto R., Ogielski A., Perumalla K, "Parallel simulation techniques for large-scale networks", IEEE Communications Magazine, Vol.36, No. 8, 1998

[Bishop95] Bishop Christopher M., "Neural networks for pattern recognition", Oxford University Press Inc., New York, 1995

[Bromirski96] Marek bromirski "ATM traffic Shape with Neural Control", Fourth Workshop on Performance Modelling and Evaluation of ATM Networks. July 1996, Ilkely, UK

[Brown94] Timothy X Brown, Neural Networks for Switching", Neural Networks in Telecommunications, Kluwer Academic Publishers, 1994

[Caceres91] Ramon Caceres, Perter B. Danzig, Sugih Jamin, Danny J. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations", ACM SIGCOMM '91,

[Cassandras97] Cassandras CG. Gong W. "New Directions Towards real-time Simulation", Proceedings of SPIE – The International Society for Optical Engineering, Apr. 1997, Orlando FL, USA

[Chen92] Chen Davis X., and Mark Jon W.,, "Delay and Loss Control of An Output Buffered Fast Packet Switch Supporting Integrated Services", ICC'92, 1992

[Computer82] "Computer Networks and Simulation III", Amsterdam, The Netherlands: North-Holland, 1982

[Deman 98] Deman SW., Boel RK, "Importance sampling techniques for the estimation of the CLR and its sensitivity", AEU-International Journal of Electronics & Communications, Vol. 52, No. 3, 1998, pp. 141-51. Publisher: S. Hirzel, Germany

[Devetsikiotis93] Devetsikiotis M., Townsend J.K., "Statistical Optimization of Dynamic Importance Sampling Parameters for Efficient Simulation of Communication Networks", IEEE/ACM Trans. Networking, vol. 1, no. 3, June 1993, pp. 293-305

[Eckberg 79] Eckberg A. E., "The single server queue with periodic arrival process and deterministic service time", IEEE Trans. Commun., vol. 27, pp. 556-562, Mar. 1979.

[Fausett 94] Fausett Laurene, "Fundamentals of Neutal Networks, Architectures, Algorithms, and Applications" Prentice Hall International Editions, 1994

[Feller66] Feller W., "An introduction to probability theory and its applications" Vol. II, New York: Wiley, 1966

[Gallardo99] Gallardo JR., Makrakis D., Orozco-Barbosa L., "Fast simulation of roadband telecommunications networks carrying long-range dependent bursty traffic", Winter Simulation Conference Proceedings, Vol. 1, 1999 pp374-381

[Guo00] Guo Yang, Gong Weibo, Towsley D., "Time-stepped hybrid simulation (TSHS) for large scale networks", Poceedings IEEE INFOCOM 2000, Vol. 2, pp441-450, 2000

[Handel94] Handel Rainer, Huber Manfred N., Schroder Stefan, "ATM networks – concept, protocols, applications", Second Edition, Addison-Wesley, 1994

[Hao98] Hao F. Nikolaidis I. Zegura EW., "Efficient simulation of ATM networks with accurate end-to-end delay statistics", Conference Record of 1998 IEEE International Conference on Communications, pp. 1799-1804, June 1998

[Hidelberger98] P. Hidelberger and R. Simha, "Fast simulation of a voice-data multiplexer", International Journal of modeling and Simulation, Vol 18, No. 4, 1998

[Kesidis96] G. Kesidis, A. Singh, D. Cheung, and W. Kwok, "Feasibility of fluid event-driven simulation for ATM networks", IEEE Globecom, London, Nov. 1996

[Kurose88] James F. Kurose, Hussein T. Mouftah, "Computer-Aided Modeling, Analysis, and Design of Communication Networks", IEEE Journal on Selected Areas in Communications, Vol. 6, No.1, Jan. 1988

[Law99] Averill M. Law, W. David Kelton, "Simulation Modeling and Analysis", McGraw-Hill, Third Edition, 1999

[Lee99] Lee Mihye, Moon Sungjin, Park Heasook, Song Kwangsuk, Kwon Boseob, Kim Daeyoung, "An effective cell scheduler guaranteeing fairness for input-queued ATM switch", International Conference on Communication Technoogy, Oct 1998, China

[Maisie/PARSEC] Parallel Simulation Environment for Complex Systems http://may.cs.ucla.edu/projects/parsec/.

[Ma00] Ma A.H., Schormans J.A., Pitts J.M., Scharf E.M., Pearmain A.J., Phillips C.I., "Design rules and equivalent capacity for buffering of Pareto source", Electronic Letters, July 2000, Vol. 36, No. 15, p1274-1275

[Mil3] www.mil3.com

[Mitchell00] K. Mitchell, A..van de Liefvoort, and J. Place 'Second-Order Statistics of an Isolated Departure Stream from a Shared Buffer with Correlated Sources', Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis, March 2000, USA

[Nicol98] Nicol D.M., Ed., Special issue on the Telecommunications Description Language, ACM SIGMETRICS Perf. Eval. Rev., vol. 25, no. 4, Mar. 1998

[Nikolaidis93] I. Nikolaidis, R. M. Fujimoto, and A. Cooper, "Parallel simulation of high-speed network multiplexers", Proceedings of the IEEE Conference on Secision and Control, Dec. 1993

[Nikolaidis94] I. Nikolaidis, R. M. Fujimoto, and A. Cooper, "Time parallel simulation of cascaded statistical multiplexer". Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer systems, pp. 231-240, may 1994

[Ng97] Ng Chee Hock, "Queueing modelling fundamentals", John Wiley & Sons, 1997

[Norros91] Iikka Norros, James W. Roberts, Alain Simonian, and Jorma T. Virtamo, "The Superposition of Variable Bit Rate Sources in an ATM Multiplexer", IEEE Journal on Selected Areas in Communications, Vol. 9, No3, April 1991

[Onvural93] Raif O. Onvural, "Asynchronous Transfer Mode Networks: Performance Issues", Artech House, 1993

[Pattavina98] Achille Pattavina, "Switching Theory" – Architecture and Performance in Broadband ATM Networks, Wiley, 1998

[Paxson95] Vern Paxson and Sally Floyd, "Wide Area Traffic: The failure of Poisson Modelling", IEEE/ACM Transactions Networking, vol. 3, pp226-244, June 1995

[Phillips91] Chris. I. Phillips, Laurie G. Cuthbert, "Concurrent Discrete Event-Driven Simulation Tools", IEEE Journal on Selected Areas in Communications, Vol. 9 No.3. April 1991

[Pitts95] J.M Pitts, "Cell_rate modelling for accelerated simulation of ATM at the burst level", IEE proceedings Communications, Dec. 1995

[Pitts01] Pitts J. M., Schormans J. A., "Introduction to IP and ATM design and performance", John Wiley & Sons LTD, Second Edition, 2001

[Roberts91] James W. Roberts, "Variable-Bit-Rate Traffic Control in B-ISDN", IEEE Communications Magazine, Sept. 1991

[Saito96] H. Saito and T. Tsuchiya, " Upper Bound of Loss Probability for self-similar traffic", Conference record of the international conferene on communications (ICC), Dallas, Texas, p.S52, June, 1996,

[Sauer83] C. Sauer and E. MacNair, "Simulation of Computer Comunication Systems", Englewood Cliffs, NJ: Prentice-Hall, 1983

[Schoemaker82] S. Schoemaker, Ed., "Computer Networks and Simulation II", Amsterdam, The Netherlands: North-Holland, 1982

[Schormans94] Schormans J., Pitts J., Cuthbert L. "Exact fluid-flow analysis of single on/off source feeding an ATM buffer", IEE Electronic Letter, 7th July, 1994, vol.30, No 14, pp1116-1117

[Schormans96] Schormans J. A., Pitts J.M., Clements B.R., Sharf E. M., "Approaximation t M/D/1 for ATM CAC, buffer dimensioning and cell loss performance", Electronic Letters, Feb., 1996, Vol. 32, No. 3, p164-165

[Schormans97] Schormans J.A.and Pitts J.M., "Solution for M/G/1 Queues" IEE Electronic Letter, 4th Dec. 1997, vol. 33, no. 25, pp2109-2111

[Schormans99A] Schormans J.A., and J.M.Pitts, "Decay Rate Modelling in Queueing System and Application to Telecommunications," IFIP Conference prceedings, pp. 10/1-10/14, April 1999.

[Schormans99B] Schormans John, Liu Enjie, Cuthbert Laurie and Pitts Jonathan 'A hybrid technique for the accelerated simulation of ATM networks and network elements', ACM TOMACS, 99- 38,

[Schwartz96] Schwartz M., “Broadband Integrated Networks ” *Prentice-Hall*, pp. 74-76, 1996.

[Seelen85] Seelen L. P., Tijms H. C., “approximations to the waiting time percentiles in M/G/c queue”, Teletraffic Issues in an advanced Information Society, ITC-11, M. Akiyama (ed.). Elsevier, Amsterdam, 53-57, 1985

[Shaikh89] Shaikh S.Z., Schwarts M, and Szymanski T.H., “Performance analysis and design of banyan network based broadband packet switches for integrated service”, Proc. GLOBECOM’89, pp 1154-1158

[Shaikh90] Shaikh S.Z., Schwarts M, and Szymanski T.H., “Analysis, control and design of crossbar and banyan based broadband packet switches for integrated service”, Proc. ICC’89, pp 761-765

[Stewart02] Stewart R., PhD thesis, 2002

[Sun] www.doc.sun.com

[Suzuki 98] Suzuki S. Nakagawa K., “Performance evaluation of an ATM switch with back pressure by IS simulation”, Transactions of the Institute of Electronics & Communication Engineers of Japan, Part B, Vol. J81B-I, No. 6, June 1998, pp. 371-7.

[Tarassenko98] Tarassenko L., “A Guide to Neural Computing Applications”, Published by Arnold, London, 1998

[TeleSim] http://bungee.cpsc.ucalgary.ca/TeleSim

[Tijms86] Tijms H. C., “Stochastic modelling and analysis: a computational approach”, John Wiley, New York, 1986

[Timotijevic98] Timotijevic T. and Schormans J.A., “Traffic analysis of ATM over DS-CDMA over a satellite link”, UK Teletraffic Symposium, 1998

[Townsend98] Keith J. Townsend, Zsolt haraszti, Freebersyser James A., Devetsikiotis Michael, “Simulation of Rear Event in Communications Networks”, IEEE Communications, Vol. 36 No. 8, Aug. 1998

[Tunnicliffe01] Tunnicliffe MJ., Parish DJ., "A hybrid simulator for an ATM network", International Journal of Communications Systems", Vol. 12, No. 2, March 2000, pp179-184

[Turner92] Turner S. and Xu M., "Performance evaluation of the bounded Time Warp algorithm, 6th Workshop on Parallel and Distributed Simulation, vol. 24, pp. 117-128, SCS Simulation Series, Jan. 1992

[Unger94] Unger B. and Xiao Z., "The fast parallel simulation of telecommunications", Proceedings of New Directions in Simulation for Manufacturing and Communications, The Operations Research Society of Japan, August, 1994

[Wagner89] Wagner D., Lazowska E., "Parallel simulation of queueing networks: Limitations and potentials", Proc. 1989 SIGMETRICS Conf., p146-155

[Whitt93] Whitt W., "Approximations for the GI/G/m queue", Prod. And Opns. Mgmt. 2, p141-161

[Winstanley98] Winstanley S., PhD thesis, 1998

[Xuan98] Xuan Q. Y., Aggarwal R.K., Johns A.T., Dunn R.W., Bennett A., "A neural network based protection technique for combined 275 kV/400 kV double circuit transmission lines", Neurocomputing, Vol 23, No. 1-3, Dec., 1998, Elsevier

[Yang89] Yang O.W.W, Mark J.W., "Performance analysis of an integrated service switch", INFO, 27(4), pp. 430-455, 1989

[Yuhas94] Edited by Yuhas Ben, Ansari Nirwan, "Neural Networks in telecommunications", Kluwer Academic Publishers, 1994