INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road Ann Arbor, MI 48106-1346 USA
313·761-4700 800·521-0600

Order Number 9431372

A neural network model for traffic management in broadband networks

Tarraf, Ahmed Abd El Hameed, Ph.D.

City University of New York, 1994

Copyright ©1994 by Tarraf, Ahmed Abd El Hameed. All rights reserved.

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A Neural Network Model For Traffic Management in Broadband Networks

by

Ahmed Abd El_Hameed Tarraf

A dissertation submitted to the Graduate Faculty in Engineering in partial Fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1994

©1994

AHMED TARRAF

All Rights Reserved

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/25/94
Date

Prof. Tarek Saadawi
Chair of Examining Committee

4/25/94
Date

Prof. Gerard Lowen
Executive Officer

Prof. Ibrahim Habib
Prof. Chaim Ziegler
Prof. Myung Lee
Prof. Joseph Baraba
Prof. Sanghamitra Basu
Prof. Michael Conner
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

A NEURAL NETWORK MODEL FOR TRAFFIC MANAGEMENT IN BROADBAND NETWORKS

by

Ahmed Abd El_Hameed Tarraf

Advisor: Professor Tarek Saadawi

Co-advisor: Professor Ibrahim Habib

Asynchronous Transfer Mode (ATM) Broadband networks support a wide range of multimedia traffic (e.g. voice, video, image, and data). This research presents a novel framework for traffic management at the cell level using Neural Networks (NNs). Our new approach incorporates bank of NNs for traffic characterization and description, another NN system for traffic enforcement, and finally a reinforcement learning-based NN to provide a rate-based feedback access control. The NN traffic description method presents a novel approach to characterize and model the multimedia traffic. A bank of backpropagation NNs is used to characterize and predict the time bit-rate variations of the multimedia packet's arrival process. The NN traffic enforcement system includes two policing mechanisms: one is the "Neural Network Traffic Enforcement Mechanism (NNTEM)", the second is the "Reinforcement Learning Neural Network Controller". Both mechanisms do not rely upon the policing of simple parameters such as mean bit-rate, peak bit-rate, or burst duration, but rather an elaborate and very accurate policing of all higher-order moments via the probability density function (pdf) of the traffic. The rate-based feedback control is applied at the access node of the network and is implemented by the reinforcement learning method which, also, ensures an optimal

control approach. The algorithm utilizes a feedback control signal to throttle the peak bit-rate of the arrival stochastic process to the input statistical multiplexer. The feedback control signal is produced (using the NN controller) such that the system performance is maximized. The system performance is defined in terms of the buffer overflow and the coding rate of the input source(s). The results of our new approach show that it is extremely effective in controlling and managing the ATM multimedia traffic when compared to existing methods such as Leaky Bucket and other window-type mechanisms.

Acknowledgments

I wish to express my thanks to my wife, my son and my family in Egypt for their enthusiastic encouragements and unlimited support.

I would like to express my thanks and appreciation to Prof. Tarek Saadawi for his supervision, suggestions, advisement which helped me to overcome many of the difficulties I have encountered throughout my study period at the City University of New York..

My gratefully acknowledgement, appreciation and deep thanks are due to prof. Ibrahim Habib for his close supervision and contributions without which this work would have not been achieved.

Acknowledgment is also due to the members of my doctoral examination committee for their initial suggestions and evaluation of my research work.

Without enthusiastic support from so many friends, it would be impossible to complete my Ph.D. First, my thanks go to Dr. Ahmed Abdelmonem, at AT&T Bell Labs, who has provided me the enough time and the facilities to continue my research during my summer employment at AT&T. Also my special appreciation is due to Mr. Mohamed Bekhit, Mr. Rashad Shouaib, and Mr. kamal Elsharrwy for their unforgettable and unlimited help that they have been provided since I have arrived in the United States.

Table of Contents

List of Figures

I. Introduction

The hopes of carrying out a high speed network in the near future has been pinned to the rising star of Asynchronous Transfer Mode (ATM). The ATM technique has been recommended by the CCITT as the transport vehicle for the future Broadband Integrated Services Digital Networks (B-ISDN) [1]. These networks are designed to provide multimedia traffic services. These services have wide range of both traffic characteristics and demands (Quality of Services (QOS) requirements).

The ATM solution provides the flexibility to integrate the transport of services with a wide range of bandwidth requirements by assigning fixed-length cells to virtual connections on an "as needed basis". ATM also provides the potential to obtain improved bandwidth efficiency through the statistical multiplexing of variable bit-rate (VBR), or bursty traffic streams. In ATM networks, cells from different connections are multiplexed or switched using common fabrics independent of the connections' characteristics and demands. This chapter presents an introduction to the traffic management in broadband networks using Neural Networks (NNs).

1.1 Introduction to Neural Networks

A Neural Network is a parallel, distributed information processing structure consisting of processing elements (PEs) (neurons) interconnected via unidirectional signal channels called connections. Each processing element (PE) has a single output connection that branches (fans out) into as many collateral connections as desired; each carries the same

signal the processing output signal. The processing element output signal can be of any mathematical type desired. the information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signal arriving at the processing element.

The structure of the NNs makes them non-algorithmic, massively parallel computing devices. They do not require the laborious programming needed to break a problem down into a form suitable for parallel processing. Indeed, a NN cannot be programmed. Instead it is trained by exposing it to a specified data set of information environment, which enables its self-organizing and learning capabilities to produce a desired goal.

There are two operation phase in NN information processing. They are learning phase and retrieving phase. In the training phase, a training data set is used to determine the weight parameters that define the NN model. This trained model then will be used later in the retrieving phase to process real test patterns and yield classification results.

Learning Phase: A salient feature of NNs is their learning ability. They learn by adaptively updating the weights that characterize the strength of the connections. The weights are updated according to the information extracted from (new) training patterns. Usually, the optimal weights are obtained by optimizing (minimizing or maximizing) certain "energy" functions. For example, a popular criterion in supervised learning is to minimize the least-squares-errors between the desired output value and the actual output value.

Retrieving Phase: Various nonlinear systems have been proposed for retrieving desired or stored patterns. The results can be either computed in one shot or updated iteratively

based upon the retrieving dynamics equations. The final neuron values present the output to be retrieved.

The NNs are commonly categorized in terms of their corresponding training algorithms: fixed-weights networks, supervised networks, and unsupervised networks. There is no learning required for the fixed-weights networks, so a learning model is supervised or unsupervised.

Supervised Learning Rules: Supervised learning requires the pairing of each input pattern with a target pattern representing the desired output, together these are called training pair. Usually a network is trained over all a number of such training pairs. An input pattern is applied, the output of the network is calculated and compared to the corresponding target pattern, and the difference (error) is fed back through the network and weights are changed according to an algorithm that tends to minimize the error. The patterns of the training set are applied sequentially, and errors are calculated and weights adjusted for each pattern, until the error for the entire training set is at an acceptably low level.

Unsupervised Learning Rules: unsupervised learning requires no target pattern for the outputs, and hence, no comparisons to predetermined ideal responses. The training set consists solely on input patterns. The learning algorithm modifies network weights to produce output patterns that are consistent, that is, both application of one of the training patterns or application of a pattern that is sufficiently similar to it will produce the same pattern of the outputs. The training process, therefore, extracts the statistical properties of training set and groups similar patterns into classes. Applying a pattern from a given class to the input will produce a specific output pattern, but there is no way to determine prior to training which specific output pattern will be produced by a given input pattern

class. Hence, the output of such a network must generally be transformed into a comprehensible form subsequent to the training process. This does not represent a serious problem. It is usually a simple matter to identify the input-output relationships established by the network.

From an application-driven perspective, one can explore neural networks' strengths in nonlinear, adaptive, and parallel processing. NNs have found many successful applications in computer vision, signal/image processing, speech/character recognition, expert systems, medical image analysis, remote sensing, robotic processing, industrial inspection, and scientific exploration. The application domains of NNs can be roughly divided into following categories: association/clustering/classification, pattern completion, regression/generalization, and optimization.

Neural networks were proposed to solve some control problems [2],[3]. Applications to the communications networks control were reported in [4],[5]-[7]. However NNs have not been used in the context of multimedia traffic modeling and characterization. In this research, a three-layered backpropagation NN is used to characterize and predict non-renewal type processes. NNs based on backpropagation algorithm, can learn a nonlinear relation between many variables. These networks have been used in a number of deterministic and stochastic problems [8], [9]. It has been found that these networks perform well in most cases with accurate results.

Backpropagation networks have basically three layers, referred to as input layer, hidden layer(s), and output layer. Each layer contains a number of processing elements (PE), and is fully connected to the next layer. The input data vector is presented via the input layer which fans out the input data without making calculations. The data flows along the connections toward the hidden layer(s) and the output layer. The final result

of this operation is that the input data vector is transformed (mapped) into some corresponding output vector at the output layer. Each PE in a hidden or output layer has a connection from each PE in the preceding layer. Associated with each of these connections is an adaptive weight. The output of a PE in a hidden or output layer is calculated by applying an activation function to the weighted sum of the input to that PE. Various activation functions such as S-shaped functions (sigmoid) and bump function (Gaussian), [9] are available.

During the learning phase of operation, the actual output data vector is compared to the desired output data vector, and the errors between these two vectors are calculated. The error values are then used to calculate the new weights for all output and hidden layers PEs and thereby reduce the error in the network output. This process is repeated until the mapping from the input vector to the output vector has been trained to the desired accuracy. The idea is to find a set of weight values that result in maximum accuracy and minimum error. The error criterion used by backpropagation networks is the Mean Squared Error (MSE) [10].

I.2 ATM Traffic Management

Performance requirements during the information transfer phase of a connection can range from stringent cell delay variation for delay-sensitive services (such as voice and video) to stringent cell loss probability for loss-sensitive services (such as data files transfer). In managing the wide range of traffic types and performance requirements, the key challenge is the equitable and efficient allocation of network resources. The network

provider may, of course, wish to accept any and all connection requests. However, the network provider also has the responsibility to ensure that the admission of a new connection does not adversely affect the performance of other connections sharing those network resources.

Therefore, traffic management functions are required to ensure that enough resources and satisfactory performance are provided to all accepted connections. This capability is relatively simple to implement if the utilization of the network resources, such as bandwidth, is not a concern. In this case, non statistical multiplexing (i.e., deterministic multiplexing) can be deployed, meaning that sufficient resources will be reserved to support each connection's peak bandwidth requirements. If the network provider is striving for increased network utilization through statistical multiplexing, additional traffic control functions must be considered. The challenge then is to design effective traffic management algorithms that allow reasonable utilization of network resources, while also remaining simple, easy to implement and robust to evolving ATM traffic characteristics.

Hence, the primary role of the traffic management is to protect the network and the user in order to achieve network performance objectives. An additional role is to optimize the use of network resources.

The objectives of the traffic management for B-ISDN are as follows:

* The traffic management should support a set of Quality of Service (QOS) classes sufficient for all foreseeable B-ISDN services.

* The traffic management should minimize network and end-system complexity while maximizing network utilization.

To meet these objectives, the following functions form a framework for managing

and controlling traffic in ATM networks and may be used in appropriate combinations [11], see figure (1.1).

* Network Resource Management (NRM): Provisioning may be used to allocate network resources in order to separate traffic flows according to service characteristics.
* Connection Admission Control (CAC) is defined as a set of actions taken by the network during the call set-up phase in order to determine whether a virtual channel connection (VCC)/virtual path connection (VPC) request can be accepted or should be rejected.
* Feedback controls are defined as the set of actions taken by the network and by the users to regulate the traffic submitted on ATM connections according to the state of the network elements.
* Usage Parameter Control (UPC) is defined as the set of actions taken by the network to monitor and control the traffic; in terms of traffic offered and validity of the ATM connection, at the user access. Its main purpose is to protect network resources from malicious as well as unintentional misbehavior, which can affect the QOS of other already established connections, by detecting violations of negotiated parameters and taking appropriate actions.
* Priority Control: the user may generate different priority traffic flows by using the Cell Loss Priority bit. A network element may selectively discard cells with low priority if necessary to protect as far as possible the network performance for cells with high priority.
* Traffic shaping mechanisms may be used to achieve a desired modification of the traffic characteristics.

As a general requirement, it is desirable that a high level of consistency be achieved between the above traffic management functions.

The QOS is defined by a set of parameters such as delay, delay variability (jitter), and Cell Loss Rate (CLR). A user requests one ATM QOS class from the QOS classes that a network provides. The requested QOS class is a part of the traffic contract. The network commits to meet the requested QOS as long as the user complies with the traffic contract.

Most of traditional implementation methods of the traffic management functions are based upon the results obtained from analytical performance evaluation such as queuing models. A major shortcoming of the currently available queuing models is that only steady-state results are tractable and consequently, any traffic management function tailored on the basis of such models can ensure optimal performance only under steady-state conditions. However, performance control methods that dynamically manage traffic flows according to changing network conditions require understanding of dynamics. Furthermore, in these models, hypotheses concerning the detailed knowledge of systems under study can seldom be justified in real-life.

This research presents an adaptive traffic management model using Neural Networks (NNs) which can meet the foregoing requirements of the ATM traffic management. This research is motivated by applying the powerful apparatus of NNs to traffic management problems. The potential benefits expected from using Neural Networks for traffic management are:

1-Adaptive learning, detailed description and mathematical understanding about the underlying network to be controlled are not required as a NN can learn from observations or examples during the course of network operation.

2- High computation rate due to the massive parallelism of the hardware implementation of NNs.

3- Generalization on learning, a NN has the ability to generalize learning to what has never been seen. This particularly useful for learning a dynamic environment for traffic management in ATM networks where observations may be incomplete, delayed or partially available.

4- High degree of robustness or fault tolerance due to the nature of distributed processing.

I.3 Neural Network Model for Traffic Management in ATM Networks

In this research, we have introduced a solution for the ATM traffic management using NNs which can automatically adapt to new network situations. The block diagram of the proposed ATM traffic management using NNs is shown in figure (1.2). As illustrated in the figure, NNs are applied to the call level control functions such as connection admission control (CAC) because their abilities to learn nonlinear functions with many inputs and outputs allow prediction of QOS from observed traffic and, hence, the choice of the optimal control values. NNs are, also, applied to the cell level control functions such as traffic measurements, traffic enforcement (policing) and rate-based feedback congestion control at the access to the network. Moreover, the NNs are expected to be applied to the network control functions such as optimal link capacity assignment and dynamic routing because of their ability to solve optimization problems. NNs could approximate the complicated input-out reactions by selecting significant inputs and

deriving feature parameters from input data autonomously. Thus, a controller employing NNs would be very efficient when used in an ATM network by learning the changing characteristics of the traffic. Moreover, the training of many NNs interconnected to each other is expected to be able to integrate all control levels from the bottom up to the network control level.

This research is primarily concerned with applying NNs to the cell level control. Our contributions are several: 1) First, we have proposed a new traffic description tool using NNs. We will prove that the NN descriptor does, in fact, predict the time bit-rate variations of all types of multimedia traffic. Hence, it is possible to design new congestion control algorithms that can avoid congestion by pre-allocating the necessary resources (bandwidth and buffers). 2) Next we provide a new traffic enforcement algorithm, also, based upon the use of NNs. We will also prove that our algorithm is capable of providing the maximum policing function via the policing of not only first-order moments but also all higher order moments of the traffic. This is made possible by policing the pdf of the traffic. 3) Finally, we apply NNs to the problem of providing the optimal control of potential congestion arising in the access node statistical multiplexer buffer.

In the rest of this section we survey both the call level and the cell level control functions and the proposed NNs solutions. We start with the traffic measurement and prediction because of its importance in determining the parameters that specify different traffic patterns.

1.4 Traffic Measurement and Description

Any traffic management function must make use of the information, in terms of traffic parameters, that passes across the User Network Interface (UNI). Traffic parameters describe the traffic characteristics of an ATM connection. For a given ATM connection, traffic parameters are grouped into a source traffic descriptor, which in turn is a component of a connection traffic descriptor [11].

Traffic Parameters:

A traffic parameter is a specification of a particular traffic aspect. It may be quantitative or qualitative. Traffic parameters may for example describe peak cell rate, average cell rate, average burst length, and/or source type (e.g., data terminal, telephone,video phone, etc..).

ATM Traffic Descriptor:

The ATM traffic descriptor is the generic list of traffic parameters that can be used to capture the traffic characteristics of an ATM connection.

Source Traffic Descriptor:

A source traffic descriptor is a subset of traffic parameters belonging to the ATM traffic descriptor. It is used during the connection set-up phase to capture the intrinsic traffic characteristics of the connection requested by a particular source. The set of traffic parameters in a source traffic descriptor can vary from connection to connection.

Connection Traffic Descriptor:

The connection traffic descriptor specifies the traffic characteristics of the ATM connection at the UNI. The connection traffic descriptor is the set of traffic parameters

in source traffic descriptor, the Cell Delay Variation (CDV) tolerance and the conformance definition that is used to unambiguously specify the connection traffic descriptor to allocate resources and to derive parameter values for the operation of the UPC. The connection traffic descriptor contains the necessary information for conformance testing of cells of the ATM connection at the UNI.

Any traffic parameter and the CDV tolerance in a connection traffic descriptor should fulfill the following requirements:

* be understandable by the user or terminal equipment; conformance testing should be possible.
* be useful in resource allocation schemes meeting network performance requirements
* be enforceable by the UPC.

These criteria should be respected since users may have to provide these traffic parameters and CDV tolerance at connection set-up phase. In addition, these traffic parameters and CDV tolerance should be useful to the CAC algorithm so that network performance objectives can be maintained once the connection has been accepted. Finally, they should be enforceable by the UPC in case of non-compliant usage in order to maintain the network performance.

Traffic Contract Specification:

A traffic contract specifies the negotiated characteristics of a connection at a UNI. The traffic contract at the UNI will consist of a connection traffic descriptor and a requested QOS class. It also will include the definition of a compliant connection. The connection traffic descriptor consists of all parameters and the conformance definition used to specify unambiguously the conforming cells of the ATM connection, i.e.:

* The source traffic descriptor (e.g., peak cell rate, average cell rate and average burst length)
* The CDV tolerance

Traffic Contract Parameter Specification:

* Peak cell rate is a mandatory traffic parameter in any source traffic descriptor.
* The cell loss rate and QOS class must be explicitly specified in the connection-establishment SETUP message.
* The cell delay variation tolerance is a mandatory parameter in any connection traffic descriptor.

Peak Cell Rate:

The peak cell rate traffic parameter specifies an upper bound on the traffic that can be submitted on an ATM connection. Enforcement of this bound by the UPC allows the network operator to allocate sufficient resources to ensure that the network performance objectives (e.g. for cell loss rate) can be achieved.

Average cell rate:

The average cell rate is an upper bound on the conforming average rate of an ATM connection. Enforcement of this bound by the UPC could allow the network operator to allocate sufficient resources, but less than those based on the peak cell rate, and still ensure the performance objective can be achieved.

Cell Delay Variation Tolerance:

ATM Layer function (e.g. cell multiplexing) may alter the traffic characteristics of ATM connections by introducing cell delay variation. When cells from two or more ATM connections are multiplexed, cells for a given ATM connection may be delayed while cells of another ATM connection are being inserted at the output of the

multiplexer. Consequently with reference to the peak emission time T (i.e., the inverse of the contracted peak cell rate), some randomness may affect the inter-arrival time between consecutive Virtual Path Connection (VPC)/ Virtual Channel Connection (VCC) cells as monitored at the UNI. The upper bound on the "clumping" measure is the CDV tolerance.

I.5 Neural Networks Traffic Measurements

Variable Bit-Rate (VBR) sources produce multimedia traffic that possess high bit-rate fluctuations over relatively short time periods. This traffic is highly bursty and correlated in the sense that its interarrival time has a squared coefficient of variations C^2 (which is the ratio of the variance to the square of the mean value) that is higher than that of the simple uncorrelated Poisson process. For example, it was shown in [12] that the value of C^2 is 18.1 for the packet arrival process due to a single voice source, which is very large considering that of a Poisson process which has $C^2 = 1$. It was also observed that over short time intervals, the superposition process of M voice sources does look like a Poisson process (provided that M is large enough $M > 120$), however over long time intervals, positive correlations exist among the successive packet interarrivals. The cumulative effect of these correlations causes the process to substantially deviate from the Poisson process.

In general, the traffic can be characterized by a point process [13]. It is essential, in ATM networks, to provide efficient admission control and traffic enforcement techniques in order to avoid serious congestion problems. These techniques require

specific knowledge of the statistical-behavior of the input traffic declared via its traffic descriptors [14],[15]. Parameters such as peak bit-rate, average bit-rate, and burst length (which is the duration of the peak bit-rate) are often used as a simple set of parameters that can be used to characterize the multimedia traffic [15],[16]. More complicated second-order time domain parameters (e.g. IDC, IDI, ...etc.) are also used to fully capture the burstiness and the correlations properties of the arrival stochastic process especially those of VBR video and voice sources [17]. Mathematical models such as Continuous Time Markov Chain (CTMC), Semi Markov Modulated Process (SMMP), and Markov Modulated Poisson Process (MMPP) [18] are used to characterize and model the traffic with many approximations that limit their efficacious. Traffic characterization using simple parameters often ignores most of its important correlations and burstiness properties. Hence, leading to the definition of incorrect UPC parameters, that could seriously advert the network performance [19]. The performance of the UPC function can be improved by using more sophisticated parameters that police the probability density function (pdf) of bit-rate of the source [20],[21]. Unfortunately in most cases, this pdf can not be described by a known mathematical model.

Moreover, almost all traffic management and enforcement mechanisms are based upon the assumption that the user must explicitly declare the UPC parameters (such as the peak and average bit-rates). In practice, this assumption is impossible to meet since the user, in most cases, have no explicit knowledge of the connections' traffic statistics. Therefor it is unrealistic to implement any mechanism based upon such conjectures. The practical and more realistic approach is to design a traffic measurement tool that will measure the traffic parameters in real-time. However such tools will not be able to compute second-order moments in real-time. Another alternative to mathematical

approaches is to employ NNs which is the subject of our contribution.

We have introduced a novel approach to this problem by using NNs [22],[23]. NNs can characterize and predict the bit-rate variations of complex stochastic processes using simple parameters such as the number of cell arrivals within a certain time period (count process). The traffic prediction is achieved as the NN learns the actual pdf of the traffic via the count process which keeps track of the number of cell arrivals in a fixed time interval. The traffic measurement function shown in figure (1.2) involves a backpropagation neural network which is used to characterize and predict the statistical variations of the packet arrival process resulting from the superposition of N packetized video sources and M packetized voice sources. The NN is trained to characterize (and predict) the arrival process over a fixed measurement period of time based upon sampled values taken from the previous measurement period. The duration of each sample is chosen such that the correlation properties of the arrival process are captured. The reported results show that the NNs can be successfully utilized to characterize the complex non-renewal process resulting from the aggregate video-packet and voice-packet arrival processes with extreme accuracy. Hence, NNs have an excellent potential as traffic descriptors in admission control and traffic enforcement functions in ATM networks.

I.6 Connection Admission Control

Connection admission control is defined as the set of actions taken by the network at the call set-up phase in order to establish whether a virtual channel connection or a virtual

path connection can be accepted or rejected. The information contained in the traffic contract will be accessible to the CAC function. On the basis of connection admission control in an ATM based network, a connection request is accepted only when sufficient resources are available to establish the connection through the whole network at its required QOS while maintaining the QOS of existing connections.

For each connection request, the CAC function will be able to derive the following information from the traffic contract:

* Values of parameters in the source traffic descriptor
* The requested QOS class
* The value of the CDV tolerance
* The requested conformance definition

Connection admission control makes use of the derived information and the network operator's definition of a compliant connection to determine:

* whether the connection can be accepted or not;
* traffic parameters needed by the UPC;
* routing and allocation of network resources.

I.7 Neural Networks for Connection Admission Control

This section describes the NN connection admission control function shown in figure (1.2). Here are two NNs connection admission control methods: learning control method and hybrid admission method.

1.7.1 Learning Control Method for Admission Control

In the learning control method for the admission control [4], a NN creates a decision function by learning the behavior of the operating multiplexer. The shape of the function is expected to be correctly modified when the characteristics of the multiplexed calls change. Control accuracy will also improve as the NN gains experiences.

The controller uses a 3-layered fully connected NN with backpropagation. Input signals to the NN will be the observed multiplexer status (such as cell arrival rate, cell loss rate, call generation rate, trunk utilization rate, number of connected calls) and the declared parameters contained in a call set-up request (such as average bit-rate and bit-rate fluctuation of the call and holding time). Also a history of the past observed status will be input to the NN in parallel format when the sequence of the data is expected to contain significant information. Output signals are the predicted service quality parameters and the decision values for acceptance or rejection (such as predicted values of cell arrival rate and cell loss rate, and the cell rejection rate)

1.7.2 Hybrid Admission Control

The hybrid admission control [24] method defines a subroutine of CAC called link admission control (LAC), which is performed at each node along the network path connecting the source and destination node. The purpose of LAC is to establish whether the link can accept a new connection request or it should be rejected. The hybrid admission control method uses the NN to improve the link bandwidth efficiency. The scheme comprises a supervised multi layer perception (MLP), combined with a tractable

analytical approximation.

A NN is used to implement an accurate performance model. The multi perception is trained to recognize the nonlinear relationship between a link state vector and a QOS measure calculated by the heterogeneous fluid flow model. The link state vector contains statistical measures of the requested aggregate link traffic and is computational easy to obtain. The link state vector may be based on user-provided traffic descriptor or on traffic measurements that reflect actual user characteristics.

I.8 Usage Parameter Control

Usage parameter control is defined as the set of actions taken by the network to monitor and control traffic in terms of traffic offered and validity of the ATM connection, at the user access. Its main purpose is to protect network resources from malicious as will as unintentional misbehavior which can affect the QOS of other already established connections by detecting violations of negotiated parameters and taking appropriate actions.

UPC Requirements

The operation of the UPC mechanism, utilized by a network operator, will not violate the QOS objectives of a compliant connections. A number of desirable features of the UPC algorithm can be identified as follows:

* capability of detecting any non-compliant traffic situation.
* selectively over the range of checked parameters (i.e., the algorithm could determine whether the user's behavior is within an acceptable region).

* rapid response time to parameter violations.

* simplicity of implementation.

I.9 Neural Networks Policing Mechanisms

Due to the absence of a channel structure in ATM-based networks, the user's traffic can easily overwhelm the network resources leading to serious congestion problems. Traffic enforcement (policing) is required to control the user's traffic to an agreed "contract" determined at the call set-up phase. The policing mechanism controls the traffic via a set of user declared descriptors or parameters such as the peak bit-rate, mean bit-rate, and the burst duration. These parameters are also used by the connection admission control (CAC) algorithm that decides to accept a new connection on the link if the required QOS is guaranteed for the new connection while maintaining the QOS of the existing connections. Hence, at the call set-up phase, a traffic contract is "agreed-upon" between the network and the user. According to that contract, the CAC algorithm maintains a specific QOS for the user's connection as long as the user does not violate his contractual parameters that characterize the traffic. If a violation of this contract (either caused by a malfunctioning terminal or due to a deliberate attempt by the user) is detected, the policing mechanism will enforce the original contractual parameters by an appropriate action. This action can be either cell-dropping [25] or throttling the source bit-rate [26].

The policing mechanism must fulfill the following requirements:

1-It must be capable of detecting and taking a swift control-action, whenever a violation occurs, on real-time basis.

2-It must be simple, and cost effective to implement in hardware.

3-It must be transparent for connections that respect the traffic contract and take no policing actions on their cells. On the other hand, it should act on all cells violating the contract in order to confine the traffic to its contractual agreement.

4-It must have a short dynamic reaction time (which is measured by the delay from the instance a violation occurs till it is detected) in order to avoid the flooding of the relatively small buffers in the network by violating cells.

To meet these somewhat conflicting requirements, several policing mechanisms have been proposed, such as the leaky bucket [27]-[29], source rate throttling [30], window mechanisms [20],[31], and several others (see [32],[33] and the many references there-in).

In summary, any policing mechanism can be classified into a measuring algorithm, that monitors the traffic parameters to be policed (UPC parameters), and an action to be taken when these parameters are violated. Several options exist for the choice of the policed parameters. For example, one might consider a set of parameters such as peak and mean bit-rates. One might, also, choose a set such as the maximum number of cells (X) that can arrive in certain time interval T. There are two conflicting factors that influence the choice of the UPC parameters: 1) The set must be simple enough, such that it can be easily monitored and enforced. 2) The set must fully characterize the traffic bit-rate time-variations in order to enhance the statistical multiplexing gain required for bandwidth allocation in the CAC algorithm.

Most of the existing policing mechanisms attempt to police the peak and the mean bit-rates of the traffic. However the peak and mean policing functions check only one parameter of the pdf of the bit-rate of the source. The fundamental policing problem,

with respect to mean cell-rate policing, is that the characteristics of the source must be estimated in a short time period (sampling period), which gives rise to a certain probability of incorrect policing decision. An extension of the sampling period, on the other hand, increases the reaction time of the mechanism, leading to possible inaccuracies and uncertainties in estimating the traffic characteristics and consequently causing an increase in the margin between the policed bit-rate and the actual bit-rate. Thus the effectiveness of the policing function as well as that of the CAC function is reduced.

A promising policing mechanism is the one that attempts to police the pdf of the traffic [21],[34]. Unfortunately in most cases, this pdf can not be described by a known mathematical model. Moreover it involves complex calculations of higher order moments that can be not achieved on a real-time basis. For example, the Gabarit policing mechanism [21] approximates the pdf by a mathematically well defined distribution (e.g., a Gaussian distribution). However, due to the complexity of its implementation, it is unfeasible to police the Gaussian envelop continuously over all points. Hence, a stair-case shape function is used to approximate the Gaussian envelope leading to errors in the estimation of the policing parameters.

The policing function shown in figure (1.2) involves a NN-based policing mechanism. We have introduced two policing mechanisms using NNs, they are "Neural Network Traffic Enforcement Mechanism (NNTEM)" and "Neural Network Controller Using Reinforcement Learning Method for ATM traffic Policing". These mechanisms are based upon an accurate estimation of the pdf of the traffic via its count process and implemented using NNs. The pdf-based policing is made possible only by NNs. This is due to the fact that pdf policing requires complex calculations, in real-time, at very high

rates which is not feasible via conventional mathematical approaches.

The NNTEM method [35] does not explicitly calculate any higher order moments in order to police the pdf of the traffic. It does that, however, via the NN transfer function which implicitly learns the pdf of the traffic count process through several millions of learning trials. Hence the NNTEM, does not rely upon the policing of simple parameters such as mean bit-rate, peak bit-rate, or burst duration, but rather an elaborate and very accurate function (pdf) which includes all the statistical properties of the traffic. Moreover, the NNTEM actually predicts (through the NN leaning algorithm) any violations in the count process bit-rate violations. If the source traffic violates its "agreed-upon" characteristics declared via its pdf, a violating signal (error signal) is generated. This signal is used to regulate the traffic by dropping the violating cells.

The architecture of the NNTEM is composed from two inter-connected NNs. The first NN is trained to learn the pdf of an "ideal non-violating" traffic, whereas the second NN is trained to capture the "actual" characteristics of the "actual" offered traffic during the progress of the call. The output of both NNs (which is an accurate estimate of the traffic bit-rate fluctuations in the next measurement period) is compared. Consequently, an error signal is generated whenever the pdf offered traffic violates its "ideal" one. The error signal is, then, used to shape the traffic back to its original values.

The Neural Network-based controller using reinforcement learning method for traffic policing [36] is an extension of the NNTEM. Where the error signal produced in NNTEM is fed to another NN-based controller that interprets it, together with some predefined optimization function. The reinforcement learning method is used to tune the weights of that NN-based controller to minimize the violations of the source traffic from its "agreed_upon" characteristics.

The architecture of the Neural Network Controller Using Reinforcement Learning Method is composed from critic function and control function. The critic is composed from the previous NNTEM and a performance measure (cost function). The role of the NNTEM in this case is just to generate the error signal. which is used, by the critic function, to produce an evaluation signal based upon a certain performance measure. A reinforcement learning method uses the evaluation signal to adjust the weights of a third NN, in the control function that generates a control signal capable of maximizing the system performance by dropping the violated cells (i.e., minimization of the traffic violations). The reported results prove that our policing mechanisms are very effective in detecting (and controlling) all possible kinds of traffic violations (e.g., peak and mean bit-rates violations).

Another advantage of the proposed NN policing mechanisms over other ones (such as leaky bucket) is that these mechanisms, from the control systems point of view, are considered as open loop control systems where the traffic is observed before the dropping switch with no provisions for measuring the actual traffic fed to the network. However, our approach is a closed loop control system that measures the traffic after the action of the dropping switch (just before the traffic enters the network). The advantage of the closed loop control system over the open loop one [37], is that the closed loop system controls the disturbances that might have happened to the controlled variable, the traffic in our case, after the controller. Hence our mechanism provides maximum control over the actual traffic entering the network.

1.10 Congestion Control by Neural Networks

The feedback control function shown in figure (1.2) involves an adaptive congestion control model using NNs which can meet the foregoing requirements of congestion control. The presented algorithm is a preventive type and is applied at the access node of the network (i.e., at the User Network Interface (UNI)) [26]. It is a rate-based optimal control implemented by the reinforcement learning NN. The algorithm uses a feedback control signal to throttle the peak bit-rate of the input arrival process to the input statistical multiplexer. The feedback control signal is applied to the input source coder. This signal will control the source rate by decreasing the coding rate (number of bits per sample). The feedback control signal is produced by a NN-based controller. This feedback control signal is an optimal control signal in the sense that it is determined in such way to maximize the system performance. The system performance is measured by a certain performance index which combines two important system performance measures:

1) The input multiplexer buffer overflow (this will reflect the CLR for the accepted calls)

2) The level of the coding rate of the input source(s).

We have solved the optimal control problem using the reinforcement leaning NN method. In this method, the weights of a NN-based controller are tuned continuously in order to maximize the system performance.

Algorithm features

Since the proposed algorithm is a rate-based optimal control implemented by the

reinforcement learning NN. Therefore, it has the advantages of the rate-based control method, the optimal control method, and the reinforcement learning NN method.

The advantages of the rate-based method are as follows:

1) This method is a preventive type congestion control, where it is applied at the input access node to the network, and its speed is not limited by the propagation delay. Hence, any control action will be taken in time to avoid the potential congestion.

2) This method is a closed loop scheme, so unlike the open loop methods (e.g., Leaky bucket, window mechanisms) it measures the level of the cells in the input multiplexer buffer and consequently it does not allow excess cells to enter the network by generating a feedback control signal which is function in the number of the cells in the input multiplexer buffer. Hence this method decreases the CLR and consequently maintains the QOS.

3) This method provides the means of maximum possible shaping of the input arrival process through decreasing the peak bit rate (unlike the cell dropping methods or shaping buffers methods). Consequently the bandwidth allocated to the input call can be reduced and yet the same required QOS will be achieved.

4) The statistical multiplexing gain is enhanced, since more sources can be supported for each multiplexer.

The optimal control nature of the proposed algorithm introduces the following advantages:

1) The proposed algorithm optimizes the system behavior not only by minimizing the CLR but also by maximizing the level of the coding rate.

2) Since the performance index contains two performance measures (the CLR and the level of the coding rate), one can weigh the relative importance of these two performance

measures. For example, one might put more weight on the CLR than on the level of the coding rate, this will lead to decrease the CLR to very low value and decrease the quality of the coding information.

Solving the optimal control problem using the reinforcement learning NN method introduces the following advantage:

An accurate mathematical model for the system to be controlled is not essential in the reinforcement learning NN method. For example to solve this problem using the classical optimal control theory, an accurate mathematical model for the arrival process and for the distribution of the number of the cells in the input multiplexer buffer is mandatory. Any error in that mathematical model will degrade the operation of the controller and consequently lead to an accurate control. More over, this model (if it exists) requires complex computations which are infeasible to be implemented in real time. While, the reinforcement leaning method is based upon improving the performance of the system in terms of the defined performance index and is built using simple model for the system to be controlled.

The reported results of this controller prove its efficacious in both decreasing the CLR of the accepted calls while increasing the quality of the coded information.

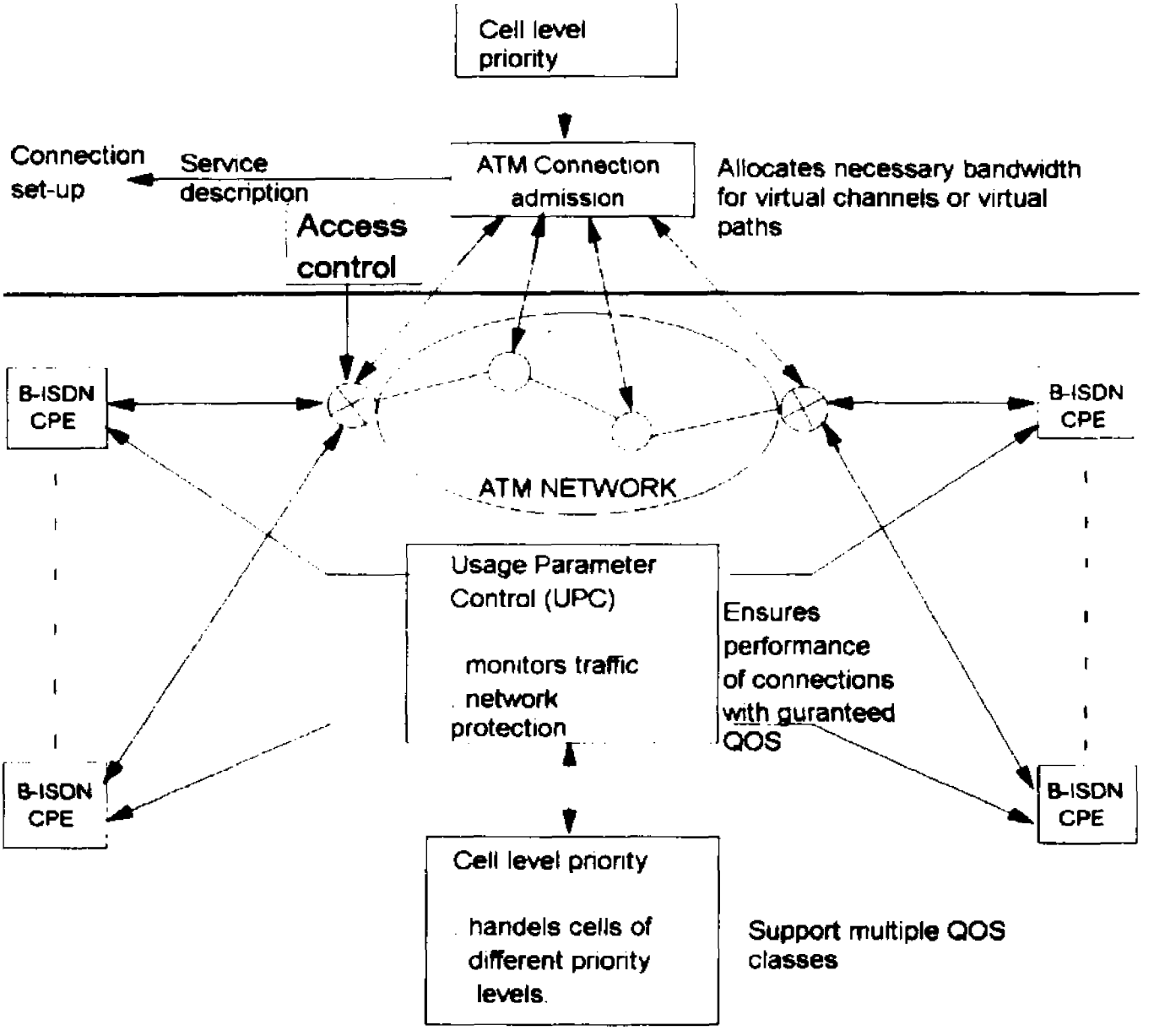


Fig. 1.1 ATM Traffic Management Framework
"Adopted From CCITT"

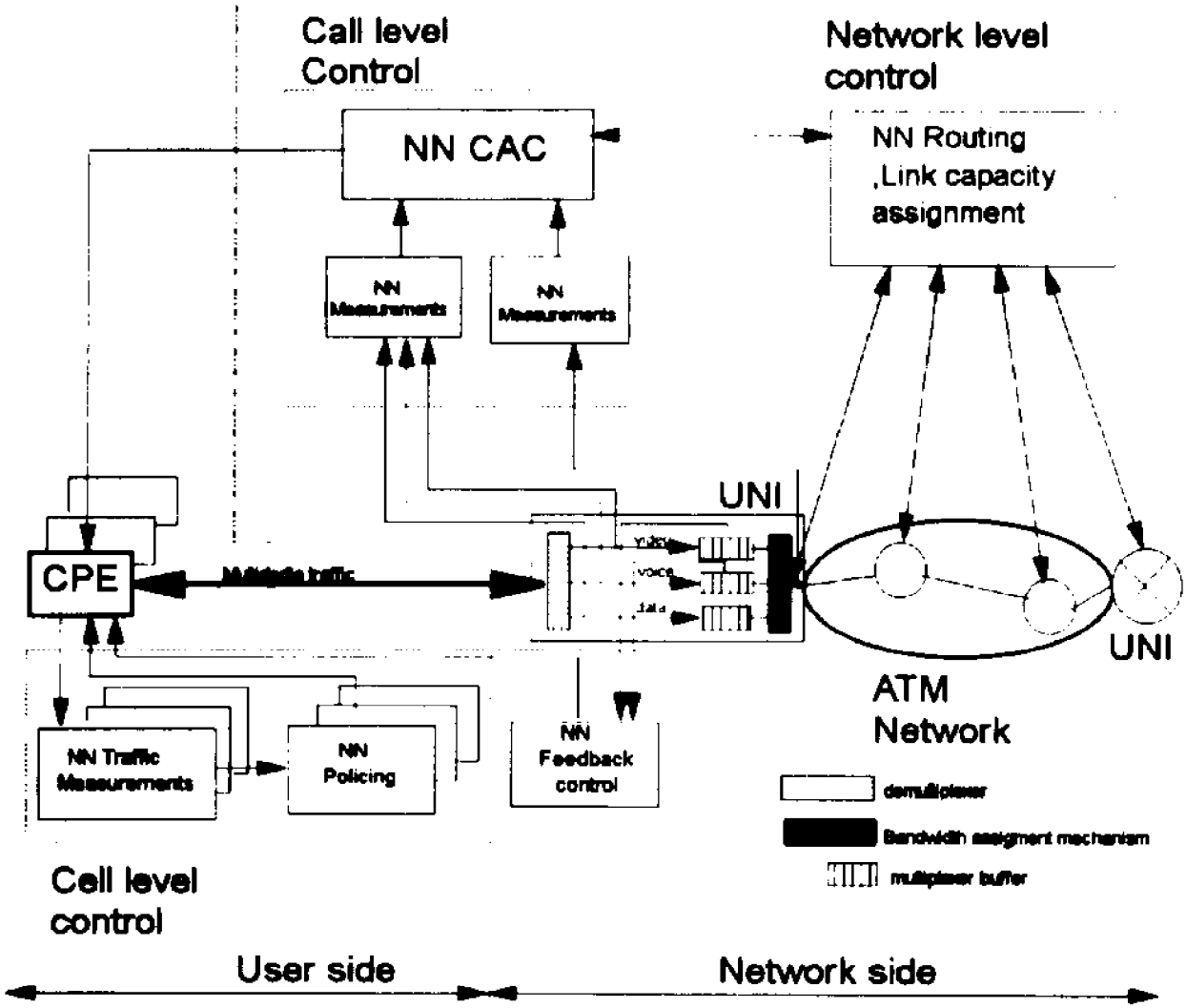


Fig. 1.2 Neural Networks ATM Traffic Management

II. Characterization and Prediction of Packetized Multimedia Traffic in ATM Networks Using Neural Networks

This chapter explains the operation of NNs as traffic descriptors and predictors in ATM networks as shown in figure (1.2). The role of the NN, in this application, is to capture the unknown complex relationship between the past and future values of the traffic. In other word, the NN is employed as an adaptive predictor that learns the stochastic properties of the traffic [22],[23].

II.1 A Neural Network Model For Traffic Characterization And Prediction

Figure (2.1) illustrates the basic idea in training a NN to act as a predictor. The packets' arrival process, from the source(s), is represented by the data vector [H(i+m)] which is the NN target output vector. [H(i+m)] provides the NN with the bit-rate information from which the predictions will be made. The NN predicts the bit-rate variations by exploiting the inherent correlations that exist among the arrivals in the packet arrival process. For training purposes, the input data vector [(H(i)] to the NN, is the delayed value of the data vector [H(i+m)]. The NN, then tries to match the target output data vector [H(i+m)] with its predicted output data vector $[\hat{H}(i+m)]$ by adjusting its weights. It, then, follows that when the input to the NN bypasses the delay unit, the output vector

$[\hat{H}(i+m)]$ is a prediction of the values the traffic will have in the future. The delay unit, shown in figure (2.1), delays its input $[H(i+m)]$ for m time steps. Assuming that the NN requires a negligible amount of time to compute its output from its input, then the NN, after training, provides estimates for the values of traffic $[\hat{H}(i+m)]$ m steps in the future. This approach to adaptive prediction rests on the assumption of a parameterized class of models for functional relationship between the current and past values of the traffic and its later values, or equivalently between earlier values of the traffic and its current values.

Figure (2.2) shows a typical backpropagation NN architecture used in this application. The NN model used is

$$[\hat{H}(i+m)] = NN_f\{[H(i)], [W]\} \qquad (2.1)$$

NN_f denotes the neural network transfer function, where [W] presents the weight matrices of the hidden and output layers. The vector [H(i)] can be used to present the instantaneous values of bit-rate over the past measurement period (T_m) up to the present instant i. Alternatively, the same vector [H(i)] can be used to represent the count process N(0,t) which measures the number of packet arrivals in time (0,t). In this proposal, the count process N(0,t) is used for traffic descriptor in ATM networks due to its robustness against the delay variability of the packets interarrival times. Thus, the vector [H(i)] consists of m samples of bit-rate process or m samples of the count process. These samples are obtained by sampling the arrival process at every sampling period (T_s). $[\hat{H}(i+m)]$ is the output vector from the NN, presenting the expected traffic over the next measurement period (T_m), see figure (2.3). It then follows that,

$$T_m = m * T_s \tag{2.2}$$

The Neural Network input traffic pattern [H(i)] is expressed as

$$[H(i)] = \begin{bmatrix} h(i) \\ h(i-1) \\ \cdot \\ \cdot \\ \cdot \\ h(i-m-1) \end{bmatrix} \tag{2.3}$$

Where h(i) is the value of h(t) at the sampling instant i expressed in packets/sec. The target traffic pattern for the next m measurement intervals [H(i+m)] is expressed as

$$[H(i+m)] = \begin{bmatrix} h(i+m) \\ h(i+m-1) \\ \cdot \\ \cdot \\ \cdot \\ h(i+1) \end{bmatrix} \tag{2.4}$$

where m represents the number the PEs in the NN input and output layers.

The measurement period T_m, and the sampling period T_s have a direct effect on the NN structure and complexity. The NN can characterize the traffic over an arbitrary length measurement interval T_m. However, increasing T_m while maintaining a small number of samples, within it, will give a poor prediction. Because, in this case, the number of samples in the input traffic pattern (equal to m) will not be sufficient to capture the fluctuations of the arrival process. On the other hand, increasing T_m and decreasing T_s (i.e., increasing number of samples) will give an excellent prediction, at the expense of a massive increase in the number of PEs in the NN. Increasing the number of PEs of the NN leads to a prolonged training time of the NN (the training time

of the backpropagation is proportional to the number of PEs). Needless to mention, the complexity of the NN physical realization will be magnified by several orders of magnitude. So firstly, T_m should be selected to give a reasonable prediction window. Secondly, T_s should be selected such that the input traffic vector [H(i)] would reveal the bit-rate fluctuations of the arrival process during the measurement interval. In the mean time, a reasonable number of PEs in the selected NN architecture is maintained.

The proposed model was verified by matching the statistical characteristics of the arrival process to those of the proposed NN model. The arrival process is a correlated non-renewal process and can be characterized by several parameters such as the mean arrival rate, the variance of the number of packet arrivals, the index of dispersion for intervals (IDI), the index of dispersion for counts (IDC), the autocorrelation R(n) of the number of packet arrivals and the third moment of the number of packet arrivals. The variance of the number of packet arrivals in an interval t V(t), the IDC, and the R(n) were used to measure how well the output process, from the NN, matched the actual arrival process. These parameters were chosen since they could best capture the increase in the variance of the arrival process over the sum of consecutive intervals [15].

Let the count process N(0,t) denote the number of packet arrivals in interval (0,t). Let $M_i(t)$ be the ith moment of N(0,t), it then follows that:

$$M_i(t) = E[N^i(0,t)] \tag{2.5}$$

where the variance of the number of arrivals in (0,t), V(t) is given by

$$V(t) = M_2(t) - M_1^2(t) \tag{2.6}$$

The index of dispersion for counts satisfies

$$I(t) = \frac{V(t)}{M_1(t)} \qquad (2.7)$$

The autocorrelation R(n) is defined as follows:

$$R(n) = \frac{E[(x(m) - A)(x(n+m) - A]}{Var[x(m)]} \qquad (2.8)$$

where x(m) is the number of packets arrivals during the m^{th} interval, E[.] is the expectation, var[.] is the variance, and A is the mean of x(m). A detailed calculation of V(t) , I(t), and R(n) is given in the next section

II.2 Characterization of Packetized Voice Traffic

This section reports the NN characterization of voice traffic, where the offered traffic to the neural network is selected to be the arrival process resulting from superposition of M packetized voice sources. This process possesses positive correlations among the packet arrivals in adjacent time intervals. The complexity of this process stems from the bursty nature of the packet arrival process from single voice source [18]. Figure (2.4) shows a typical ON/OFF periodic process due to a single packetized voice source. The ON and OFF time periods are assumed to be exponentially distributed random variables with mean $1/\alpha$ and $1/\beta$, respectively [18]. During the ON period a fixed number of packets is generated, each having duration of T milliseconds.

In order to best select the sampling interval T_s, a number M of voice sources were simulated and their aggregate arrival process was observed at every T_s. Several experiments had been performed using different values for T_s . Figures (2.5) and (2.6)

show the arrival process sampled at T_s = 10 msec, and T_s = 100 msec, respectively. The power spectral density [38] for the sampled traffic was calculated for each experiment. It was found that the maximum frequency component of the autocorrelation function of the arrival process to be 30 Hz as illustrated in figures (2.7) and (2.8). In this work, the sampling interval T_s was selected to be 10 msec corresponding to a sampling frequency of 100 Hz (that is greater than twice the maximum frequency component). This choice guarantees that the used sampled version of the arrival process captures all correlations contained in the actual process. Figures (2.7) and (2.8) show the power spectral density of the arrival process sampled at T_s=10 msec, and T_s=100 msec respectively.

II.2.1 Simulation Results

Extensive simulations were performed to obtain the NN data set, for both training and production phases, and also to assess the performance of various NNs architectures. The packet arrival process, resulting from superposition of M packetized voice sources, was simulated on a Sun Sparc station 330 using the C language.

In the simulation model, the packet generation process of each voice source was modeled exactly as shown in figure (2.4) with fixed packetization period T=16 msec. The peak bit-rate of the voice source is 32 kbps. Two burstiness sets for this model were used, set B1 and set B2. For set B1: the mean active period ($1/\alpha$) and the mean silence period ($1/\beta$) are 350 msec and 650 msec respectively. In that case the mean bit-rate is 11.2 kbps (that is (350/(350+650))*32). Whereas set B2 has $1/\alpha$=200 msec, and $1/\beta$=300 msec, hence, the mean bit-rate in this case is 12.8 kbps.

To generate the NN input data set, the simulation model was run for 30 minutes and about 10 million packets were generated. The prediction window size (T_m) was chosen to be 50 msec. Hence, with $T_s = 10$ msec, the length of each of the vectors [H(i)] and [H(i+m)] is 5. The training data consisted of numerous numbers of [H(i)] and [H(i+m)] vectors observed over the simulation run time.

Two simulation runs were performed. In the first one, [H(i)] and [H(i+m)] were generated by sampling the arrival stream, at every T_s , using the bit-rate as a traffic descriptor. Whereas in the second one, [H(i)] and [H(i+m)] were generated by sampling the count process N(0,t) at every T_s. Also, during the second run, the variance of the number of arrivals V(t), the IDC, and R(n) were calculated by dividing the simulation time into adjacent intervals each of length KT_s where (K=1,2,....). Let N_{KL} be the number of packet arrivals in the Lth interval. Since the variance is defined as the difference between the mean of the squared and the squared of the mean. Then the variance $V(KT_s)$ is:

$$V(KT_s) = n_K^{-1} \sum_{L=1}^{n_k} N_{KL}^2 - \left(n_K^{-1} \sum_{L=1}^{n_k} N_{KL} \right)^2 \qquad (2.9)$$

where n_K is an integer division of the simulation time by K [12]. The first term in the left hand side of above equation presents is the mean of the squared of N_{KL} , whereas the second term presents the squared of the mean of N_{KL} . The IDC is defined as the variance over the mean. Then the IDC can be given by

$$I(KT_s) = \frac{V(KT_s)}{\left(n_k^{-1}\sum_{L=1}^{n_k} N_{KL}\right)} \qquad (2.10)$$

The autocorrelation R(n) was calculated by forming a vector (V) such that the elements of this vector present the number of packets arrivals during the consecutive intervals (each of length T_s). The XCORR(.) and COV(.) functions of the MATLAB signal processing tools [39] were used to calculate R(n) as follows:

$$R(n) = \frac{XCORR(V,'biased')}{COV(V)} \qquad (2.11)$$

The NNs were simulated using HNC Inc. EXPLORENET 3001 package [40]. Several experiments were performed using various backpropagation NNs architectures. During the work-course of this section three NNs architectures were used, they are called NN-1, NN-2, and NN-3. NN-1 has 5 PEs in the input layer, 5 PEs in the hidden layer, and 5 PEs in the output layer, referred to as (5,5,5), while NN-2 is (5,10,5) and NN-3 is (5,20,5). The activation functions of these NNs were selected to be sigmoid where the steepness parameter (a factor determines the non-linearity of the sigmoid function) was selected by trial and error method to obtain good results.

II.2.2 Numerical Results

In this section, we demonstrate the validity of the proposed NN model by comparing the numerical results obtained from this model to those obtained from the simulation model. Four experiments were performed to produce the results.

Experiment (2.1):

In this experiment the parameter set B1 was used with M = 150 voice sources. NN-1 was used to predict the next five values (m = 5) of the bit-rate arrival process. Figure (2.9) shows the predicted traffic output from NN-1 after 2000 sampling intervals. It is clear that the NN has captured the stochastic behavior of the arrival process. It is very interesting to note that the NN estimate is very close to the actual traffic, with a very small estimation error.

Experiment (2.2):

In this experiment the parameter set B1 was used with M = 150 voice sources. NN-1, NN-2, and NN-3 were used to predict the number of packet arrivals over the next five (m = 5) sampling period. The variance of the number of arrivals and the IDC were calculated for the stream output from NN-1, NN-2, and NN-3 as shown in figures (2.10) and (2.11). It is clear that NN-3 performs better than NN-2, and NN-1 because it has higher number of PEs in its hidden layer (20 PEs). Figures (2.10) and (2.11) illustrate that the predicted traffic has the same statistical characteristics as those of the actual input traffic.

Experiment (2.3):

In this experiment the parameter set B1 was used with M = 50, M = 100, and M = 150 voice sources. NN-3 was used to predict the number of packet arrivals over the next five (m = 5) sampling periods. Figure (2.12) shows the increase in the variance as the number of multiplexed sources increases. It is interesting to note that the IDC of a single voice traffic source is identical to that of M sources over long consecutive intervals, according to the following equation [12]:

$$I(t;n) = \frac{VAR[\sum_{i=1}^{n} N_i(0,t)]}{nE[N_1(0,t)]} = \frac{VAR[N_1(0,t)]}{E[N_1(0,t)]} = I(t;1) \qquad (2.12)$$

where n is the number of the multiplexed sources. It is clear from figure (2.12) that NN-3 is a better predictor that NN-1 and NN-2, thus validating the results of experiment (2.2). These results can be explained as follows [12]: According to equation (2.12), the superposition packet-arrival process can never be regarded as a Poisson process (completely random) over long time intervals. Increasing the number of the multiplexed sources does not decrease the degree of randomness of the arrival process. This is due to the cumulative effect of the positive small correlations from consecutive packet arrivals over a long time period. Hence for a given NN structure, the prediction improves only as the randomness over the same long time period decreases.

Experiment (2.4):

In this experiment, sets B1 and B2 were used with M=150 voice sources. NN-3 was used to predict the number of packet arrivals over the next five (m=5) sampling periods. Figures (2.13) and (2.14) show the variance and IDC for the traffic. NN-3 predicts the traffic of set B1 better than that of set B2. The result can be easily explained similarly to the explanation given for experiment (2.3) previously.

II.3 Characterization of Multimedia Traffic

This section reports NN characterization of multimedia traffic resulting from the superposition of N packetized video sources and M packetized voice sources.

The video traffic is generated by simulating a variable bit-rate (VBR) video source

which comprise mainly head and shoulder video sequences types without scene changes. The picture sequences, produced by the video source, comprise a lot of redundant information, hence a suitable compression technique is employed to remove this redundancy, while maintaining a constant picture quality. A number of coding algorithms are employed to code the video signal such as interframe differential pulse code motion (DPCM), intraframe DPCM, conditional replenshiment (CR), motion compensation discrete cosine transfer (MC-DCT) [41]-[43]. In this section, a simulation of a VBR source employing scene without abrupt movement is used.

The change in the coding bit rate process of a VBR video source is described by a number of mathematical models [44],[45], however, the burstiness of the VBR video source depends upon the compression algorithm and the nature of the video scene. For a scene without abrupt movement (head and shoulders video type), it was proved in [44] that the bit-rate has a bell shaped stationary probability density and has exhibit significant correlations for an interval of several frames. Also, this burstiness depends upon the time scale to evaluate the coded information variation. In this section, the bit-rate encoded information is evaluated frame by frame. Rate variations caused by line scanning is assumed to be smoothed out before packet assembly. In this proposal, the simulation model used to generate the VBR video coded traffic is a continuous-state discrete-time process. A first order autoregressive (AR) Markov process X(n) that takes into consideration the autocorrelation of the sequence is used. The first order AR model does not capture the long-term slow-decaying correlations but it does capture the short-term fast-decaying correlations. The definition of the AR process is as follows [44]:

$$X(n) = aX(n-1) + bG(n) \qquad (2.13)$$

where X(n) is the bit-rate during the n th frame, G(n) is a sequence of independent Gaussian random variables, where a and b are constants.

II.3.1 Simulation Results

The packet arrival process, resulting from superposition of N packetized video sources and M packetized voice sources, was simulated on a Sun Sparc station 330 using the C language.

In order to select the sampling interval T_s, typical N video sources are simulated and their aggregate arrival process is observed at every T_s. We selected a (VBR) video source that produces 30 frame/sec. Several experiments have been performed using different values for N and T_s as shown in figures (2.15) and (2.16). The power spectral density for the sampled traffic was calculated for each experiment as shown in figure (2.17) through (2.19). It was found that the maximum frequency component of the autocorrelation function of the arrival process to be 15 Hz. In this research, the sampling interval T_s , for the video traffic, was selected to be 1/30 sec. (1/number of frame per second). This result is expected since the a video source produces 30 frame/sec, and the coded information is taken to be constant during the frame. This means that observing the video traffic at every 1/30 sec will guarantee that the obtained sampled version of the video arrival process captures all correlations contained in the actual video traffic. For the voice traffic it is shown in section II.2 that 10 msec. sampling interval is sufficient to capture all variations of the voice arrival process. For the multimedia traffic resulting

from multiplexing of N video sources and M voice sources, it is clear that sampling interval of 1/30 sec will not be sufficient to capture the voice traffic fluctuations. However, taking the sampling interval $T_s = 10$ msec ensures that the instantaneous variations for both video traffic and voice traffic will be captured.

In the simulation of the video source the parameters used in this model are [44]: $a = 0.8781$, $b = 0.1108$, the mean of G(n) is 4.3 Mbps, where the variance of G(n) is unity.

In the simulation of the voice source, the packet generation process of each voice source is modeled exactly as in section II.2. The parameters used in this model are: mean active duration $1/\alpha = 350$ msec, mean silence duration $1/\beta = 650$ msec, fixed packetization period $T = 16$ msec.

During the course work of this section, four NNs architecture have been used, they are labelled NN-4, NN-5, NN-6, and NN-7. NN-4 has 1 PE in the input layer, one slab in the hidden layer of 5 PEs, and 1 PE in the output layer, referred to as (1,{5},1), and NN-5 has 1 PE in the input layer, two slabs in the hidden layer (the first slab has 10 PEs, and the second slab has 5 PEs), and 1 PE on the output layer. NN-5 is referred to as (1,{10,5},1), while NN-6 is (5,{10,5},5), and NN-7 is (5,{20,15,10},5). The activation functions of these NNs were selected to be sigmoid where the steepness parameter was selected by trial and error method to obtain good results in the sense of matching the values V(t), IDC, and R(n) of the predicted process to those of the original process.

II.3.2 Numerical Results

In this section, we demonstrate the validity of the proposed NN model by comparing the numerical results obtained from this model to those obtained from the simulation model. Five experiments were performed to produce the results.

Experiment (2.5):

In this experiment, we used N = 1, M = 0 (one video source and no voice sources). NN-4 was used to predict the video count arrival process over the next frame interval (m = 1, T_s = 1/30 sec). Figures (2.20), (2.21), and (2.22) show that the variance, the index of dispersion, and autcorrelation of the predicted video traffic, output from NN-4, match those of the actual traffic with small error. It is clear that NN-4 has captured the stochastic behaviors of the arrival process.

Experiment (2.6):

In this experiment, we used N = 10, M = 0 (ten video sources and no voice sources). NN-5 was used to predict the homogenous superposition video count arrival process over the next frame interval (m = 1, T_s = 1/30 sec). Figures (2.23), (2.24), and (2.25) show that the predicted traffic, output from NN-5 has the same statistical characteristics as those of the actual input traffic. It is clear that NN-5 performs better than NN-4 because it has higher number of PEs in its hidden layers.

Experiment (2.7):

In this experiment, we used N = 1, M = 1 (one video source and one voice source). NN-5 was used to predict the multimedia (heterogenous superposition process) count arrival process over the next sampling period (m = 1, T_s = 10 msec). Figures (2.26), (2.27), and (2.28) show that NN-5 characterized and predicted the multimedia traffic with extreme

accuracy. It is interesting to observe that NN-5 performance in this experiment is better than that in experiment (2.6), where it was exposed to video traffic only. This can be explained as follows:

The statistics of the multimedia traffic, resulted from superposition of one video source and one voice source, are dominated by the statistics of the single video source traffic as shown in figure (2.29). Where it shows the autocorrelation function of the multimedia a traffic compared wits those of the single video source traffic and single voice source traffic. It is clear that autocorrelation function of the multimedia traffic is very close to that of the single video source traffic. Hence, when we attempt to predict heterogenous multimedia traffic composite of one video and one voice, we actually attempt to predict the video arrival process, and since in this experiment the prediction window size, ($m=1$ with $T_s=10$ msec), was smaller than that for experiment (2.6), ($m=1$, $T_s=1/30$ sec), NN-5 performed better in this case.

Experiment (2.8):

In this experiment, we used $N=1$, $M=1$ (one video source and one voice source). NN-6 was used to predict the multimedia (heterogenous superposition process) count arrival process over the next five sampling period ($m=5$, $T_s=10$ msec). Figures (2.30), (2.31), and (2.32) show that NN-6 characterized and predicted the multimedia traffic with good accuracy. Since the prediction window size has been increased in experiment, we get certain prediction error compared with the results obtained in experiment (2.7).

Experiment (2.9):

In this experiment, we used $N=5$, $M=5$ (five video sources and five voice sources). NN-7 was used to predict the multimedia (heterogenous superposition process) count arrival process over the next five sampling period ($m=5$, $T_s=10$ msec). Figures (2.33),

(2.34), and (2.35) show that NN-7 characterized and predicted the multimedia traffic with reasonable accuracy. By comparing these results with those of experiment (2.6), we can conclude that the homogenous superposition arrival process (as in experiment (2.6)) is simpler to be characterized and predicted than the heterogenous one, since the heterogenous arrival process is more bursty than the homogenous arrival process.

Experiment (2.10): **Real time video signal**

In this experiment, we used a real time video data, to generate the arrival process, instead of the AR simulation model. The real time video data used here is the output of the Motion Picture Expert Group (MPEG) coder [46]. We obtained this data from the video processing and telecommunication lab, Department of Electrical Engineering University of Pennsylvania, where there is MPEG Tool as shown in figure (2.36). Figure (2.37) shows the real time video signal for one video source (N=1) output from the MPEG Tool. Further details of the MPEG Tool can be found in [47].

NN-4 was used to predict the video count arrival process over the next frame interval (m=1, T_s =1/24 sec). Figure (2.38) shows the predicted video traffic produced by NN-4 compared with the actual real time video signal. It is clear that the predicted traffic follows the actual one even during the burst periods. Figures (2.39), (2.40), and (2.41) show that the variance, the index of dispersion, and autcorrelation of the predicted video traffic, output from NN-4, match those of the actual traffic with small error. It is clear that NN-4 has captured the stochastic behaviors of the real time video signal.

This experiment proves that not only the NN approach can characterize the simulated traffic but the real time traffic as well.

Based upon the above experiments, we have concluded that a suitable NN architecture can be chosen to characterize a specific type of traffic. After completing the

training phase of the NN, it learns the pdf of the offered traffic. Hence, the NN can be used as an effective traffic descriptor. It describes the traffic by its actual pdf (instead of the approximated simple parameters such as the peak and mean bit-rates). In ATM networks, traffic management techniques require traffic parameters that can capture the various traffic characteristics. Adaptability to changes in the traffic characteristics is also important for robustness. The proposed model using the NNs is suitable for implementing an effective ATM traffic descriptor, since it can adaptively predict the traffic by learning the relationship between the past and the future traffic variations. The potential of the proposed model is demonstrated by its efficacious to predict the packets' arrival process of the superposition of N video sources and M voice sources. The results show that NNs can be trained to learn the pdf of the arrival process hence it can function as an adaptive predictor.

Our main contributions, in proposing the NN method, is manifested by the fact that our method is a very realistic approach to not only measure but also predict the multimedia traffic. The direct consequence of this achievement is that out method does not need any user declared parameters in order to characterize the traffic. We have stated previously, it is rather conspicuous to assume that the user would declare any parameters about the traffic. Therefore, our novel approach is pioneering work in this direction.

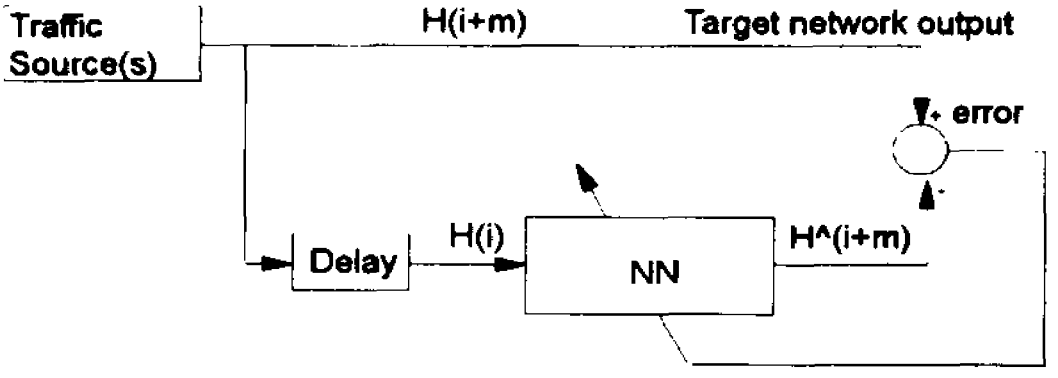


Fig. 2.1 Using Neural Network for adaptive prediction

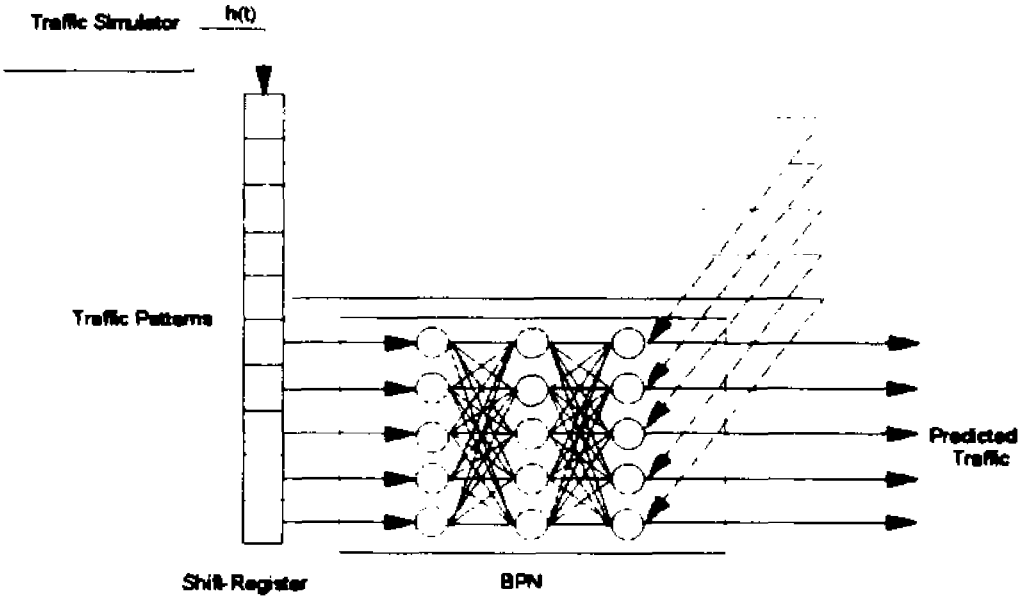


Fig. 2.2 Neural Network for traffic prediction

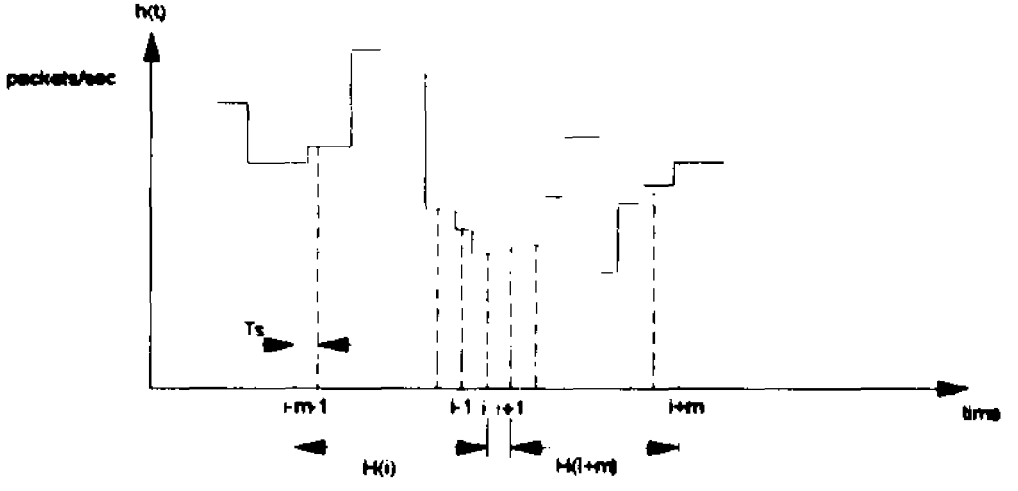


Fig. 2.3 Sampled superposition arrival process

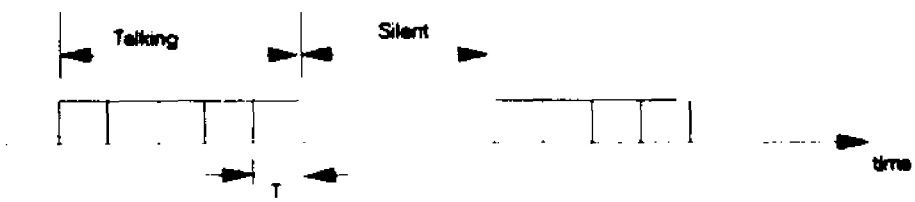


Fig. 2.4 Two phase ON/Off process

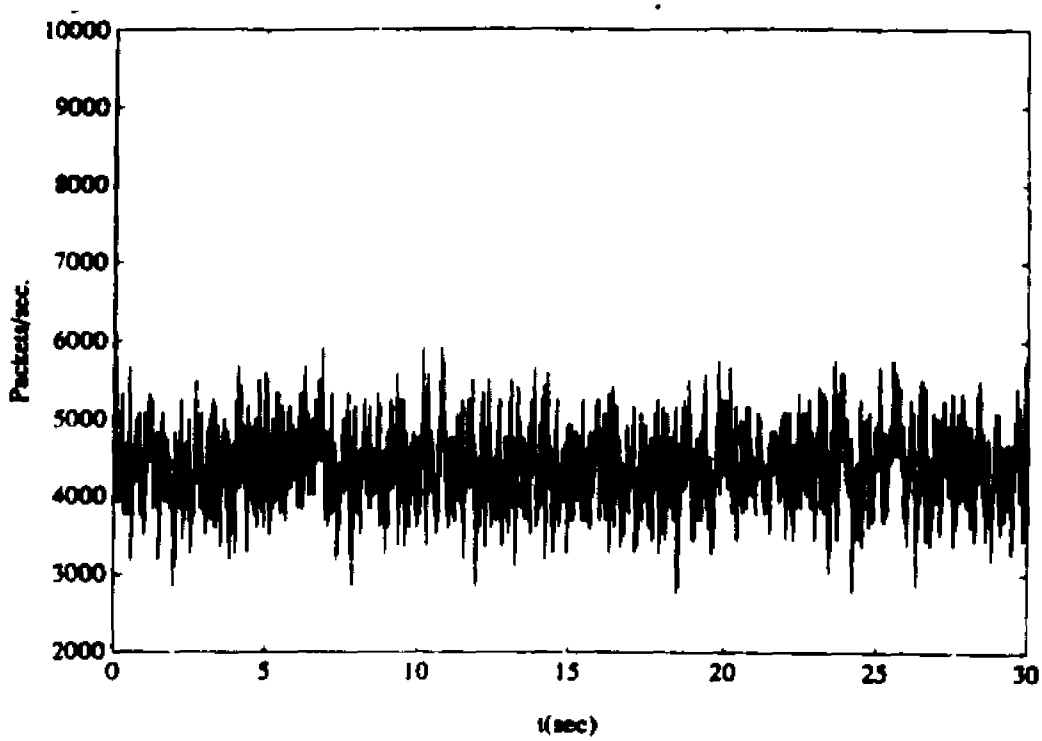


Fig. 2.5 The sampled arrival process (Ts = 10 msec)

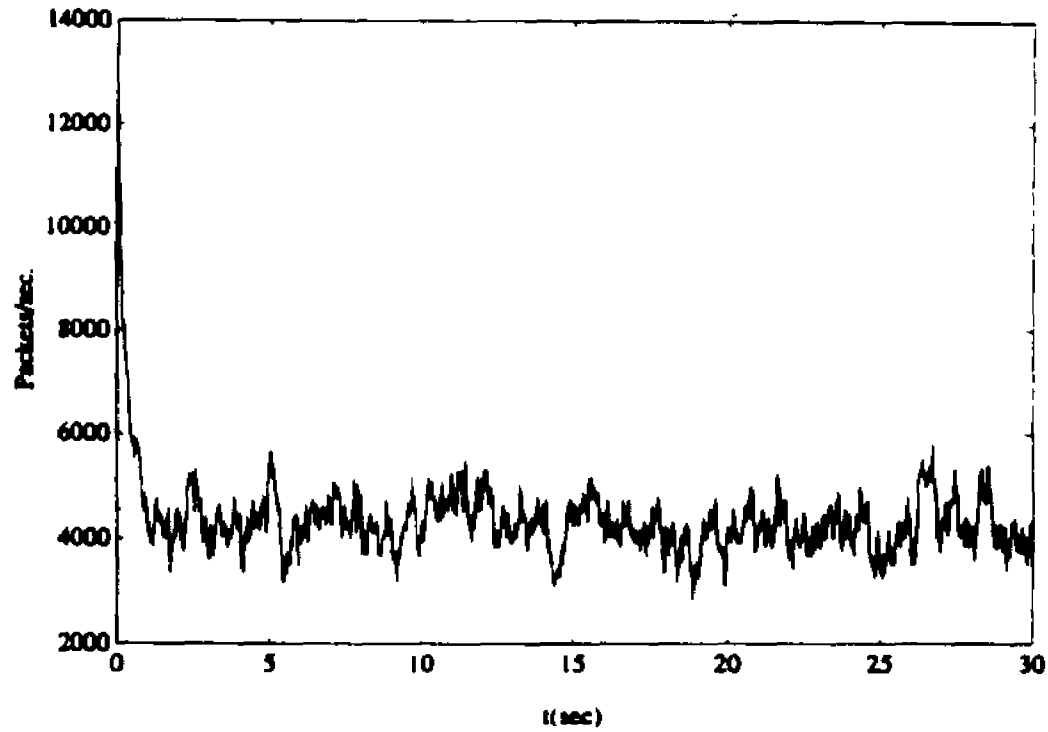


Fig. 2.6 The sampled arrival process (Ts = 100 msec)

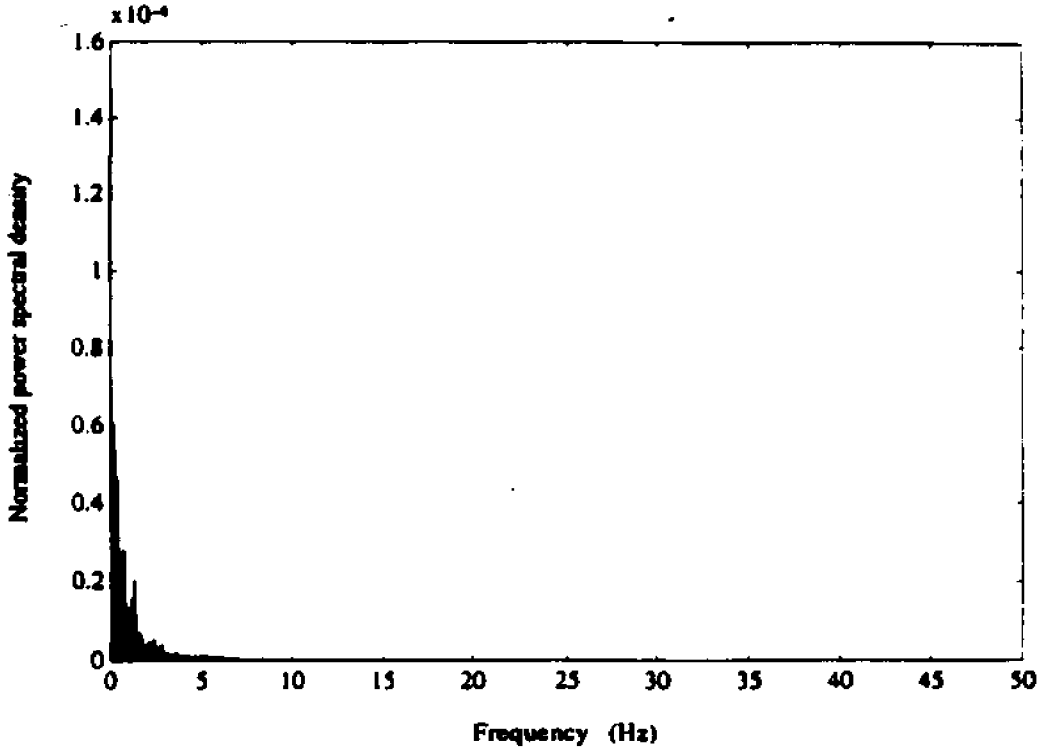


Fig. 2.7 The power spectral density of the arrival process (Ts=10 msec)

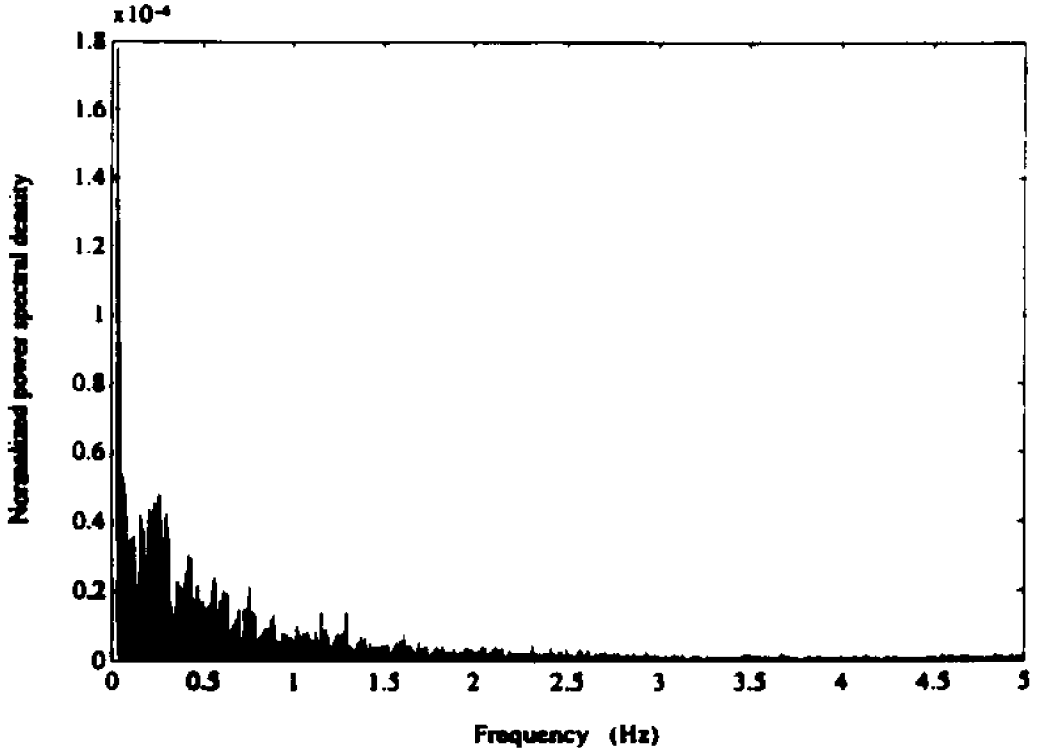


Fig. 2.8 The power spectral density of the arrival process (Ts=100 msec)

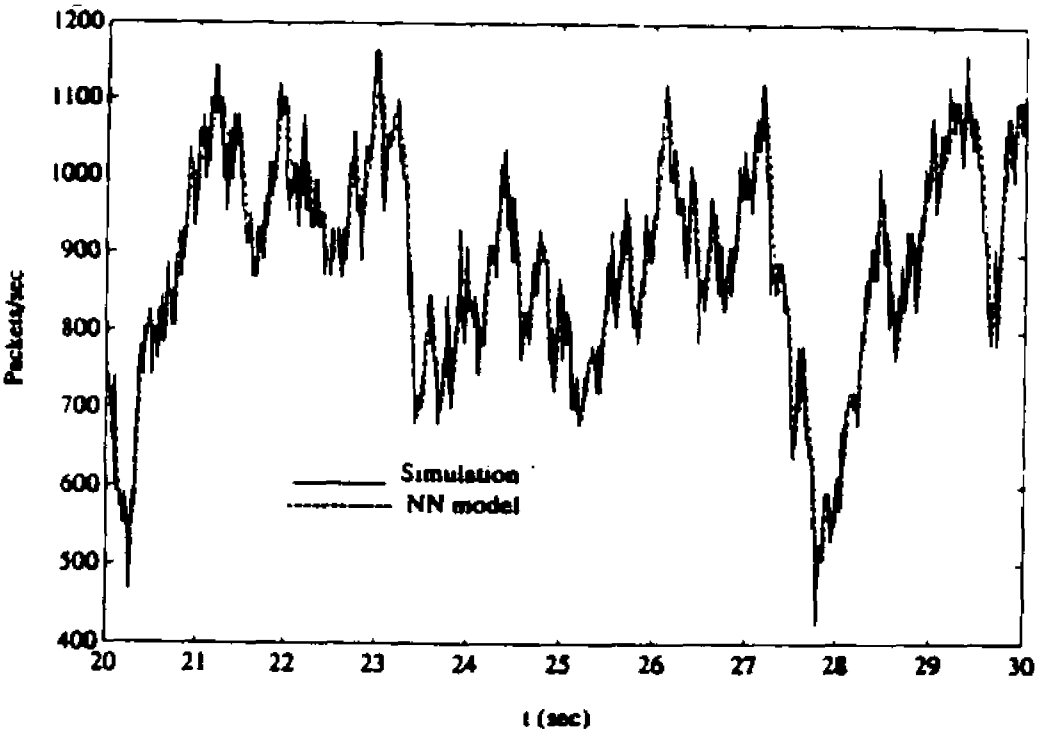


Fig. 2.9 Neural Network prediction for the packet-rate process (NN-1)

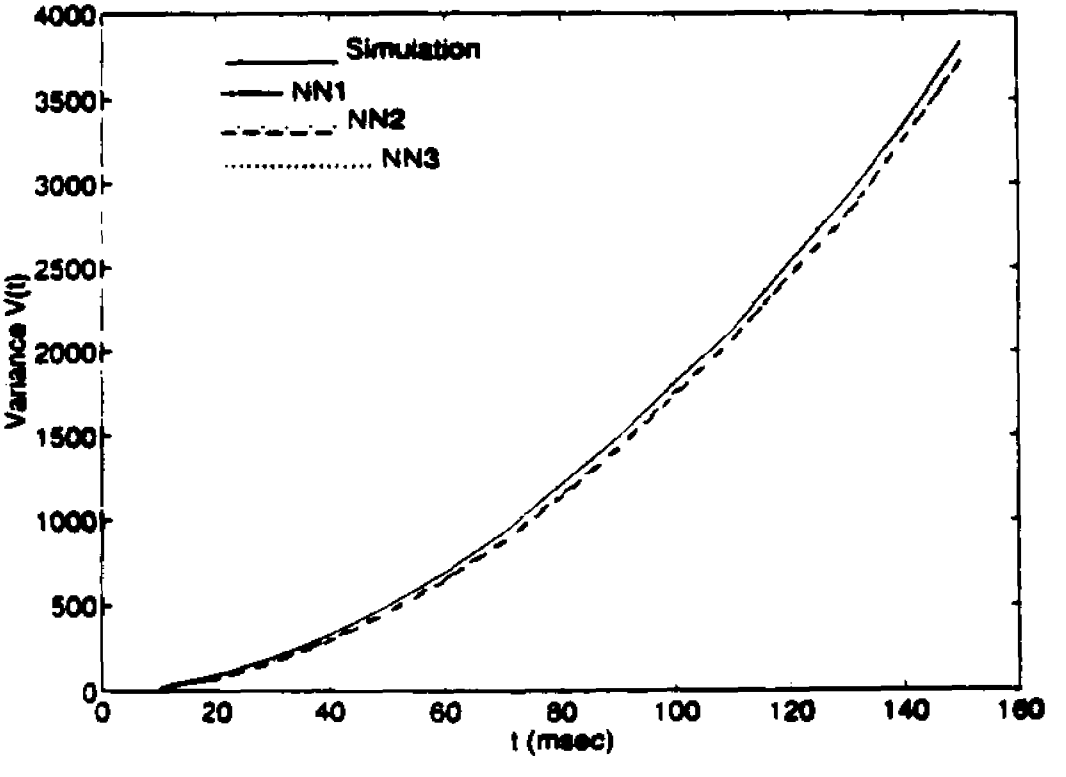


Fig. 2.10 Variance of the number of arrivals in (0,t)

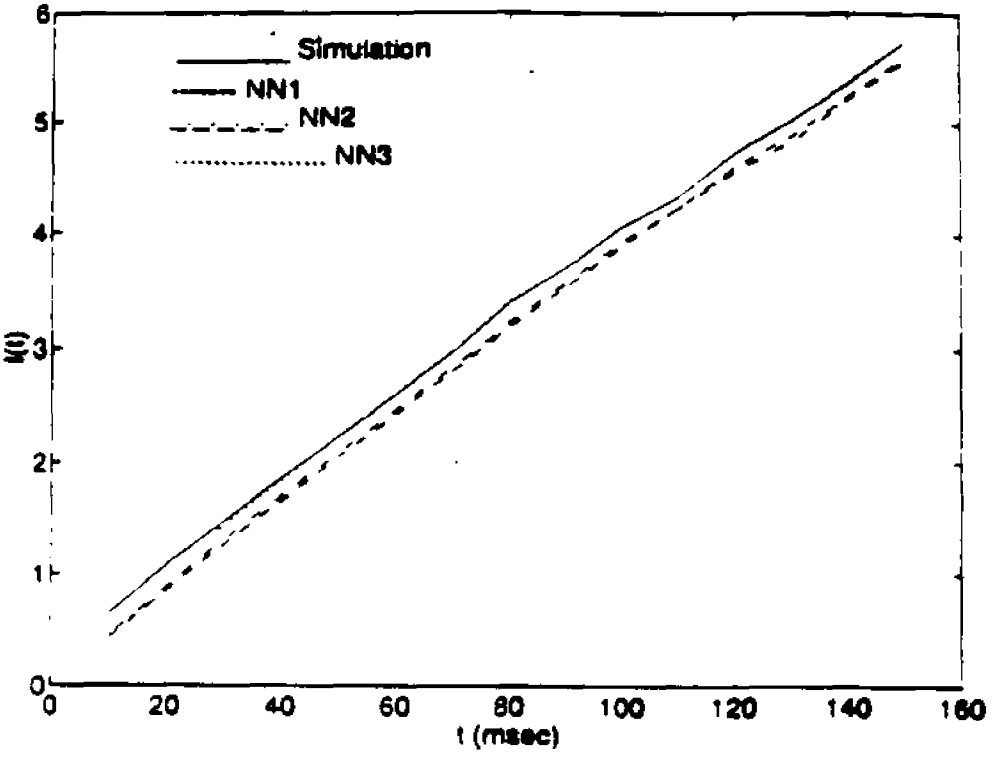


Fig. 2.11 Index of dispersion of the number or arrivals in (0,t)

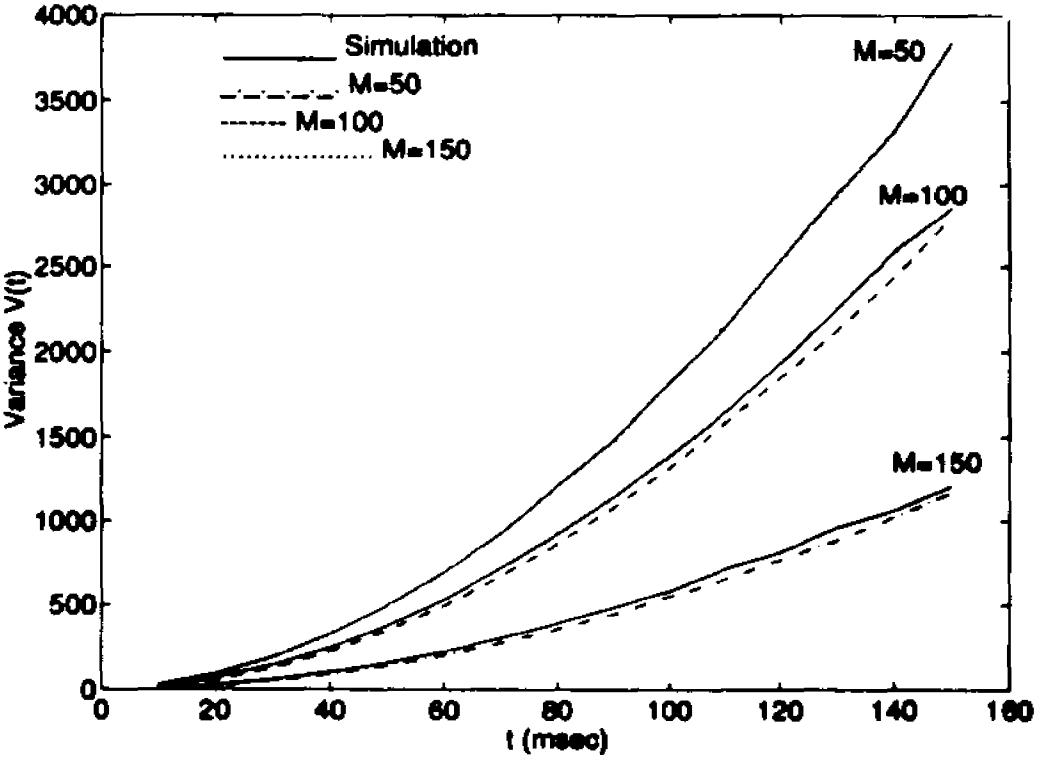


Fig. 2.12 Variance of the number of arrivals in (0,t)

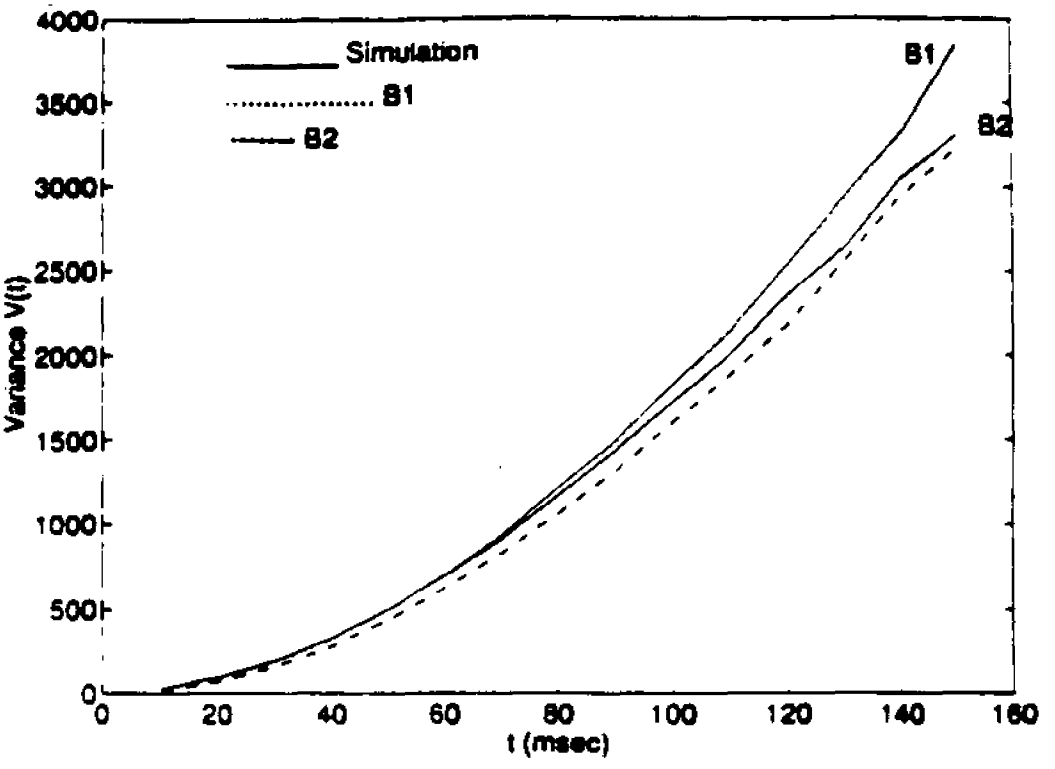


Fig. 2.13 Variance of the number of arrivals in (0,t)

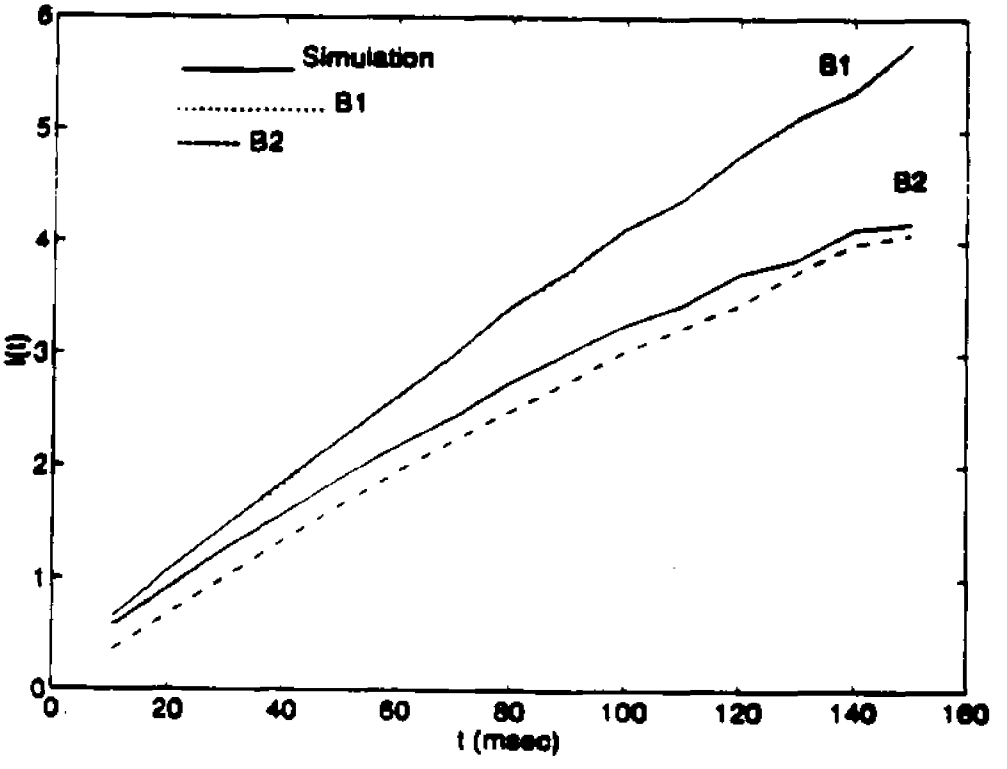


Fig. 2.14 Index of dispersion of the number or arrivals in (0,t)

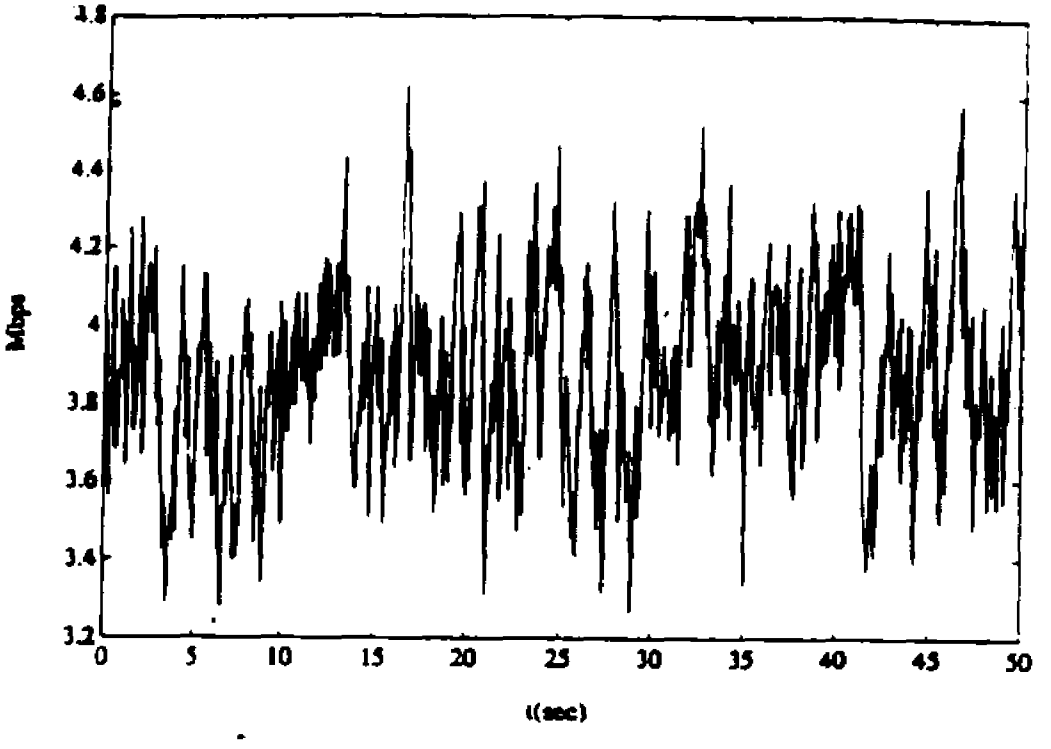


Fig. 2.15 The sampled arrival process (N=1, Ts=1/30 sec)

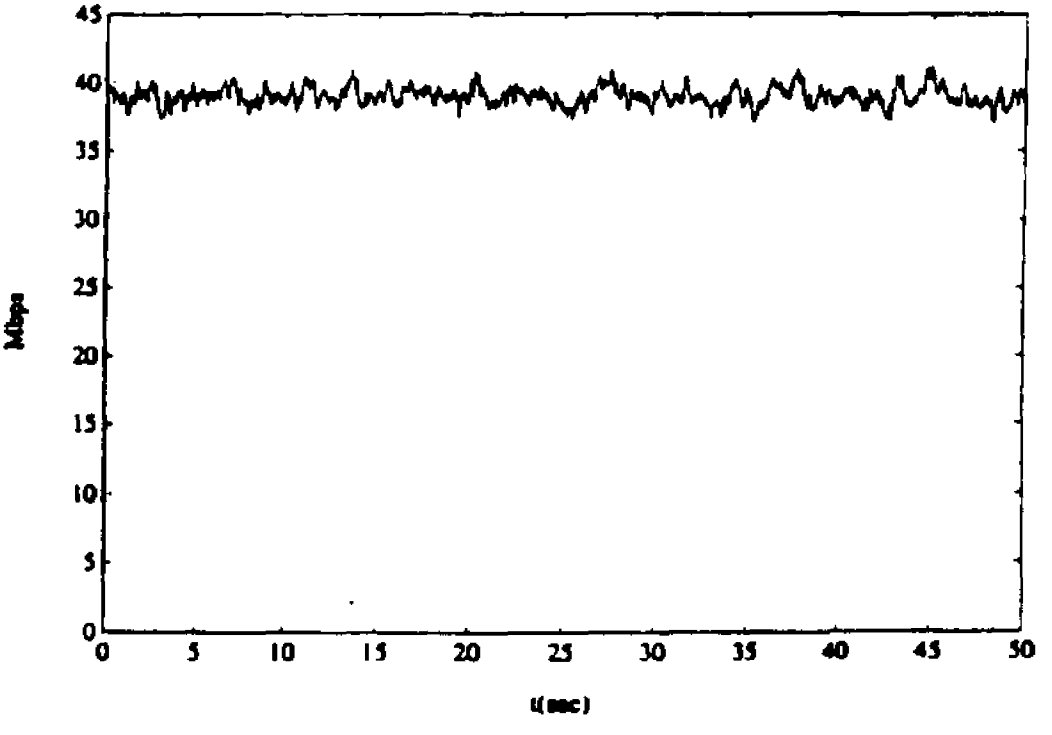


Fig. 2.16 The sampled arrival process (N=10, Ts=1/30 sec)

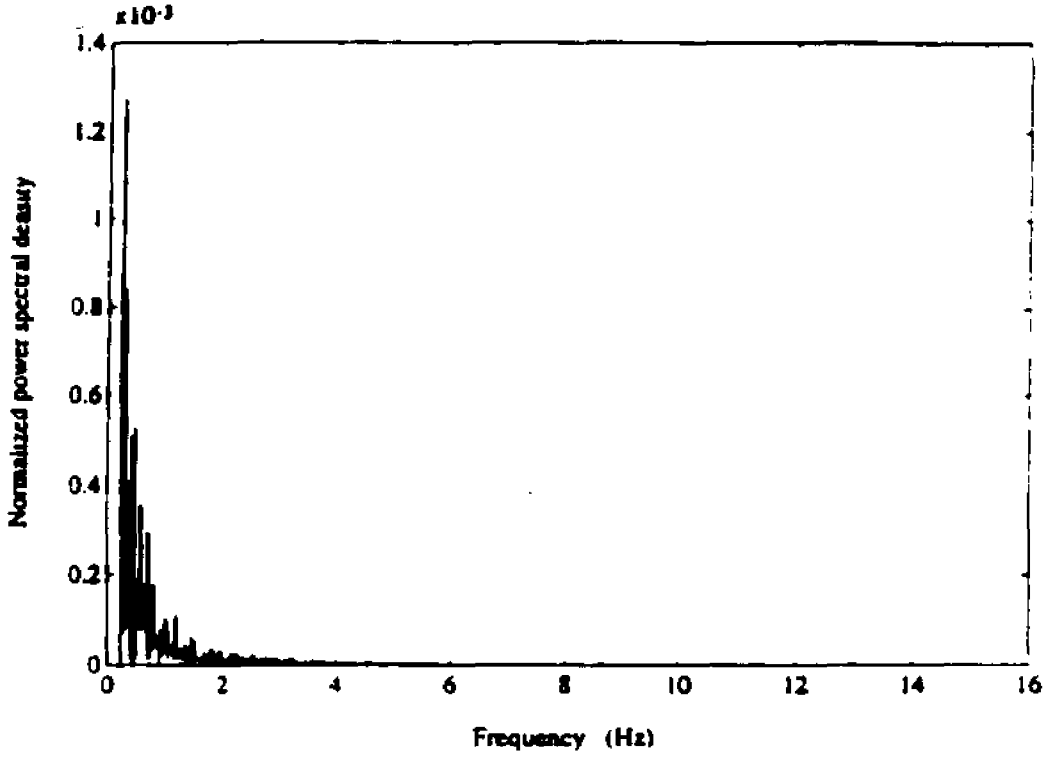


Fig. 2.17 The power spectral density of the video arrival process (N=1, Ts=1/30 sec)

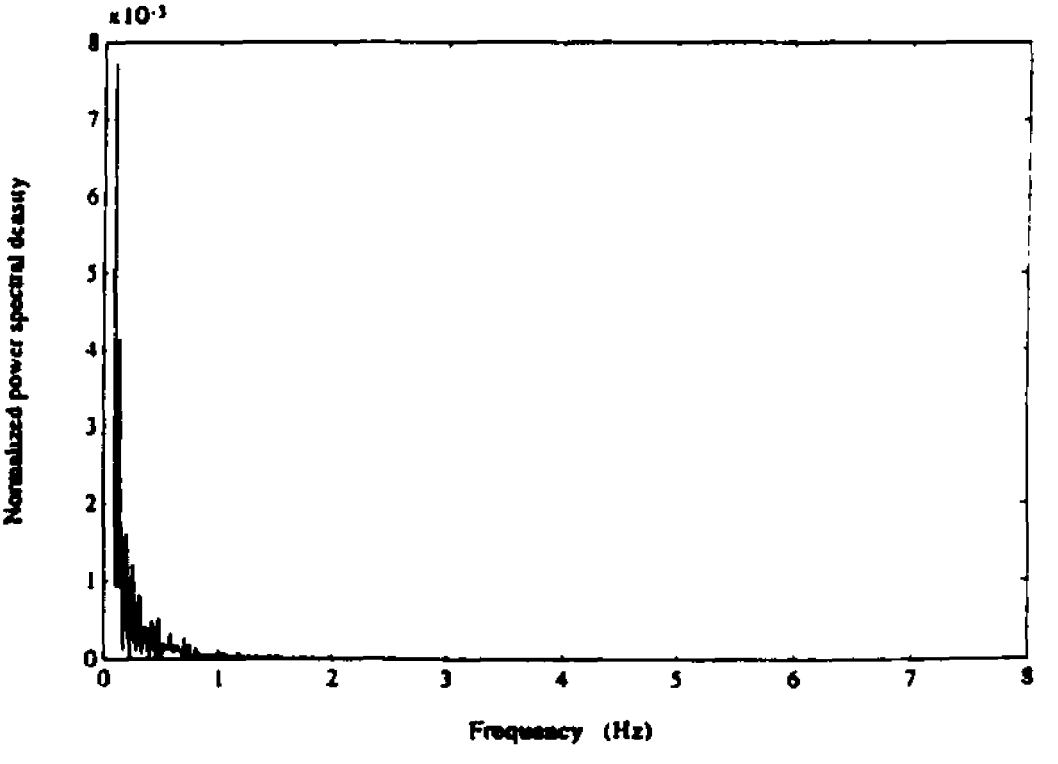


Fig. 2.18 The power spectral density of the video arrival process (N=1, Ts=2/30 sec)

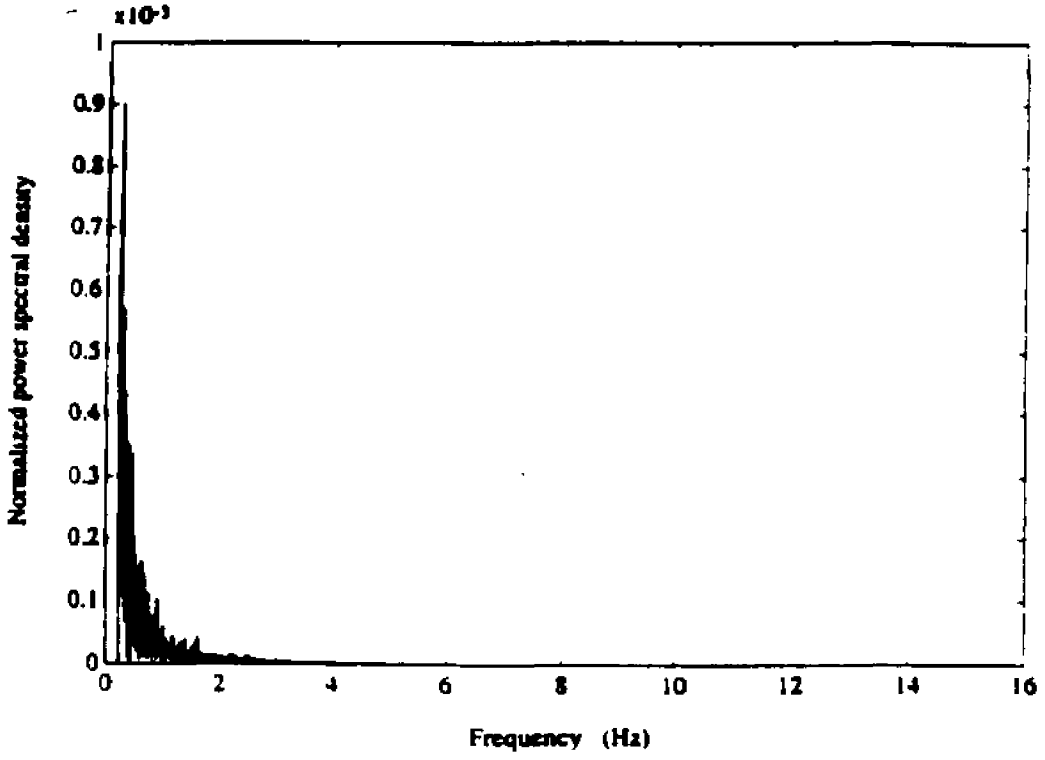


Fig. 2.19 The power spectral density of the video arrival process (N=10, Ts=1/30 sec)

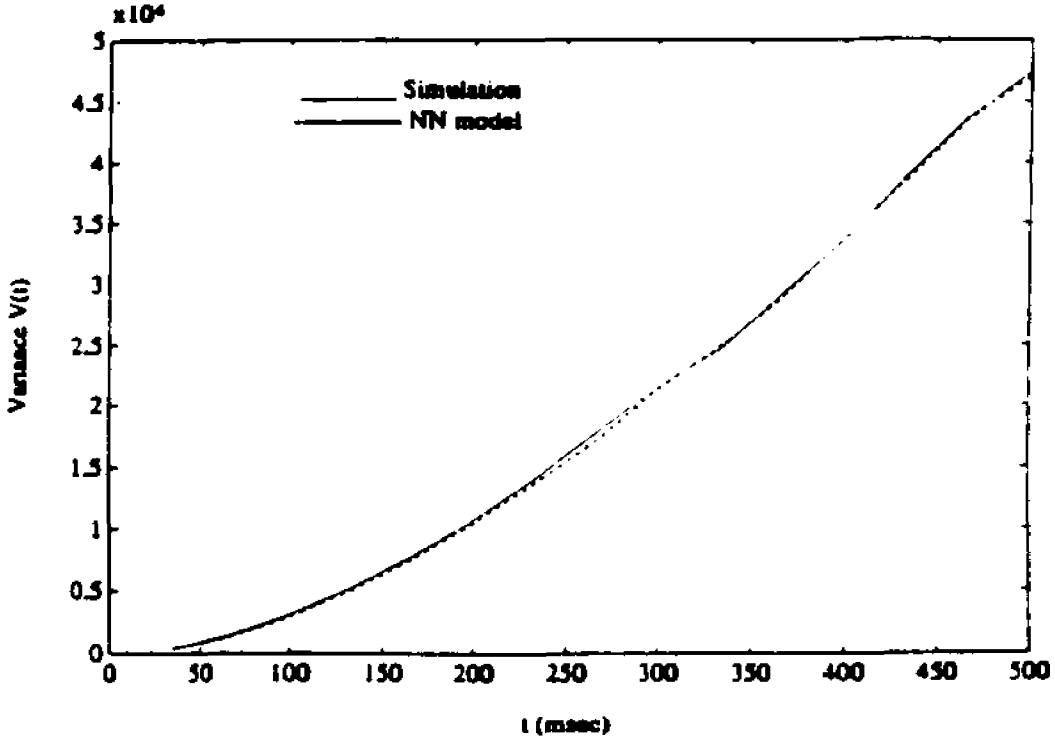


Fig. 2.20 Variance of the number of arrivals in (0,t) (N=1)

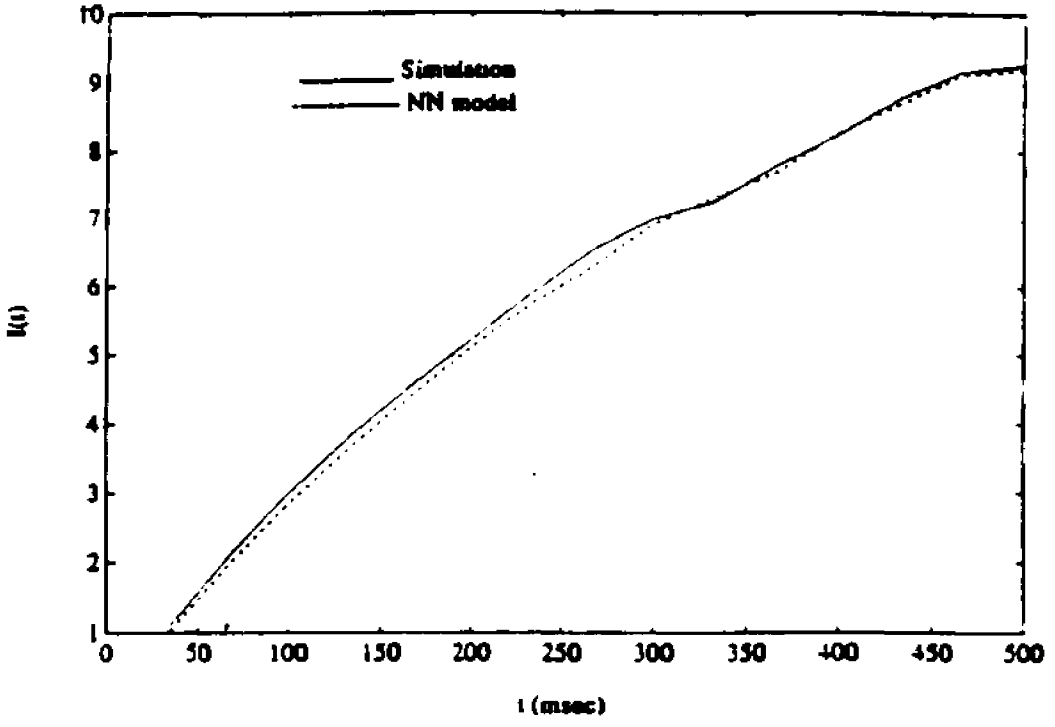


Fig. 2.21 Index of dispersion of the number or arrivals in (0,t) (N=1)

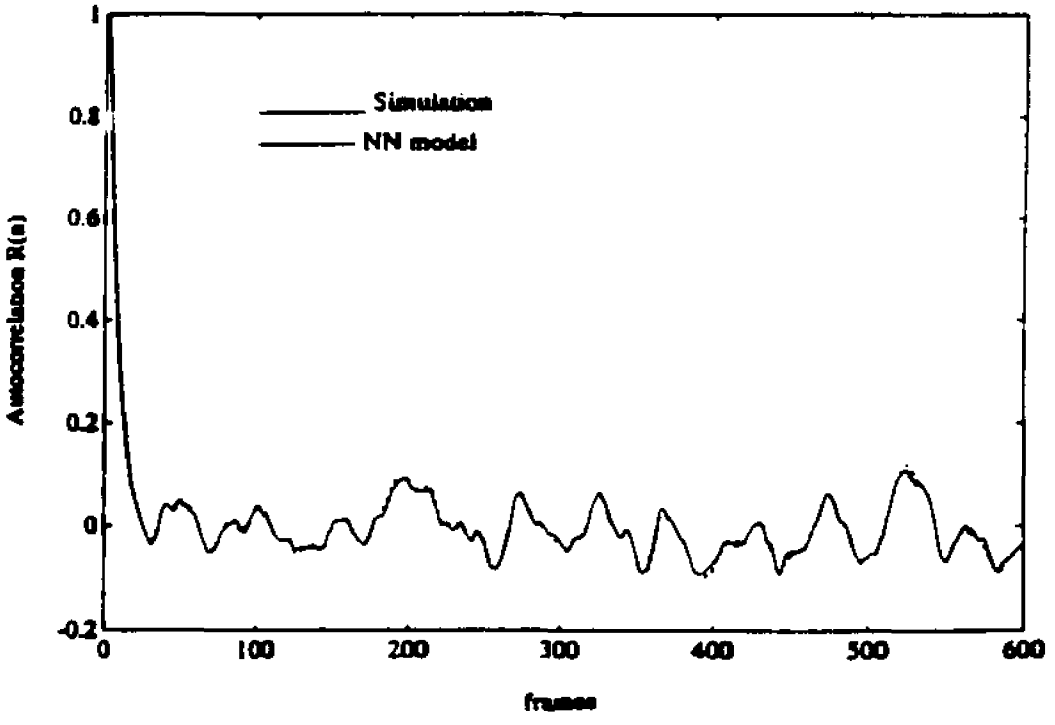


Fig. 2.22 Autocorrelation of the video count process (N=1)

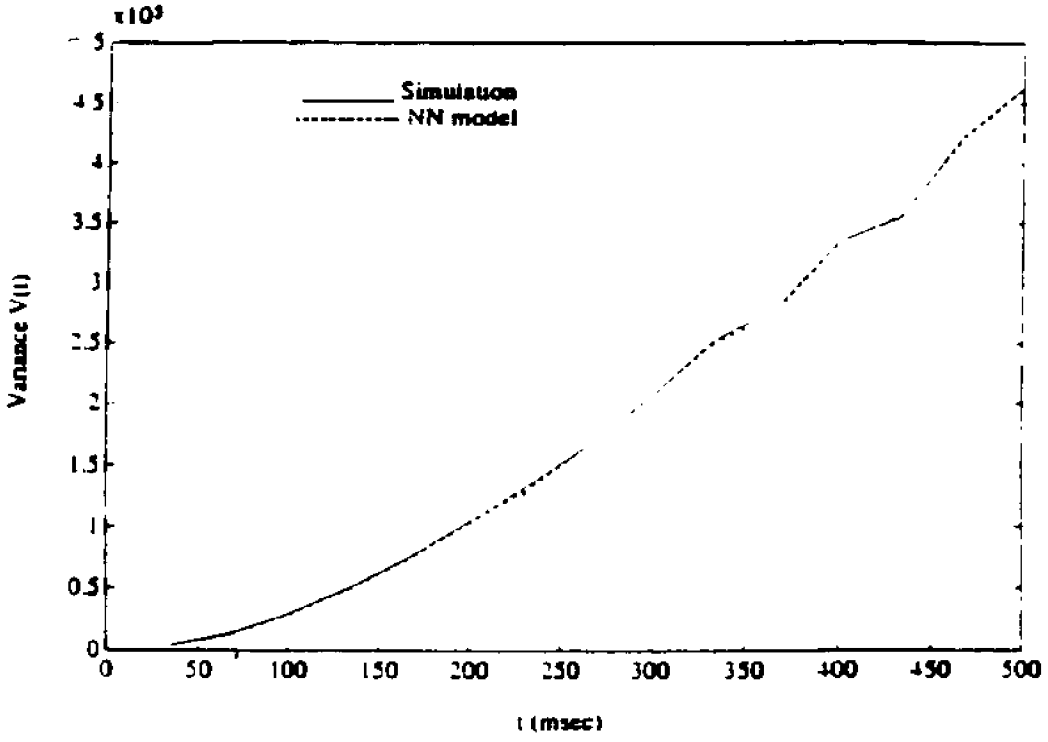


Fig. 2.23 Variance of the number of arrivals in (0,t) (N=10)

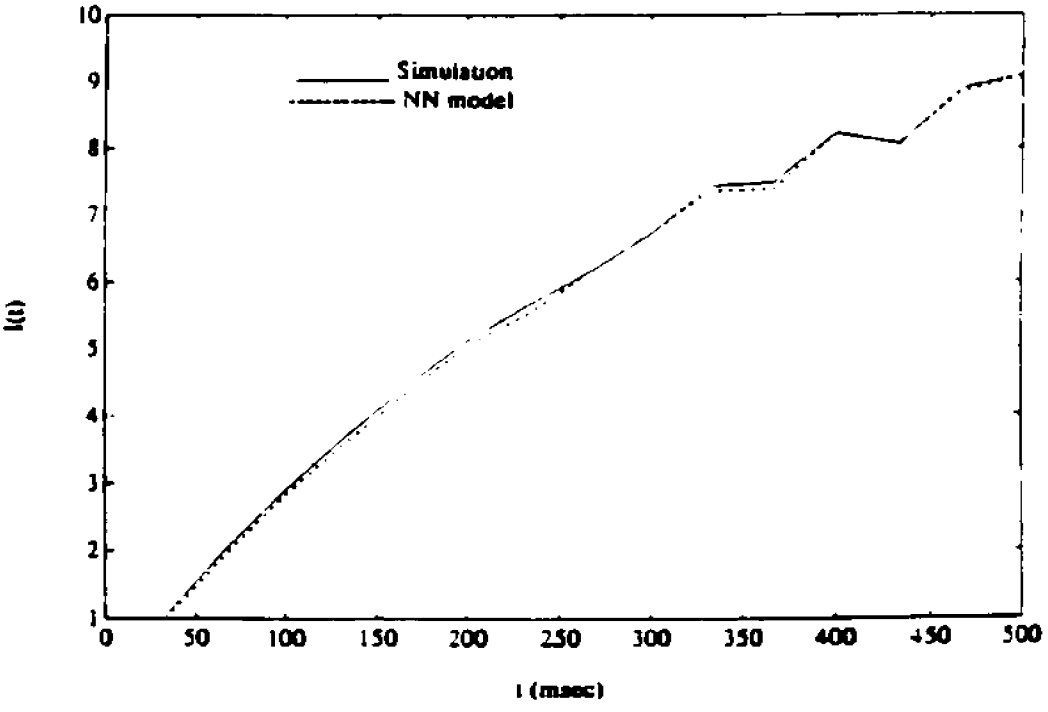


Fig. 2.24 Index of dispersion of the number or arrivals in (0,t) (N=10)

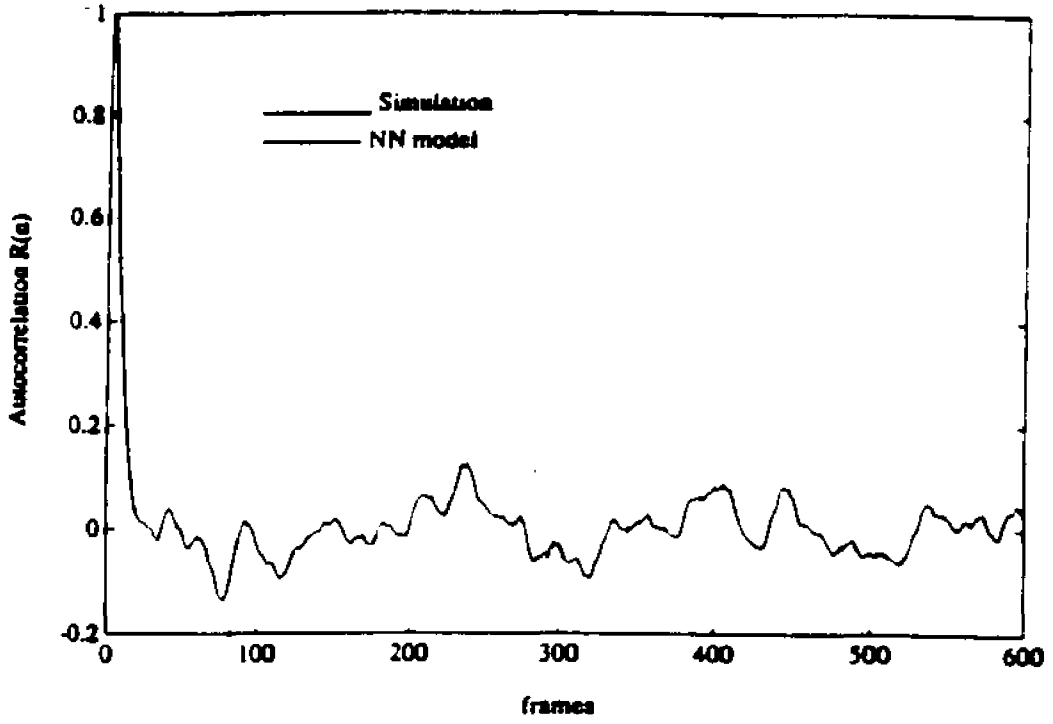


Fig. 2.25 Autocorrelation of the video count process (N=10)

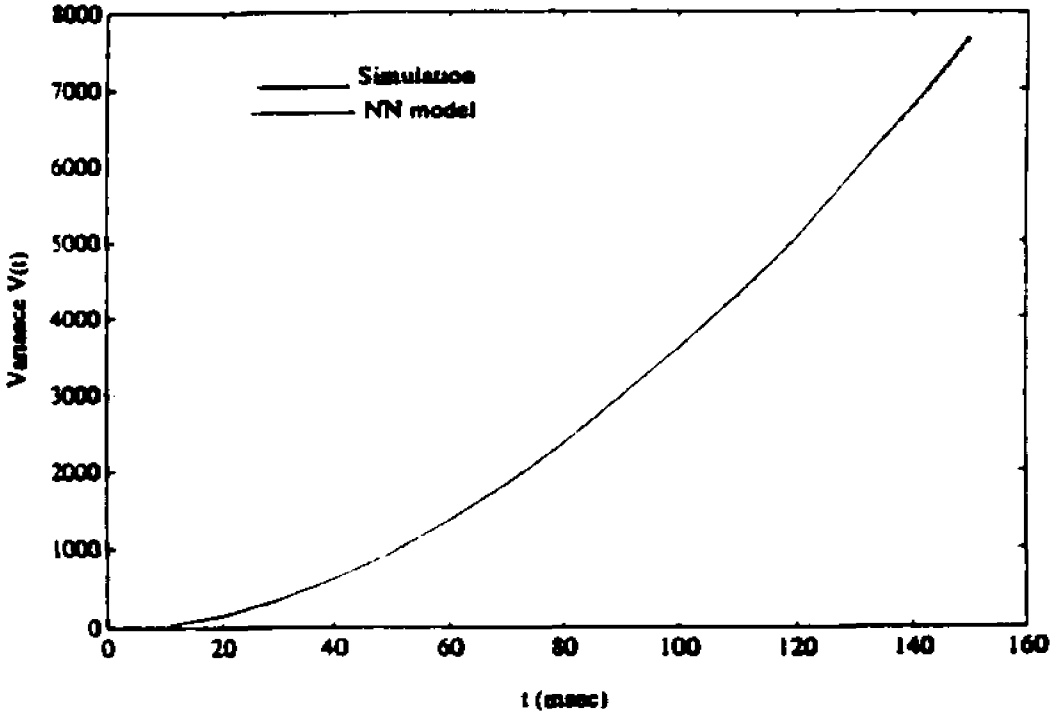


Fig. 2.26 Variance of the number of arrivals in (0,t) (N=1, M=1)

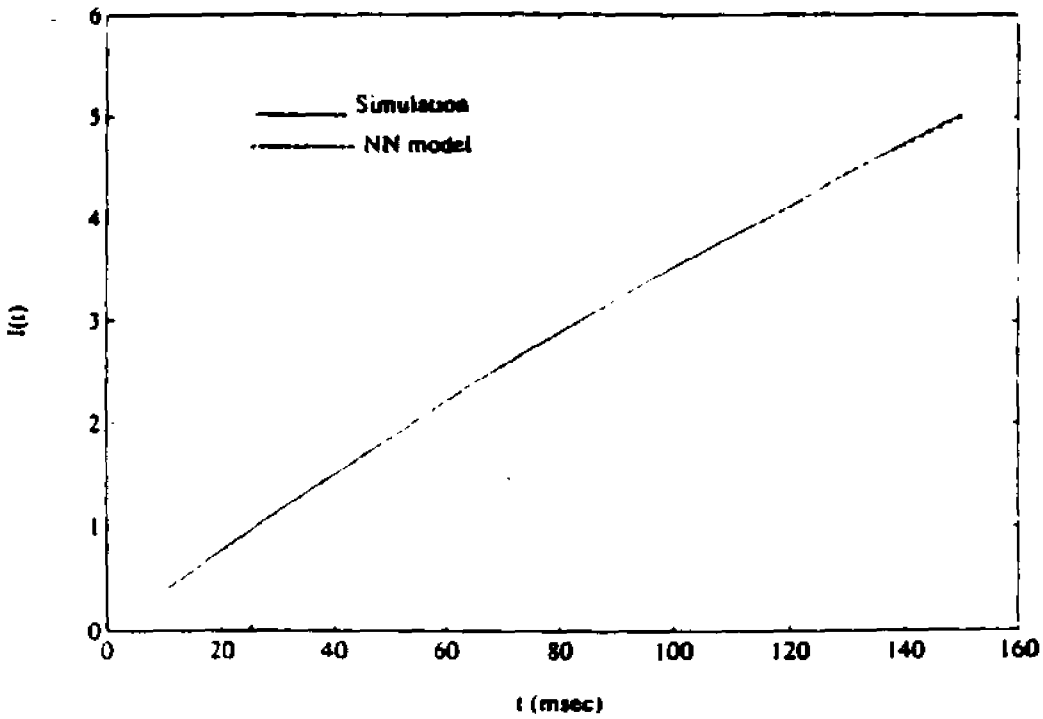


Fig. 2.27 Index of dispersion of the number or arrivals in (0,t) (N=1, M=1)

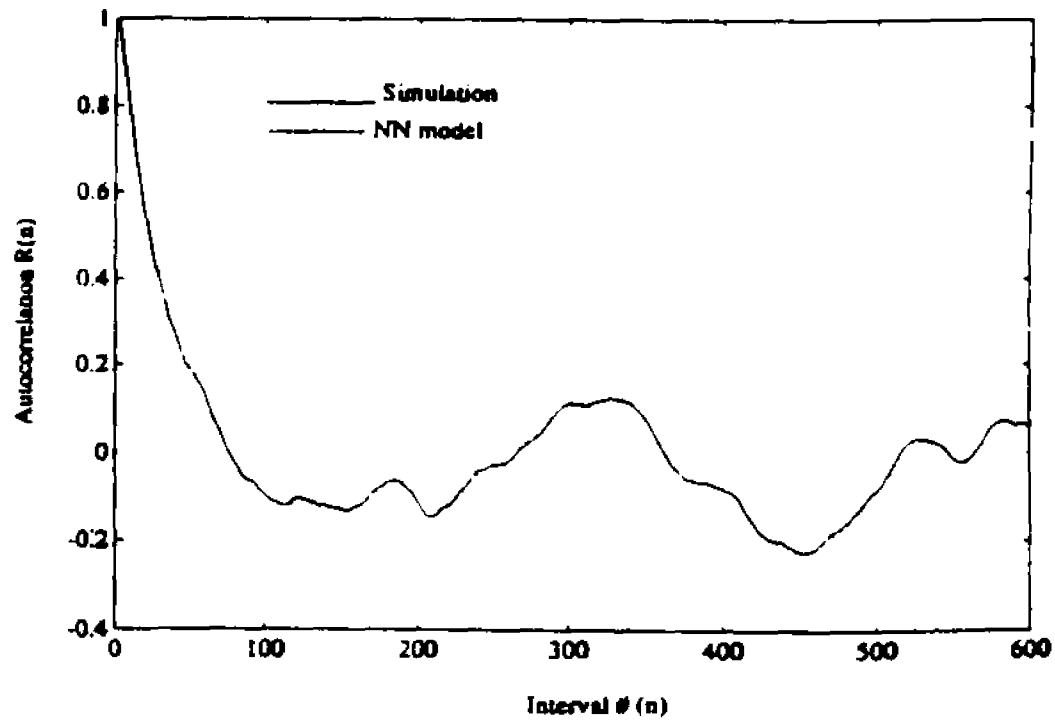
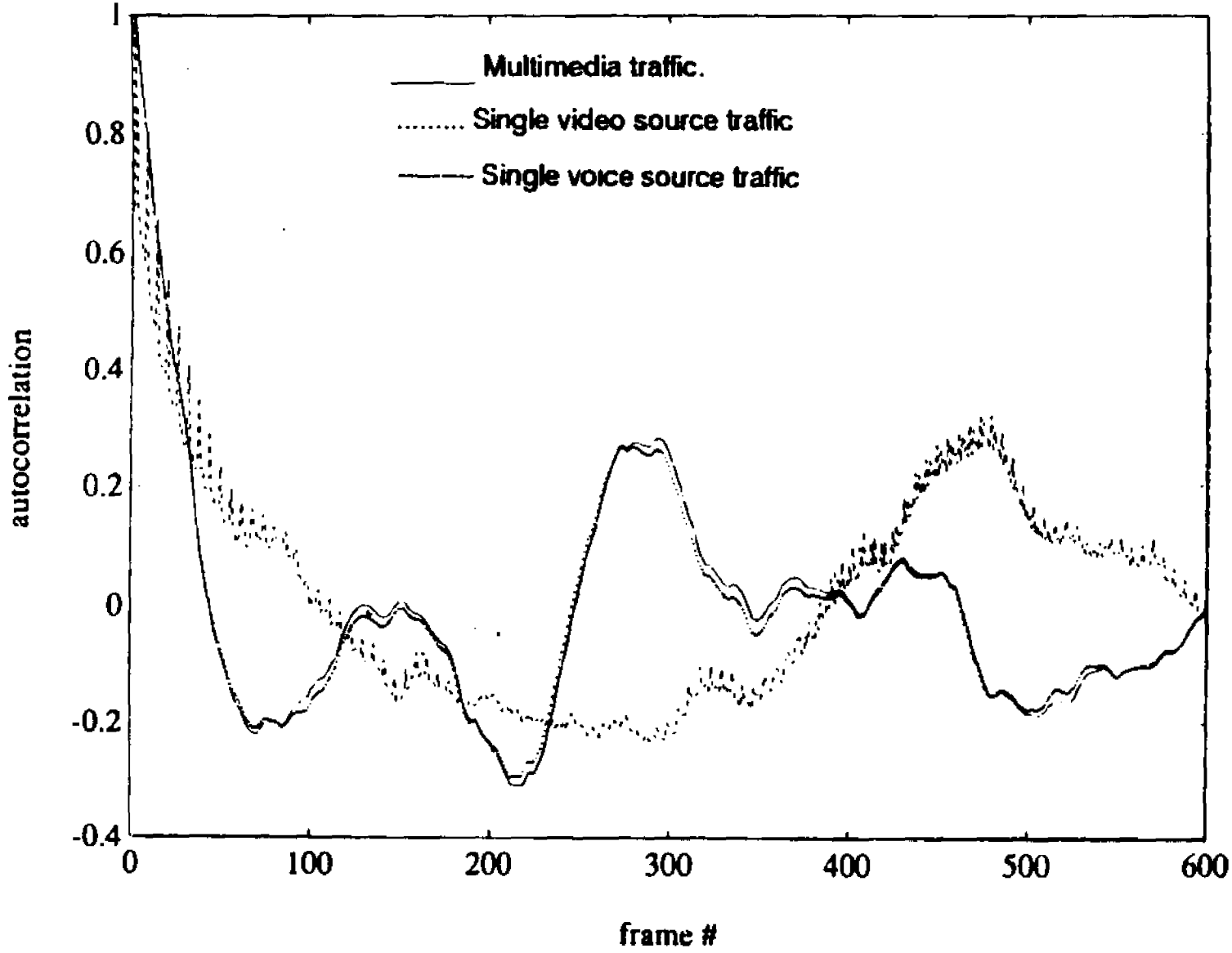


Fig. 2.28 Autocorrelation of the multimedia count process (N=1, M=1)

Fig. 2.29 Autocorrelation function



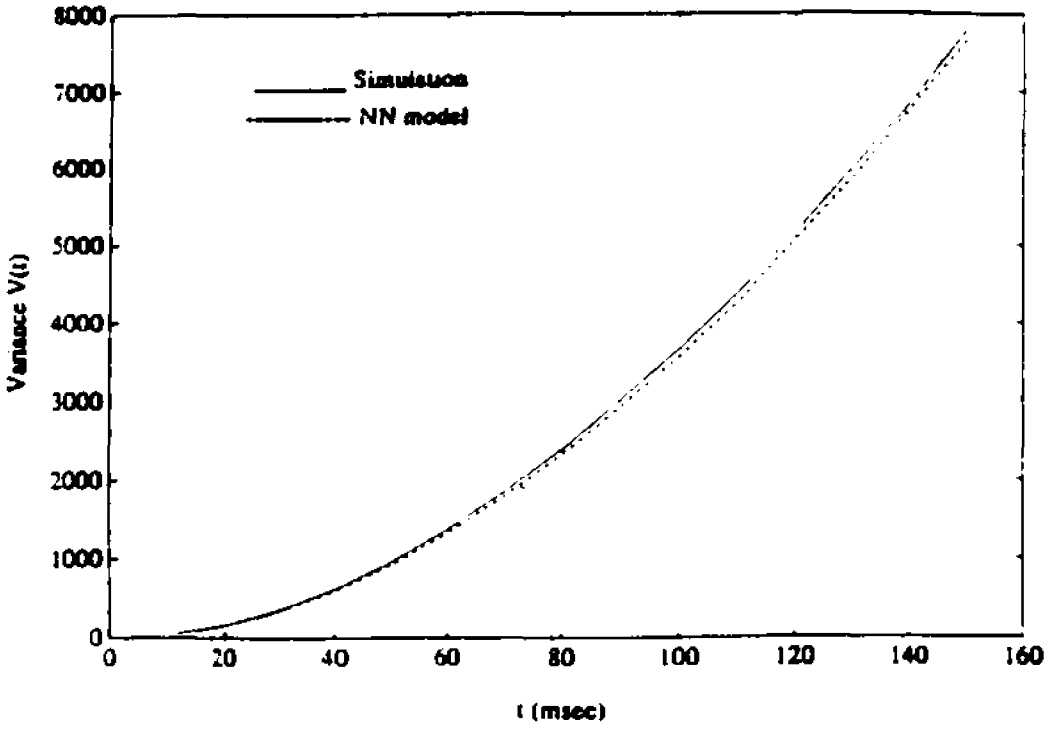


Fig. 2.30 Variance of the number of arrivals in (0,t) (N=1, M=1)

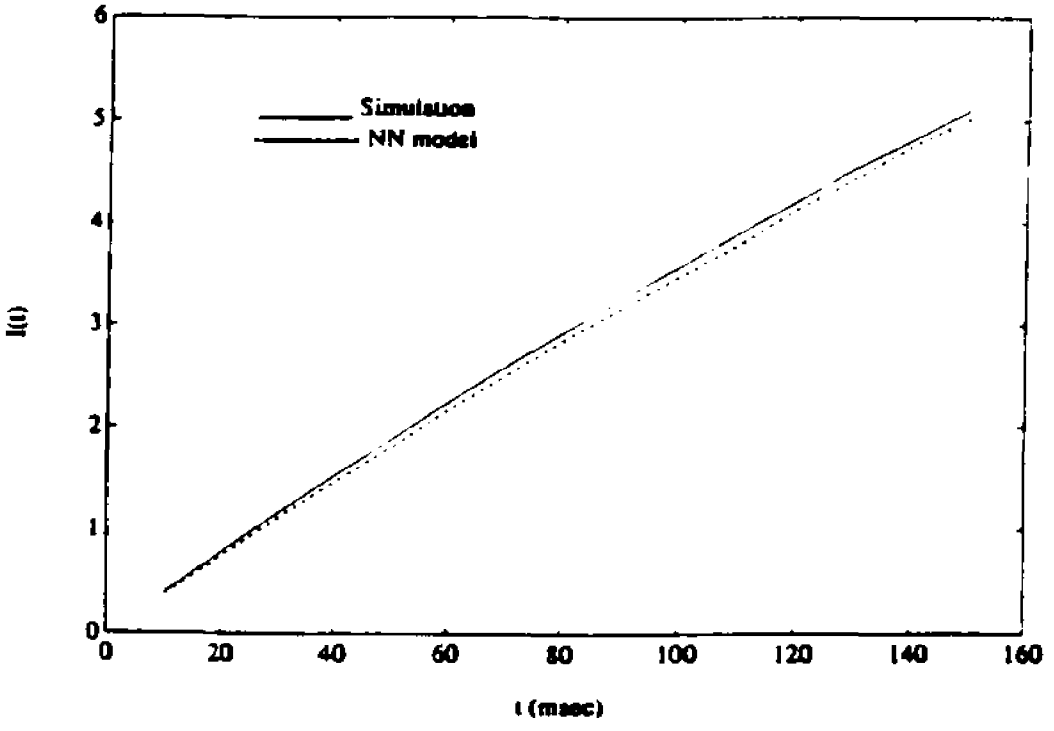


Fig. 2.31 Index of dispersion of the number or arrivals in (0,t) (N=1, M=1)

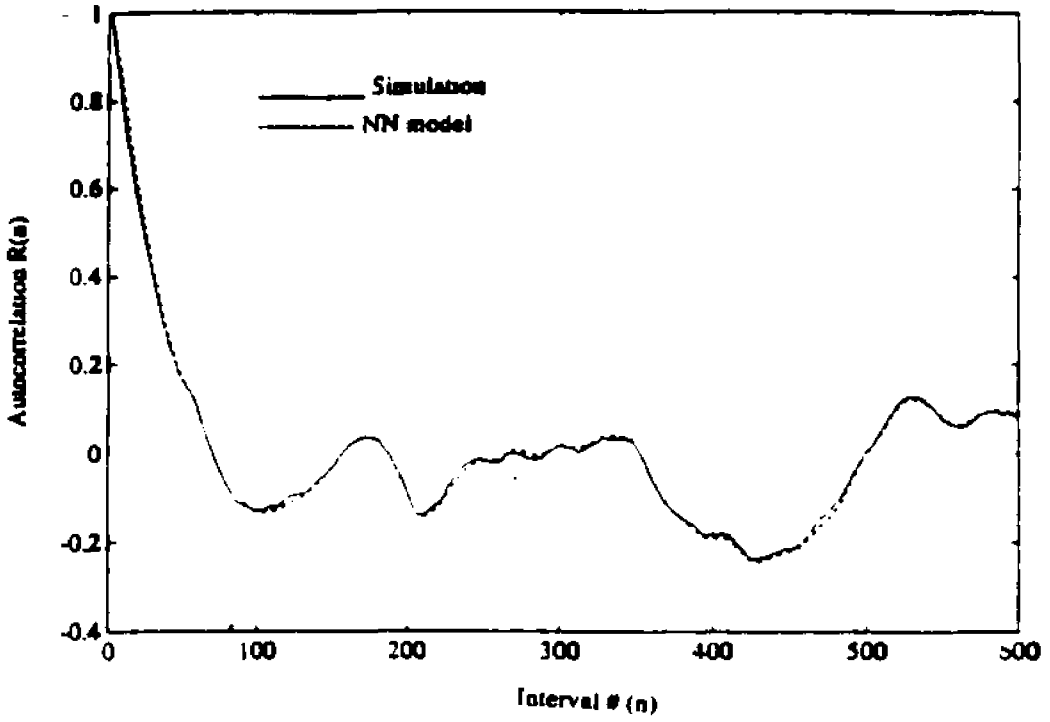


Fig. 2.32 Autocorrelation of the multimedia count process (N=1, M=1)

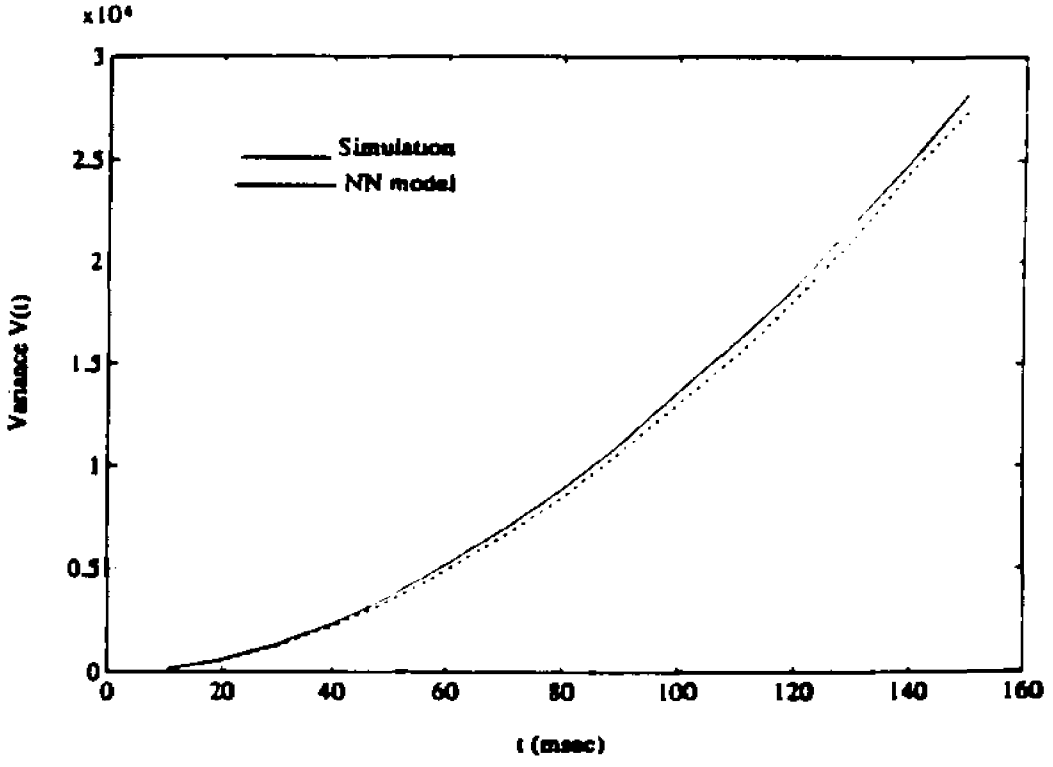


Fig. 2.33 Variance of the number of arrivals in (0,t) (N=5, M=5)

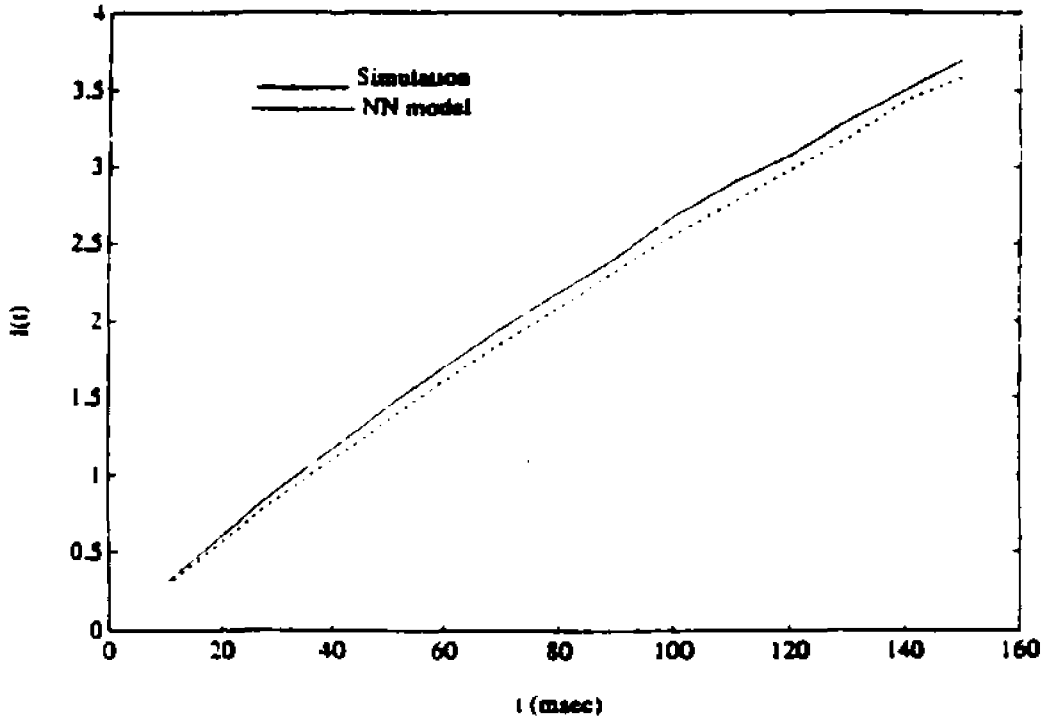


Fig. 2.34 Index of dispersion of the number or arrivals in (0,t) (N=5, M=5)

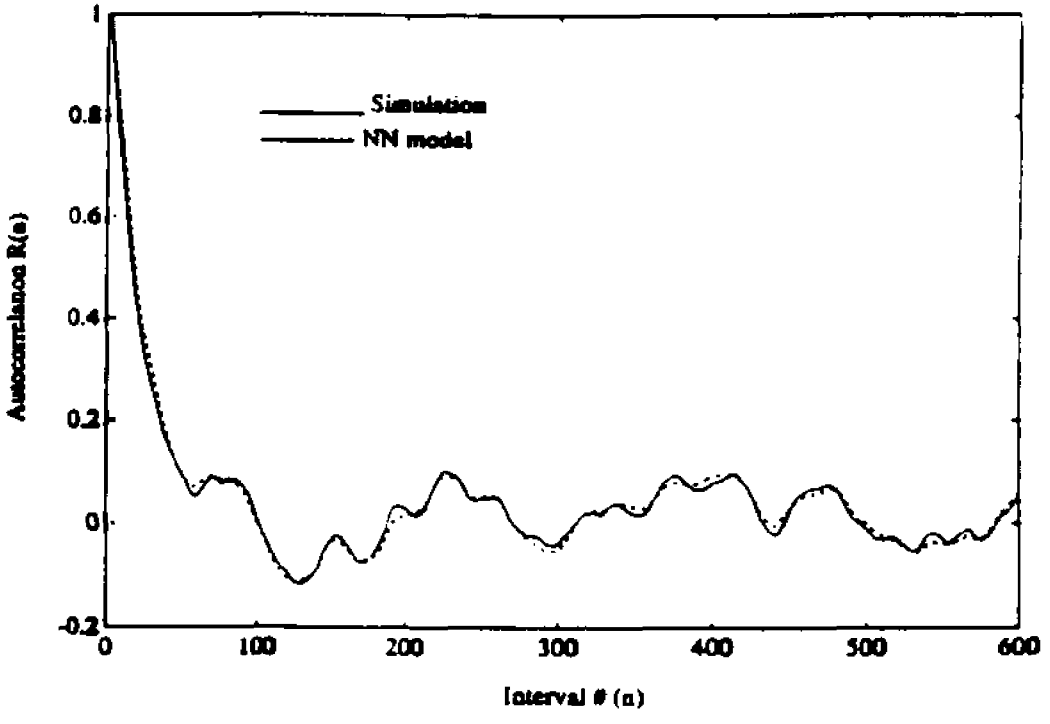


Fig. 2.35 Autocorrelation of the multimedia count process (N=5, M=5)

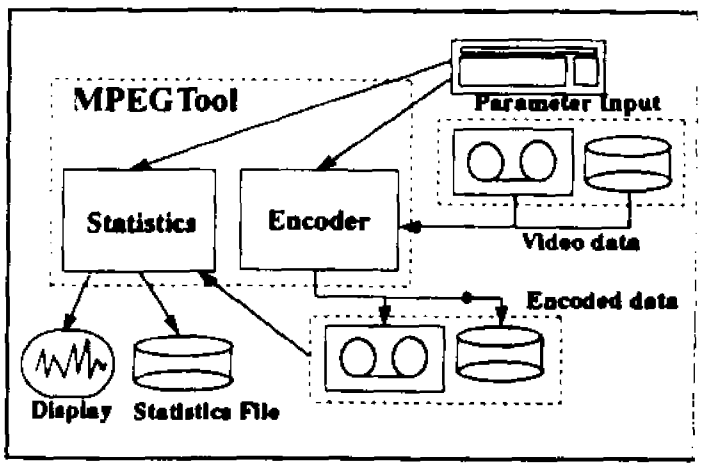


Fig. 2.36 Flow Control diagram of MPEGTool

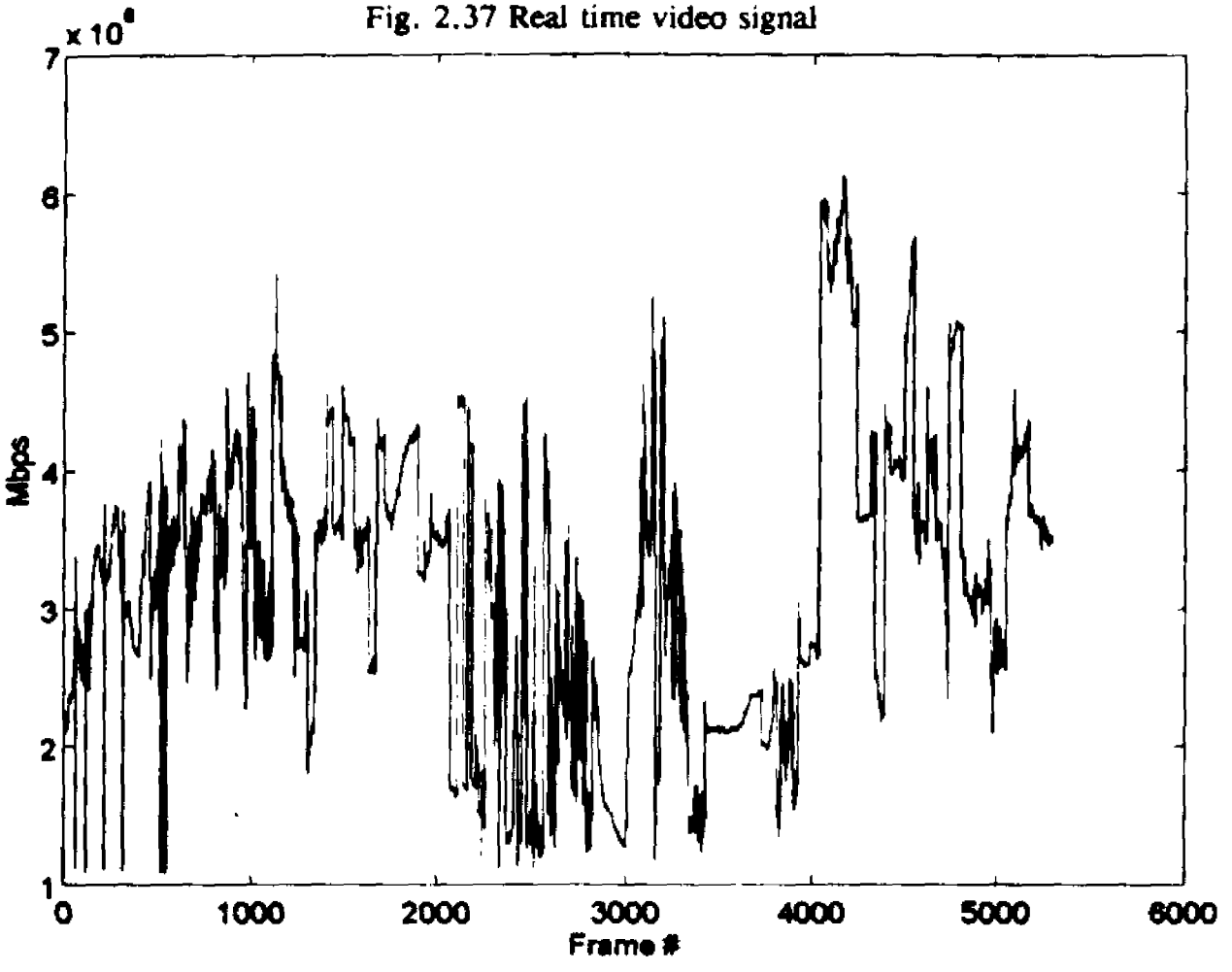


Fig. 2.37 Real time video signal

Fig. 2.38 NN prediction compared with the actual real time video signal

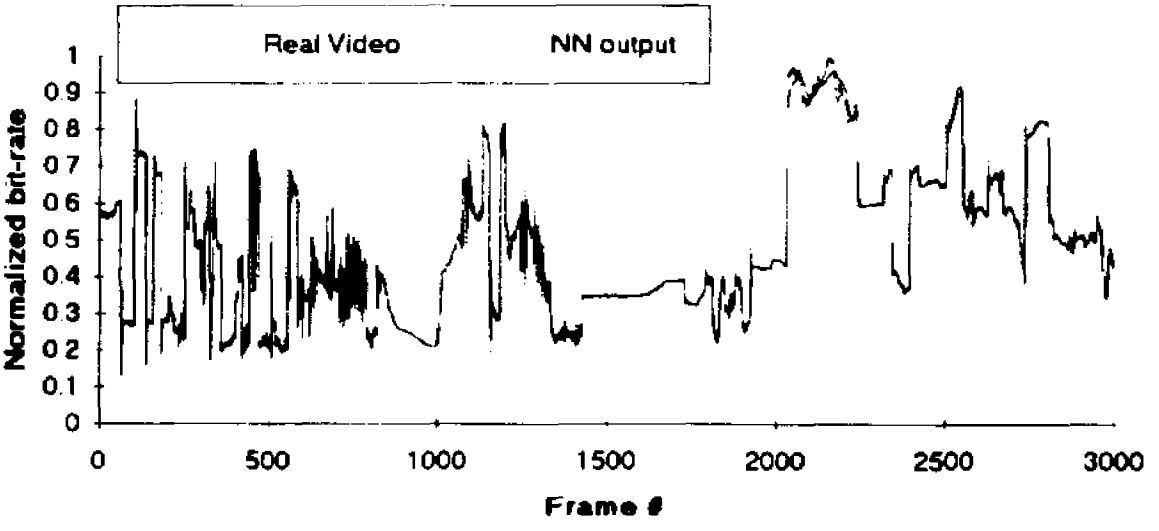


Fig. 2.39 Variance of the number of arrivals in (0,t)

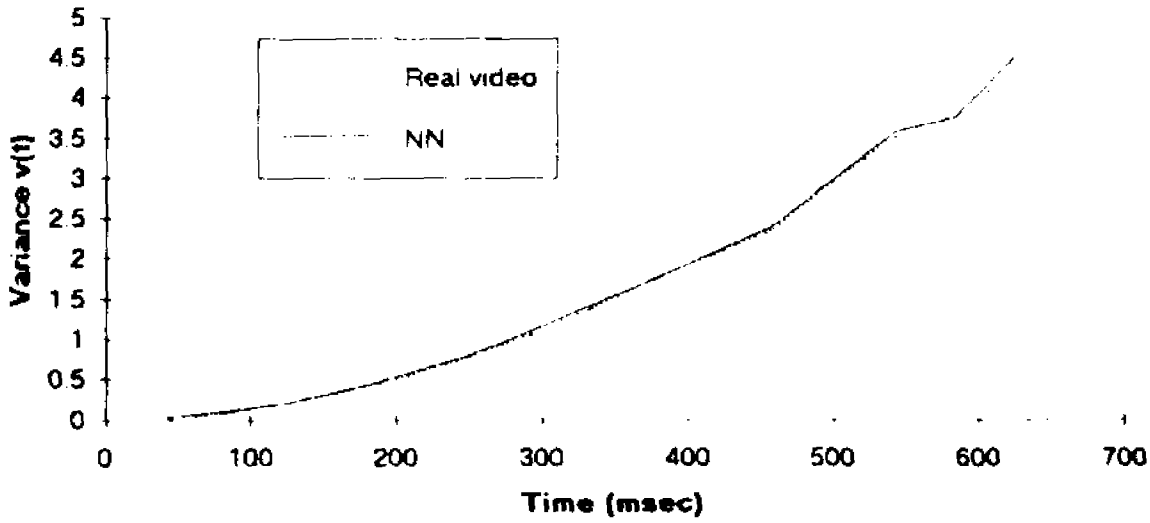


Fig. 2.40 Index of dispersion of the number of arrivals in (0,t)

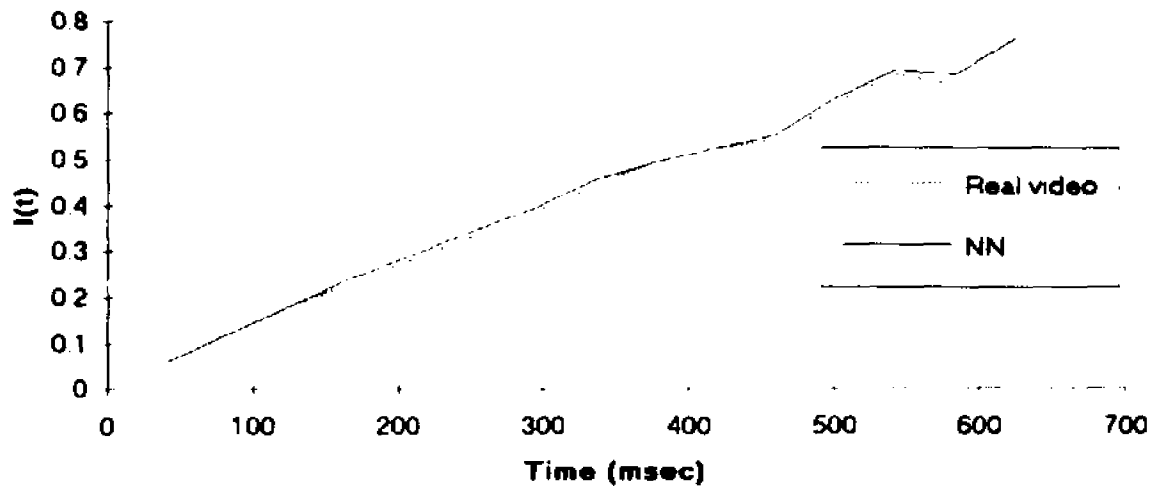
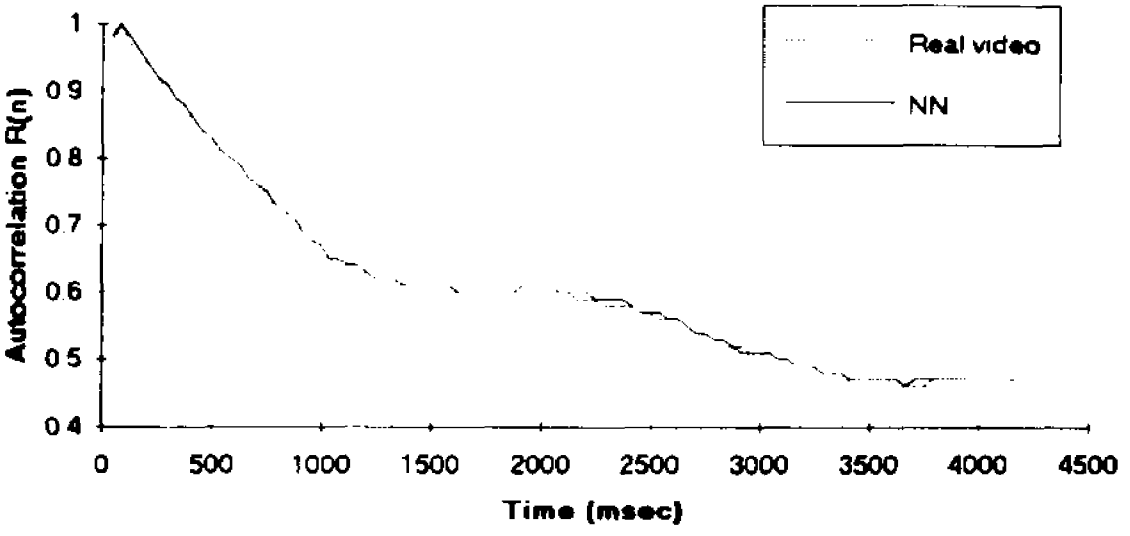


Fig. 2.41 Autocorrelation function



III. Neural Network Traffic Enforcement Mechanisms

This chapter presents how NN can be used as traffic enforcement mechanism in the ATM networks. We recall that in ATM networks, a set of user declared parameters that describes the traffic characteristics, used by the UPC algorithm, is required for the CAC and traffic enforcement (policing) mechanisms. At the call set-up phase, the CAC algorithm uses the UPC parameters to make a call acceptance decision. During the call progress, the policing mechanism uses the same parameters to control the user's traffic within its declared values in order to avoid possible congestion problems. In this chapter, a novel policing mechanism using neural networks (NNs) is presented. The mechanism is based upon an accurate estimation of the probability distribution function (pdf) of the traffic via its count process and implemented using NNs. The pdf-based policing is made possible only by NNs. This is due to the fact that pdf policing requires complex calculations, in real-time, at very high rates which is not feasible via conventional mathematical approaches. The architecture of the policing mechanism is composed from two inter-connected NNs. The first NN is trained to learn the pdf of an "ideal non-violating" traffic, whereas the second NN is trained to capture the "actual" characteristics of the "actual" offered traffic during the progress of the call. The output of both NNs (which is an accurate estimate of the traffic bit-rate fluctuations in the next measurement period) is compared. Consequently, an error signal is generated whenever the pdf offered traffic violates its "ideal" one. The error signal is, then, used to shape the traffic back to its original values. The reported results, prove that our policing mechanism is very effective in detecting (and controlling) all possible kinds of traffic

violations (e.g., peak and mean bit-rates violations).

III.1 Neural Network Traffic Enforcement Mechanism (NNTEM)

The system structure for the NNTEM is shown in figure (3.1) [35]. In the NNTEM, the traffic source is characterized via the pdf of its count process. The NNTEM method does not explicitly calculate any higher order moments in order to police the pdf of the traffic. It does that, however, by the NN transfer function which implicitly learns the pdf of the traffic count process through several millions of learning trials. Hence the NNTEM, does not rely upon the policing of simple parameters such as mean or peak bit-rates, it rather uses an elaborate and more accurate function which includes all the statistical properties of the traffic. Moreover, the NNTEM actually predicts (through the NN leaning algorithm) any violations in the count process bit-rate violations. If the source traffic violates its "agreed-upon" characteristics declared via its pdf, a violating signal (error signal) is generated. This signal can be used to regulate the traffic by different ways, the simplest one is to use this signal to drop the violated cells. We define two phases of operation: 1) Off-line training phase. 2) On-line operation phase.

In the Off-line training phase, NN1 is trained, to capture the actual pdf of the ideal traffic non-violating (a traffic that does not violate its pdf during the call progress). For example, consider a multimedia source that generates three types of traffic (data, voice, and video) NN1 is trained to learn the characteristics of each type and any possible combinations of those types (e.g., video and voice, data and voice, etc.). Hence, different classes of traffic characteristics are defined and stored in NN1. In the call set-up

phase, the user will declare the connection's class of traffic and accordingly, NN1 will select the corresponding input-output mapping function that contains specific information about the pdf of that class of traffic.

The NN learns the pdf of the traffic as it learns the relationship between current and past values of the number of cell arrivals within a certain measurement period (count process). NN2 is also a backpropagation NN which is trained to predict the future values of the pdf of the actual offered traffic (which may exhibit violations in its pdf) during the call progress. In the on-line phase (the call progress phase), the current and past values of the count process are fed to both NN1 and NN2. NN1 predicts the future values of the count process of the traffic based upon its inputs and upon the actual pdf of the non-violating traffic, whereas NN2 predicts the same for the actual offered traffic (i.e., it actually predict the violations based upon past history). If the source traffic does not violate its "agreed-upon" characteristics, both NN1 and NN2 produce almost the same outputs in the next time interval. Hence, the error signal between NN1 output and NN2 output is almost zero, and no policing action is taken. While, if the NNs detect any violations an error signal, is produced, and a corresponding policing will be taken. The error signal is function only, of any violations that may occur in the shape of the traffic bit-rate variations depicted by the count process. However, in order to achieve an acceptable level of robustness, in the policing action, one might specify a certain threshold of the error signal. Any error signal that is less than the threshold value will be discarded and no policing action will be taken.

Traffic Source model:

In this section, we used an ON/OFF voice source model (explained in section II.2.1) to demonstrate the efficacious of the policing mechanism. This model has been

used in earlier publications to model packetized voice [12],[13],[18], still picture [20], and interactive data services. It allows the relevant parameters namely peak bit-rate [b], mean burst (ON) duration [$1/\alpha$], mean OFF (silence) period [$1/\beta$], mean bit-rate[m], to be varied independently of each other. This model was used to generate the count process (X, T_s), where (X) is the number of cells arriving in a time interval (T_s).

NNTEM:

As mentioned above, during the off-line training phase, NN1 learns the bit-rate variations of the traffic (in terms of the count process (X, T_s)). It predicts future value(s) of (X, T_s) (represented by[$\hat{H}(i+m)$]) based upon the current and past values of (X, T_s) (represented by [$H(i)$]). By doing this prediction process, NN1, actually learns pdf of (X, T_s). In other word, after completion of the training phase of NN1, and upon the introduction of the traffic pattern of (X, T_s) to its inputs, it maps these inputs to its outputs via the already captured pdf of the non-violating traffic. Once NN1 learns the pdf of the ideal "non-violating" traffic, it does not learn any new traffic-patterns. During the on-line operation phase, it acts as reference bench-mark model for the ideal "non-violating" traffic. Whereas in the on-line operation phase (call progress) NN2 continues to learn new traffic-patterns which are violations of the original ideal traffic.

It is clear from the previous discussion, that the traffic generated from a source can be categorized into one of two possible categories. Either a "non-violating" (ideal) traffic category, or a "violated" traffic category. If NN1 is exposed to a traffic that belongs to the ideal category, it produces outputs that are extremely close to the actual future values. NN2 is trained "off-line" as well as "on-line". The main purpose of NN2 is to predict the future value(s) of (X, T_s) for the two traffic categories mentioned above. During the off-line learning phase, NN2 is trained to both categories, and learns

how to predict the future values of both "violating" and "non-violating" traffic. It is important in the training phase, that NN2 encounters all possible violations that might occur (e.g., peak, mean.... etc.). Although, the structure and the leaning rates of NN2 could be different from those of NN1, but NN1 and NN2 still have the same size of input vectors and output vectors .

In the on-line operation phase, both NN1, and NN2 are exposed to the same current and past values of (X, T_s) through the input traffic vector [H(i)]. A violating signal (error signal) is generated by subtracting the output vector [$\hat{H}$(i+m)] of NN1 from that of NN2. This signal tells how far the current traffic, generated from the source, violates its "agreed-upon" contractual values. The violating signal will be zero if the traffic does not violate its "agreed-upon" pdf, and will be non-zero if it does. This signal is then, used to drop the violated cells, or more efficiently, it can be fed to another third NN that interprets this signal, together with some pre-defined optimization function. Based upon that function, the error signal could be ignored, amplified or multiplied by some factor. For example, one might wish to ignore the short term violations in the peak bit-rate as long as the mean bit-rate is not violated. Such added functionality can be simply incorporated by adjusting the optimization function such that less weight is given to the error signal generated by such types of violations.

III.1.1 Simulation Results

The cells' arrival process, resulting from the packetized voice source, was simulated assuming a voice source peak bit-rate of 32 kbps and an ATM cell size of 53 bytes. The nominal values of the relevant parameters for that source are: b_0=32 kbps, $1/\alpha$=350

msec, $1/\beta = 650$ msec, and $m_0 = 11.2$ kbps. A traffic that has these parameters and does not violate them during the call progress is an ideal "non-violated" traffic. NN1 is trained using the above parameters in the off-line phase of operation, to learn the pdf of the count process. It is trained to predict the count process for the next sampling period ($T_s = 10$ msec) based upon 20 past values of the count process. Hence it contains 10 PEs with hidden layer, and one PE in the output layer. The training data set of NN1(20,10,1) consisted of millions of sets of [H(i)] and [H(i+m)] vectors observed by running the simulation model for several hours. Similarly, in this phase of operation, NN2 is trained to predict the count process for different types of traffic (violating and non-violating). The training data set of NN2 contained all possible traffic patterns that might appear due to any violations in its pdf. For simplicity, we considered the following cases of traffic violations:

case(1): The traffic violates both the peak bit-rate and the mean bit-rate.

case(2): The traffic violates the peak bit-rate only.

case(3): The traffic violates the mean bit-rate only.

The "on-line" operation phase starts as soon as the call gets accepted by the CAC algorithm at the call set-up phase. During this phase, the learning rate parameter (a parameter determines the speed of learning of the NN) of NN1 is set to zero, since it is not required from NN1 to learn any characteristics about the offered traffic. Since the function of NN2 is to predict correctly the future values of the offered traffic, its learning rate is adjusted such that NN2 continues to learn new traffic-patterns.

During this phase of operation , both NN1 and NN2 are exposed to both the past and the current values of the traffic during the call progress, through the vector [H(i)]. An error signal is generated, by comparing the output of NN1 and that of NN2, which

is used to control the status of the cell dropping switch (see figure (3.1)). If the source traffic is a non-violating one, both NN1 and NN2 give almost the same value for the count process (X, T_s) over the next sampling period. In this case the error signal is zero, and no cells will be dropped. Whereas if the traffic violates its contractual UPC parameters, NN1 gives a prediction of the count process (X, T_s) based upon the pdf of the ideal traffic stored in it, whereas NN2 predicts the violating traffic, over the next sampling period. In that case the error signal will not be zero, and the violating cells will be dropped.

III.1.2 Numerical Results

In this section, we demonstrate the effectiveness of the proposed NNTEM by demonstrating its capability to produce an error signal whenever a violation of the pdf occurs. Four experiments were performed to explore how the error signal is generated for the different violations cases mentioned in the previous section. Figure (3.2) shows the pdf of the ideal traffic as well as for each of the different violations cases.

Experiment (3.1):

In this experiment, the offered traffic during the call progress does not violate its contractual UPC parameters, the pdf of that traffic is shown in figure (3.2a). Figure (3.3) shows the NN1 output compared with the actual one. It is clear that NN1 prediction is very close to the actual traffic values, and NN1 capture the statistical properties of the ideal traffic as can be assisted by figure (3.4) which shows the autocorrelation function of the traffic produced by NN1 compared with that of the actual traffic. Figure (3.5) shows the error signal for that case. In this case, the signal is zero because the offered

traffic to NN1 has the same pdf, as that captured by NN1. No policing action is taken in this case.

Experiment (3.2):

In this experiment, the offered traffic violates its contractual UPC parameters through an increase in both its peak and mean bit-rate. The resulting pdf of this traffic is shown in figure (3.2b) . In this case the traffic relevant parameters were $b=1.5b_0$, and $m=1.5m_0$. Figure (3.6a) shows the generate error signal which indicates how far the offered traffic has deviated from its ideal case. All violating cells are then dropped. Figures (3.6b) show the predicted count process produced by NN2, predicted output process produced by NN1, the error signal, and the policed count process over the first 5 sec. Figures (3.6c) show the histogram of the offered traffic (before the dropping switch), the histogram of the policed traffic (after the dropping switch) and the histogram of the "ideal" traffic. It clear that the histogram of the policed traffic is close to that of the ideal traffic. For example, for the ideal traffic, no more than 5 cells are arrived during one sampling period T_s (as shown by the histogram of the ideal traffic). For the policed traffic, also, during one sampling period T_s, no more than 5 cells are allowed to pass the switch and enter to the network.

Experiment (3.3):

In this experiment, the traffic violates, only, the peak-bit rate such that $b=1.5b_0$. The pdf of this traffic is shown in figure (3.2c). Figure (3.7) shows the error signal for that case. From figures (3.6) and (3.7), we can see that the error signal generated from peak and mean violations and that generated from a peak violation only, have similar time varying characteristics and are very close in magnitude. This implies that violations of the peak bit-rate dominate those of the mean bit-rate. It, also, explains the reason that almost all

existing policing mechanisms have tremendous difficulties enforcing both peak and mean bit-rates violations. As a matter of fact, these results prove that our NNTEM is capable of policing both short-term and long-term violations.

Experiment (3.4):

In this experiment, the offered traffic violates, only, its mean bit-rate such that $m = 1.5m_0$. The pdf of this traffic is shown in figure (3.2d). Figure (3.8) shows the error signal in that case. By reference to figures (3.6), (3.7) and (3.8), it is clear that the error signal in that case is less in magnitude than those of others types of violations and has different time varying characteristics. This result confirms that peak and mean bit-rates violations are not detectable by the same measuring algorithm. Since a measuring algorithm that can detect the peak bit-rate violations must have very short reaction time, yet the same algorithm must have a longer reaction time to detect violations in the mean bit-rate. This problem has been solved by our pdf-based NNTEM algorithm.

It is clear from the above results that the NNTEM can easily detect both mean and the peak bit-rate violations, with a single NN architecture. The problem of policing both the mean and the peak bit-rates using a single set of parameters has been a challenging problem for all existing flow enforcement mechanisms. Our proposed NNTEM solves this problem via a pdf-based algorithm utilizing the learning capabilities of NNs.

III.2 A Neural Network Controller Using Reinforcement Learning Method for ATM Traffic Policing

In this section, a NN controller using reinforcement learning method for traffic policing is presented [36]. The architecture of the NN-based controller is composed of critic function and control function. The critic function uses the NNTEM explained in the previous section. The error signal produced by the NNTEM used by the critic function to produce an evaluation signal based upon a certain performance measure. A reinforcement learning method uses the evaluation signal to adjust the weights of a third NN, in the control function that generates a control signal capable of maximizing the system performance (i.e., minimization of the traffic violations).

III.2.1 Reinforcement learning_type policing function

In supervised leaning systems [48], the correct "target" output values of the NN are known for each input pattern. But in some situations there is less detailed information available. In the extreme case there is only a single bit of information, saying whether the output is right or wrong. Reinforcement learning procedure is applicable to this extreme case. Reinforcement learning is a form of supervised learning system [49] because the NN does, after all, get some feedback from its environment. The performance measure for a supervised learning system is defined in terms of a set of targets outputs by means of a known error criterion (e.g., mean square error). Whereas reinforcement learning addresses the problem of improving performance as evaluated by

any performance measure whose values can be supplied to the learning system. Consequently, in tasks of this kind there exist desired network output signals, fed to the environment (i.e., the system to be controlled), namely those that lead to optimal environment performance-but the learning system is not told what they are because there is no agency knowledgeable enough to act as this kind of teacher. The problem, then, is to find those optimal control signals. In the case of reinforcement learning, however, the situation is more general than that for the supervised learning system in that instead of trying to determine target control signals from target environment response, one tries to determine target control signals, or desired changes in the control signals, that would lead to increases in a measure of the environment performance. This is illustrated in figure (3.9) which shows a reinforcement-learning NN-based controller interacting with the environment and receiving an "evaluation signal" generated by a "critic" capable of evaluating plant performance. Viewed this way, reinforcement learning essentially involves two problems. The first one is to construct a critic capable of evaluating plant performance. The second, is to determine how to alter controller outputs to improve performance as measured by the critic.

Since the main objective of this section is to design a controller which optimally minimize the violations of the source's traffic from its "agreed_upon" characteristics, and since there is no target control actions corresponding to a specific type of traffic violation, we have introduced a reinforcement_based learning method which tunes the weights of a NN-based controller in order to achieve an efficient policing function capable of detecting and policing the different cases of the source's traffic violations, expressed in terms of peak and mean bit-rates violations, both long term and short term ones. Our selection of reinforcement learning method is based on the following:

1- The knowledge (in terms of the target control signal required to control a certain type of traffic violation) required to apply any other learning method, is not available.

2- Model-based adaptive control systems can not be used in our case, because the range of assumptions in the model (of the system to be controlled) may not be justified to the degree necessary to obtain the desired refinements in the performance.

3- Adjusting the control rule by direct appeal to the performance measure (as in our case), can shorten the chain of assumptions on which a method is dependent.

Reasons such as these, which all involve aspects of the usual tradeoffs between the acquisition of the knowledge and its use for control, suggest that direct adjustment of the control law via reinforcement learning can play a useful role in control and police the source's traffic violations.

The proposed system consists of a critic part and NN-based controller part. The inputs to the system are the sampled values of the traffic count process and the system output is the control signal used to drop the violated cells. Figure (3.10) shows the NN scheme used as a controller while figure (3.11) shows the block diagram of the reinforcement learning type NN-based controller applied to police the source's traffic violations.

The critic part of the system consists of our previous mechanism NNFEM and the performance measure as follows:

NN1 is a backpropagation NN which is trained, off-line, to capture the actual pdf of the ideal non-violating traffic. NN2 is also a backpropagation NN which is trained to predict the future values of the actual offered traffic (which may exhibit violations in its pdf) during the call progress phase. In the on-line phase (the call progress phase), the current and past values of the count process are fed to both NN1 and NN2. NN1 predicts the

future values of the count process of the traffic based upon its inputs and upon the actual pdf of the non-violating traffic, whereas NN2 predicts the same for the actual offered traffic (i.e., it actually predict the violations based upon past history). If the source traffic does not violate its "agreed-upon" characteristics, both NN1 and NN2 produce almost the same outputs in the next time interval. Hence, the error signal (ϵ) between NN1 output and NN2 output is almost zero. While, if the NNs detect any violations an error signal (ϵ) is produced. The error signal (ϵ) is function in the violations that may occur in the shape of the traffic bit-rate variations depicted by the count process. In NNTEM, this error signal (ϵ) is used to drop the violated cells. In this section, this signal is used by the performance measure (cost function) (J) to adjust the weights of the NN-based controller in order to improve the system performance. The cost function is defined by:

$$J(P) = \sum_{k=1}^{L} Q(h_d(k) - h(k))^2 = \sum_{k=1}^{L} Q\epsilon^2(k) \qquad (3.1)$$

where P is the trial number, L is the sampling number within one trail, h(k) is the switch output (policed traffic, expressed as count process at sample number k), $h_d(k)$ is the desired value of the traffic (expressed as count process), and ϵ is the error signal between NN1 output and NN2 output (output error). Minimization of this cost function implies minimization of the source's traffic h(k) from its desired value $h_d(k)$ over the interval [1,L]. The constant Q is a weighting factor used to minimize the traffic violations.

The NN of the controller has three layers with no inner feedback loop and no direct connection from the input layer to the output layer as shown in figure (3.10). Both the hidden and the output layers have a sigmoid function [48] to provide the nonlinear mapping capability. The NN output is the environment input, u(k) (the control signal,

$0 \leq u(k) \leq 1$). This control signal is used to control the operation of the dropping switch or a more complicated device that control the flow of the cells to the network. For example, this device could introduce some delay (ΔT) between the transmission of the consecutive cells. The value of (ΔT) could be varied according to the control signal u(k). The NN controller is expressed by the following equation [50]:

$$u(k) = f\{w^T(P)\, f[W(P)\, I(k)]\} \qquad (3.2)$$

where **w** is the weight vector from the hidden layer to the output layer, **W** is the weight matrix from the input layer to the hidden layer. Function f is the sigmoid function defined by the following equation [48]:

$$f(x) = \frac{1}{1+\exp(-xS)} \qquad (3.3)$$

where x is the input to the sigmoid function and S is the parameter determining its shape. I is the input vector to the controller, expressed as follows:

$$I^T(k) = [h(k), h(k-1), h(k-2), h(k-3)] \qquad (3.4)$$

Both the weight vector, **w**, and the weight matrix, **W**, are tuned so as to minimize the cost function, J, which is defined by (1.3). There are many methods to tune the weights of the NN such as steepest descent method, Newton's method, quasi Newton method, and conjugate gradient method [49]. Because the steepest descent method has the advantages of being simple and highly parallel, we used it to tune the weights of the NN as following:

$$w(P+1) = w(p) - \eta \frac{\partial J(P)}{\partial w(P)} \qquad (3.5)$$

$$W(P+1) = W(P) - \eta \frac{\partial J(P)}{\partial W(P)} \qquad (3.6)$$

where η is the parameter determining the convergence speed of the weights tuning. The work reported in [48] and [51] has proved the conversion of the cost function using the weight tuning algorithm given in equations (3.5) and (3.6).

Our controller is designed to regulate the source's traffic around its ideal values, where it measures the output traffic from the dropping switch. The following equation describe the output of the dropping switch:

$$h(k) = u(k) * h_i(k) \qquad (3.7)$$

where k is the sampling number, $h_i(k)$ is the source's traffic expressed as the count process at sample number k, h(k) is the switch output (expressed as count process at sample number k), and .

In order to realize the reinforcement mechanism shown in (3.5) and (3.6), it is necessary to derive the details of $(\partial J(P)/\partial w(P))$ and $(\partial J(P)/\partial W(P))$ which is given in appendix I.

III.2.2 Simulation Results

In this section, we demonstrate the effectiveness of the proposed controller by demonstrating its capability to produce a control signal whenever a violation of the pdf occurs. Four experiments were performed to the show pdf of a traffic and to explore

how the control signal is generated for the different violations cases mentioned in section III.1.1.

Experiment (3.5):

In this experiment, the offered traffic during the call progress does not violate its contractual parameters. Figure (3.12) shows the pdf of that traffic, compared with the pdf of the output traffic from the cell dropping switch and the pdf of the "ideal" traffic. It is clear that the pdf of the output traffic (policed traffic) and the pdf of the "ideal" traffic are the same because the offered traffic has the same pdf captured by NN1. Hence, in this case the error signal (ϵ) is zero and the control signal U is unity, and no cells are dropped.

Experiment (3.6):

In this experiment, the offered traffic violates its contractual parameters through an increase in both its peak and mean bit-rate. The resulting pdf of this traffic is shown in figure (3.13) . In this case the traffic relevant parameters were $b = 1.5b_0$, and $m = 1.5m_0$. Figure (3.13) shows the pdf of the output traffic (from the switch) compared with that of the ideal traffic. Figure (3.13) illustrate that the pdf of policed traffic matches that of the ideal traffic with extreme accuracy. Figure (3.14) shows the control signal generated by the controller to police that traffic violation.

Experiment (3.7):

In this experiment, the traffic violates, only, the peak-bit rate such that $b = 1.5b_0$. The pdf of this traffic is shown in figure (3.15). Figure (3.15) compare the pdf of the policed traffic to that of the ideal traffic. It is clear from that figure how well is the proposed controller in detecting and policing that type of traffic violation. Figure (3.16) shows the control signal generated in this case.

Experiment (3.8):

In this experiment, the offered traffic violates, only, its mean bit-rate such that $m = 1.5m_0$. The pdf of this traffic is shown in figure (3.17). Figure (3.17) shows the pdf of the policed traffic compared with that of the ideal traffic. This figure proves the efficiency of the proposed controller to detect and police the mean bit-rate violation of the traffic. It is interesting to mention here that the same controller used to police the peak bit-rate violation is used to police the mean bit-rate violation. This justifies the fact that the proposed controller is capable of policing both the mean and the peak bit-rates violations with extreme accuracy.

It is clear from the above results that the proposed reinforcement learning type policing function can easily detect both mean and the peak bit-rate violations, with the same NN architecture. The problem of policing both the mean and the peak bit-rates using a single set of parameters has been a challenging problem for all existing traffic enforcement mechanisms. Our proposed controller solves this problem via a pdf-based algorithm utilizing the learning capabilities of NNs, and by direct adjustment the weights of the NN in order to minimize the cost function using the reinforcement learning method.

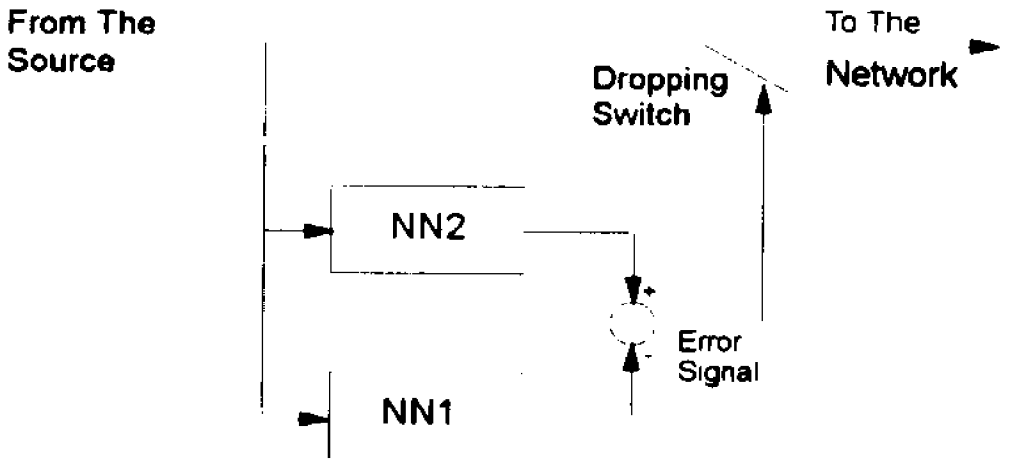


Fig. 3.1 NNTEM Structure

Fig. 3.2 Histogram of the count process

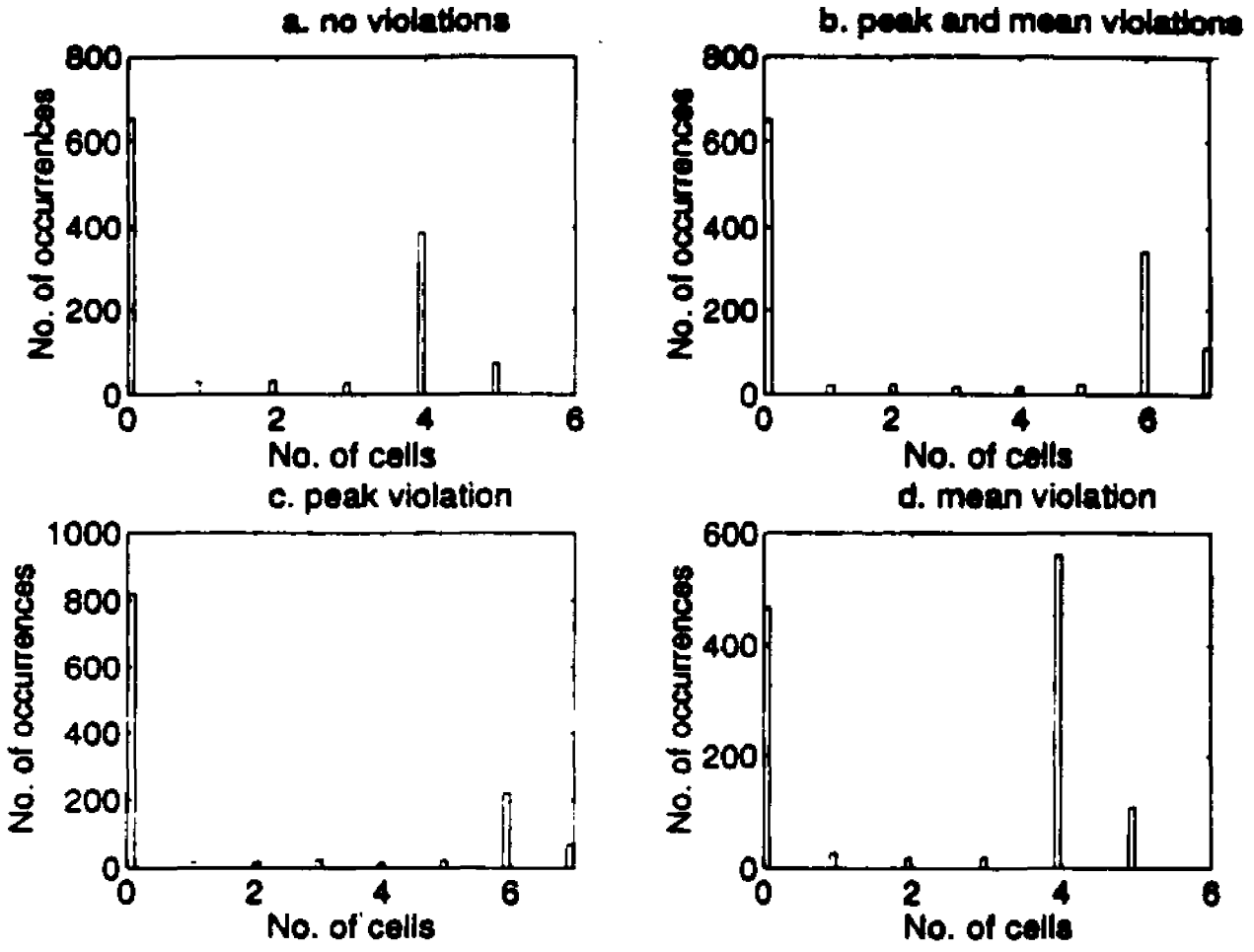


Fig. 3.3 Comparison of the output process from NN1 with the simulation one

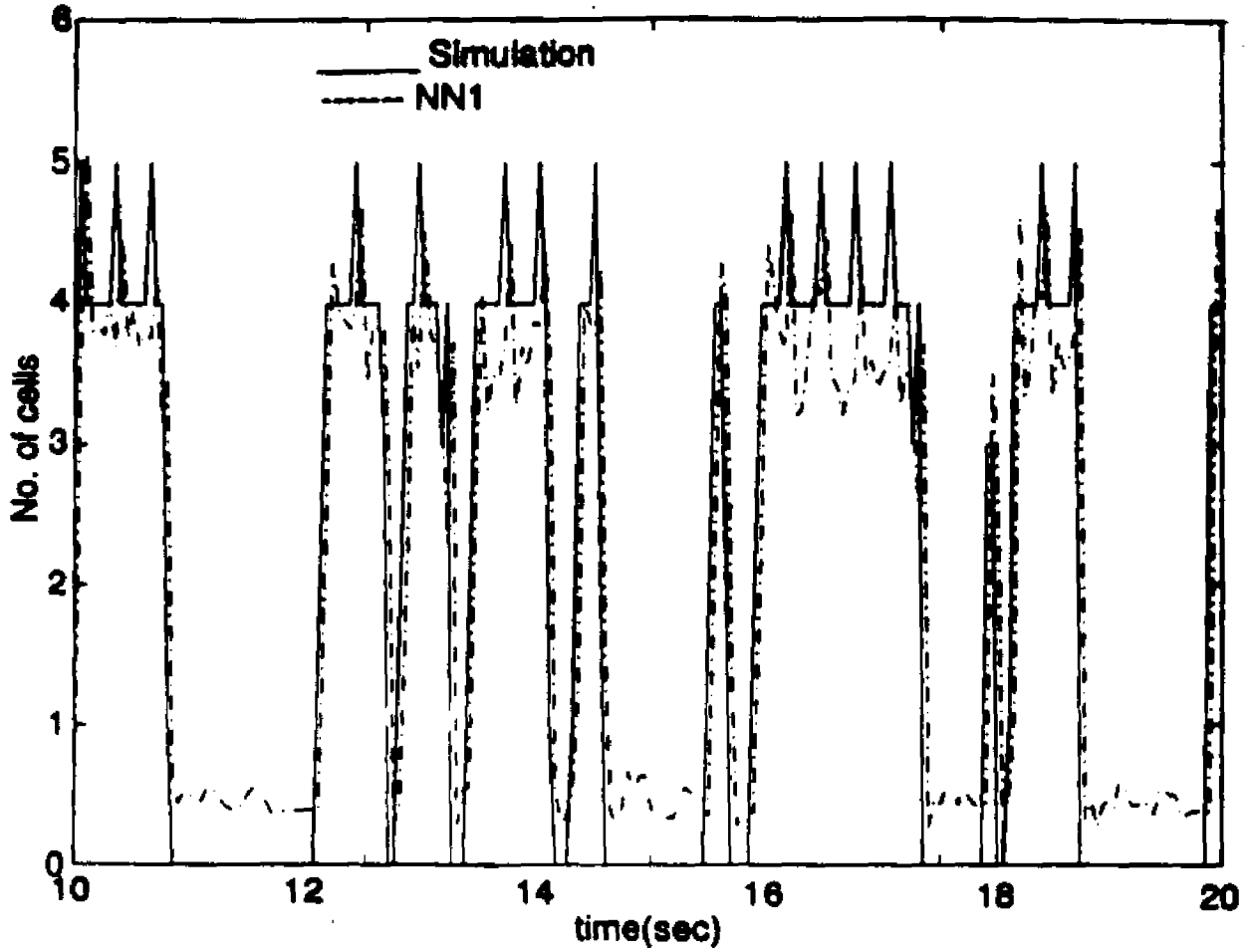


Fig. 3.4 Comparison of the autocorrelation function of the count process

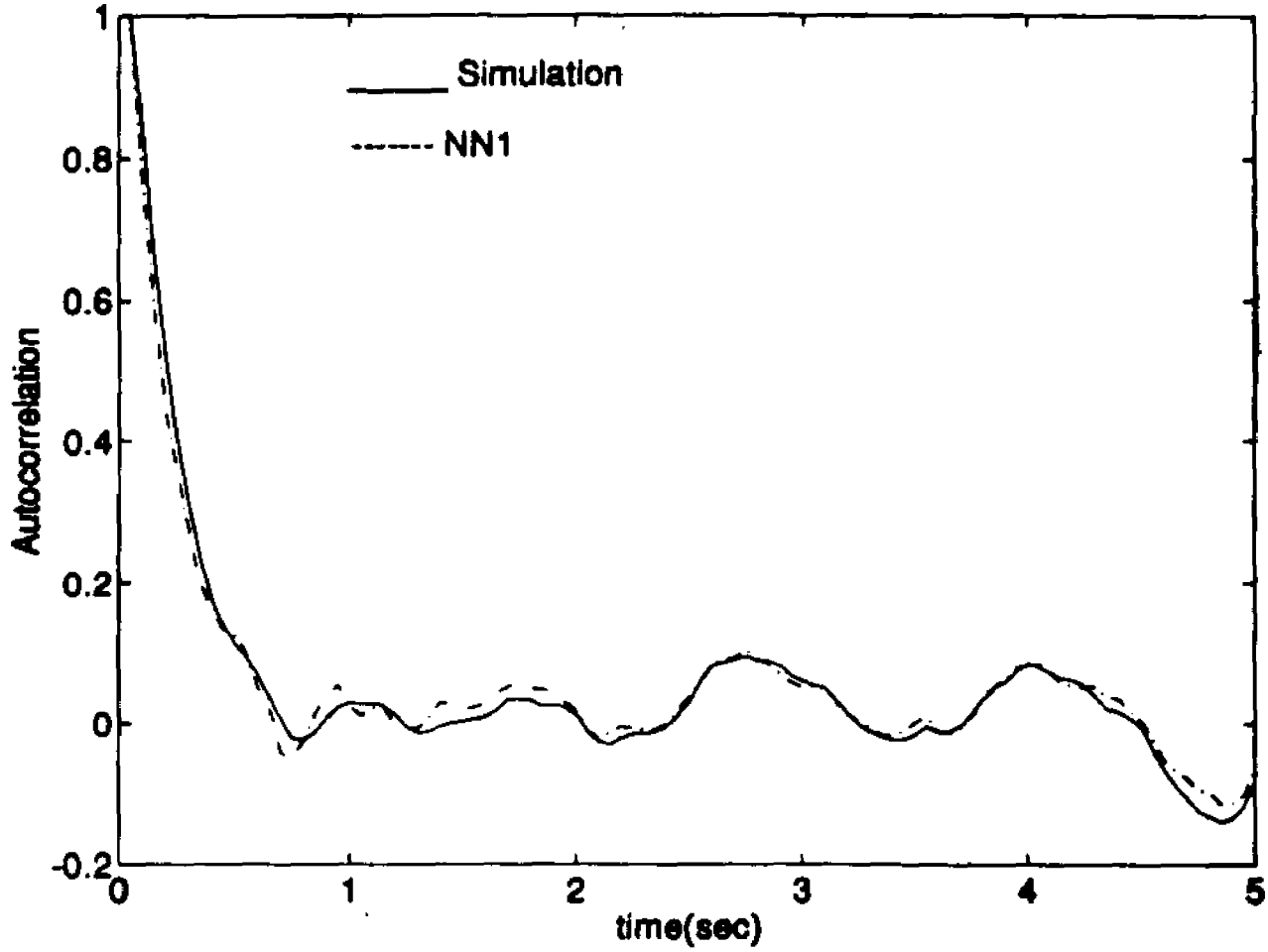


Fig. 3.5 Error signal no violation case

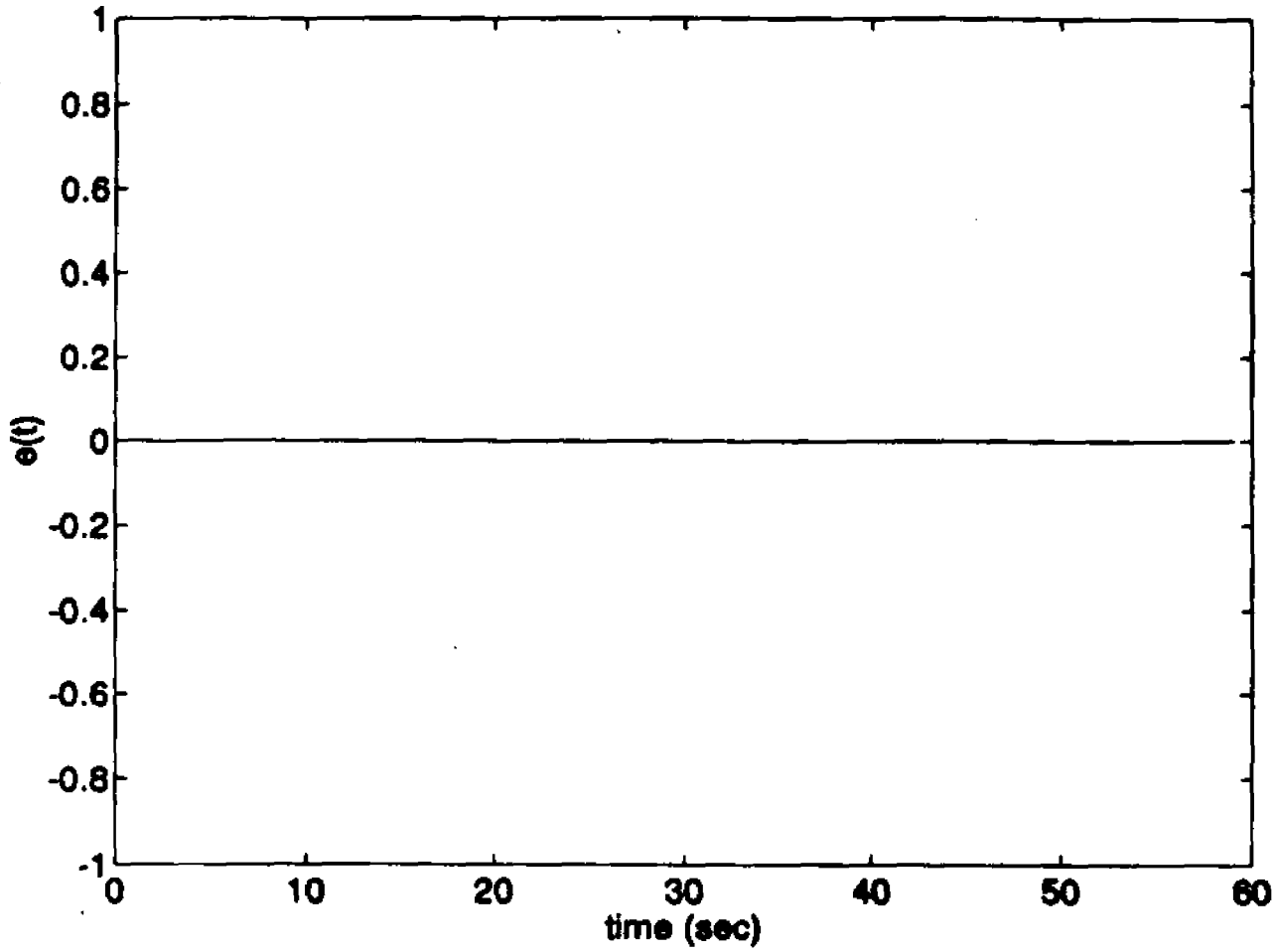
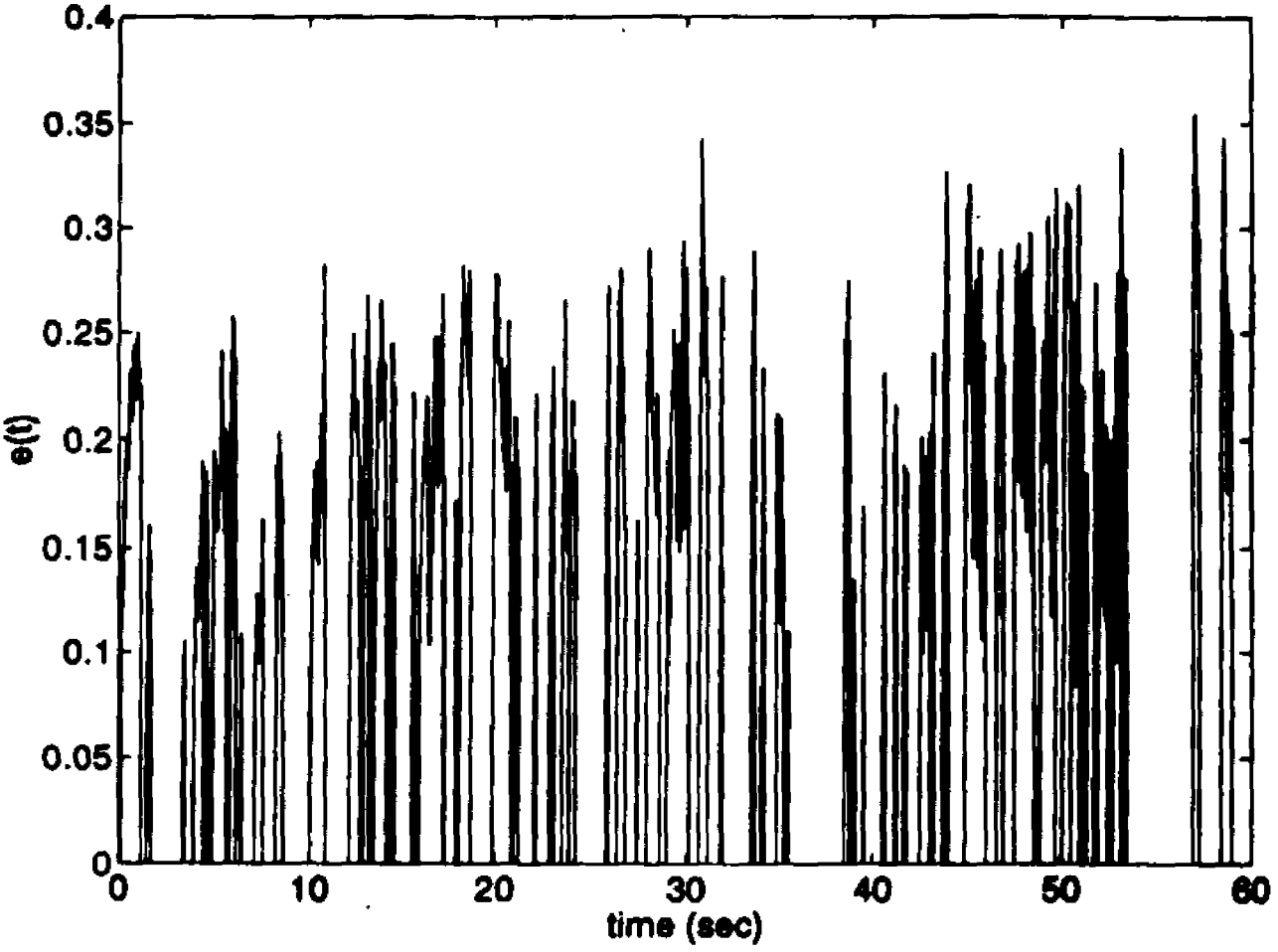


Fig. 3.6a Error signal peak and mean violations case



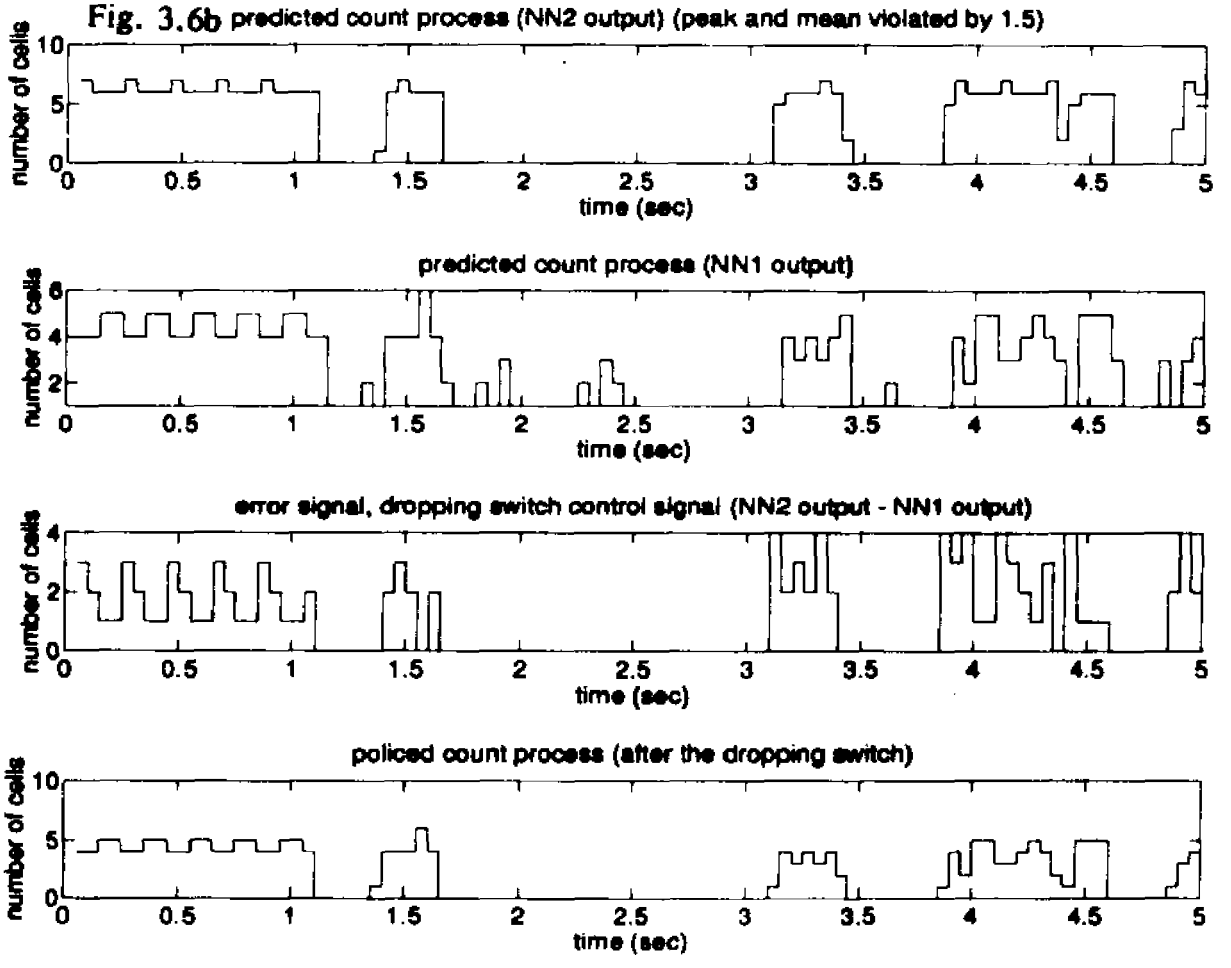
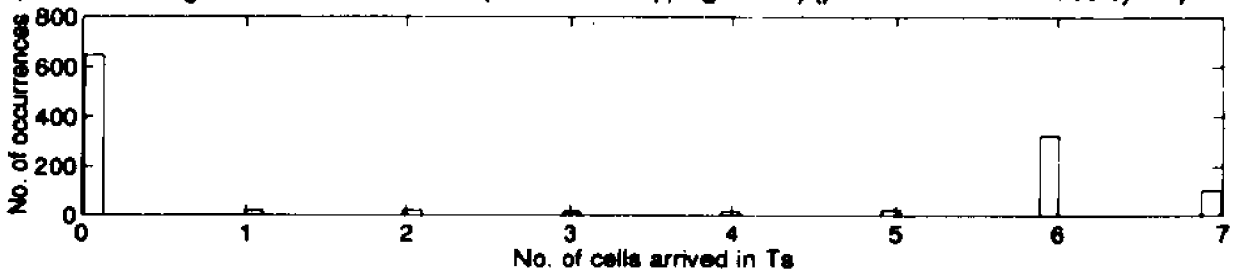
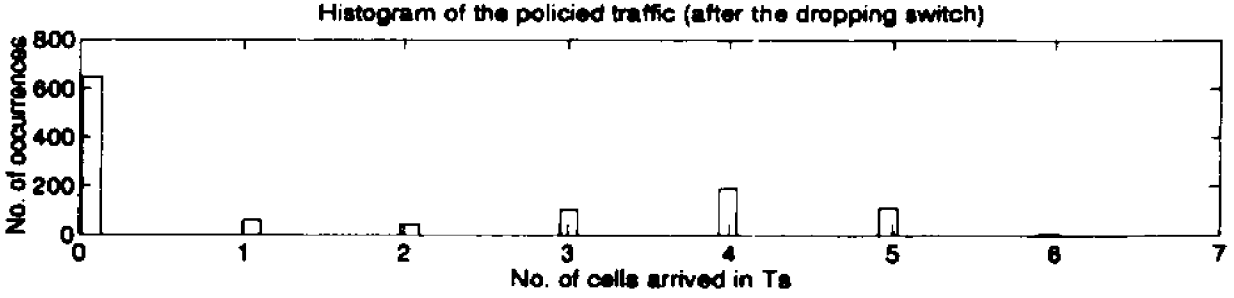


Fig. 3.6b predicted count process (NN2 output) (peak and mean violated by 1.5)

Fig. 3.6c Histogram of the offered traffic (before the dropping switch) (peak and mean violated by 1.5)





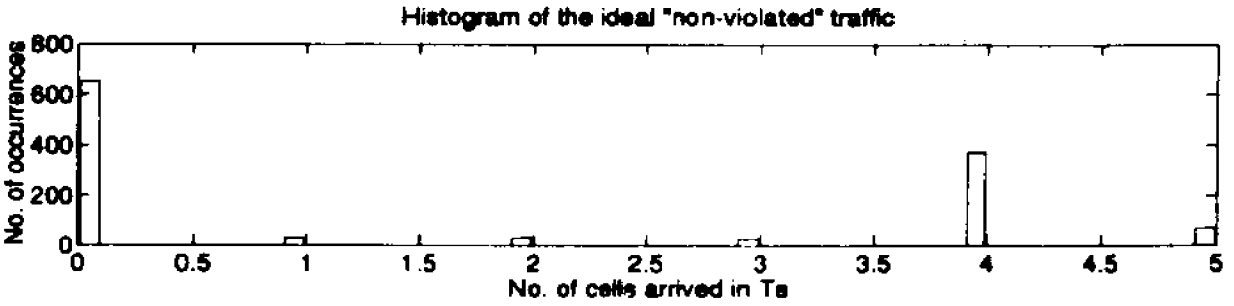


Fig. 3.7 Error signal peak violation case

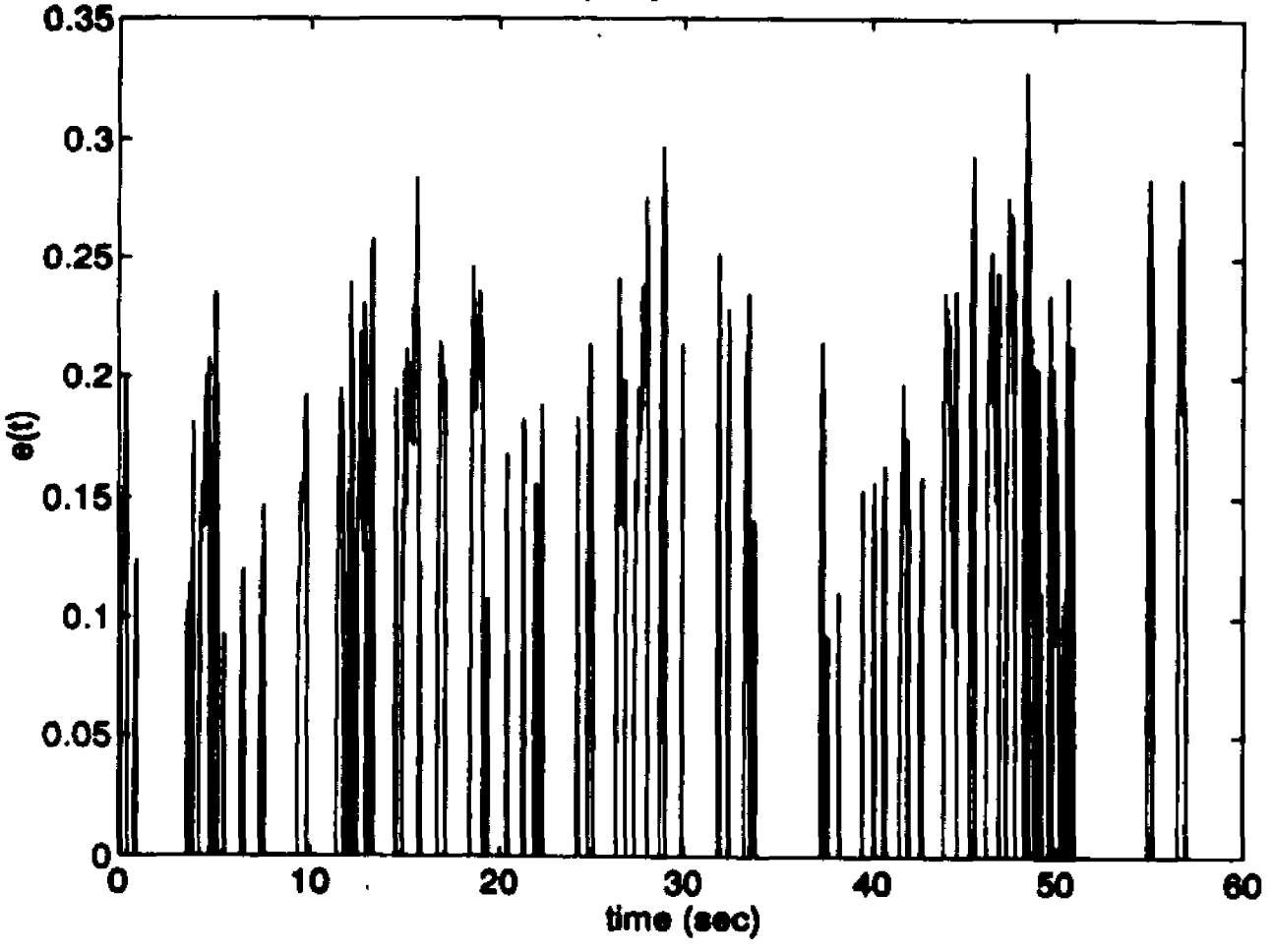
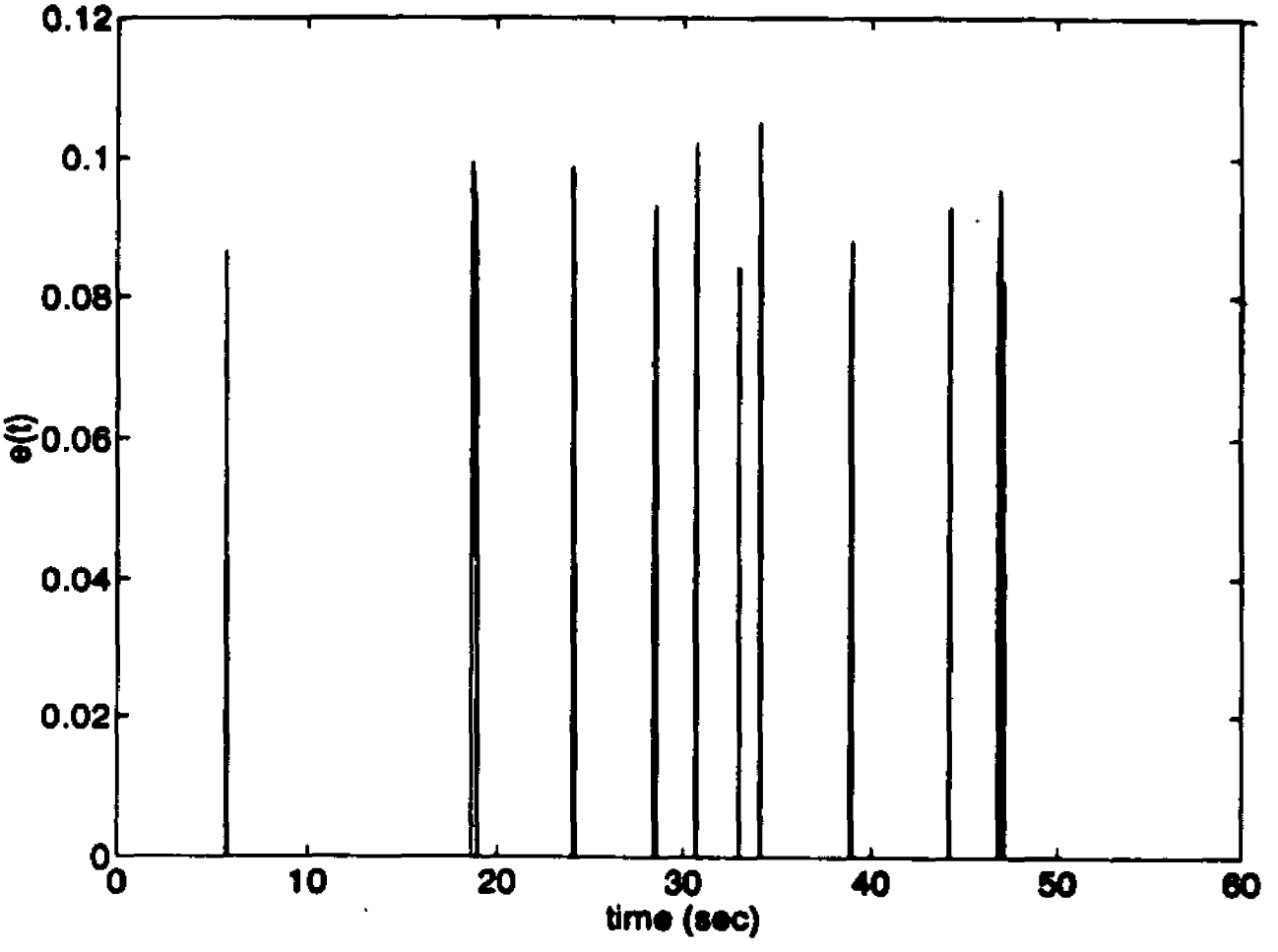


Fig. 3.8 Error signal mean violation case



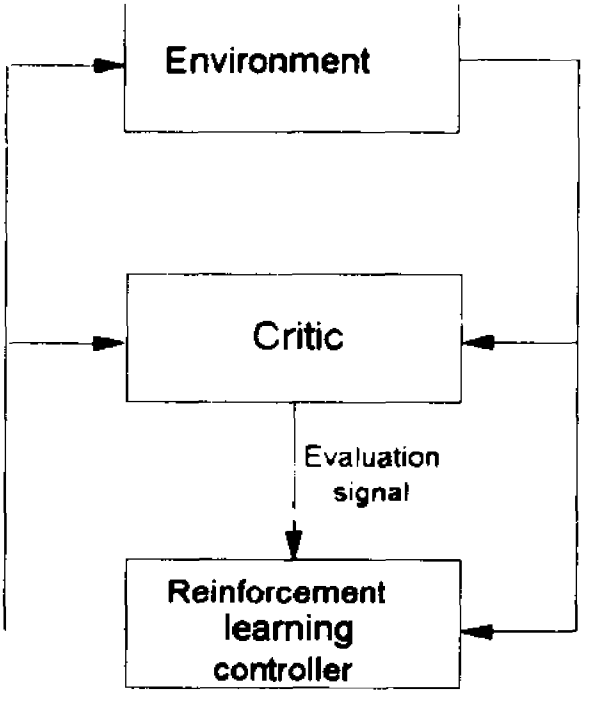


Fig. 3.9 Reinforcement learning control system

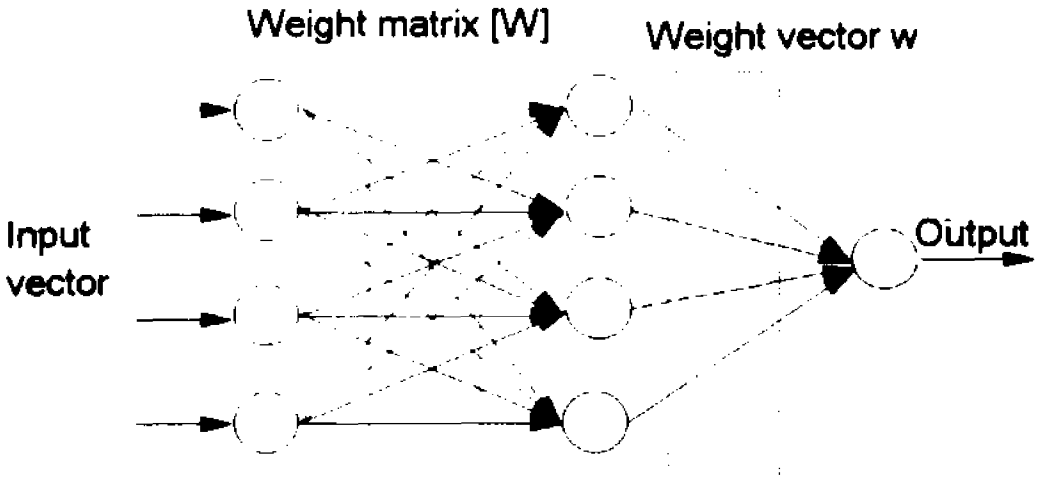


Fig. 3.10 Neural Network Controller

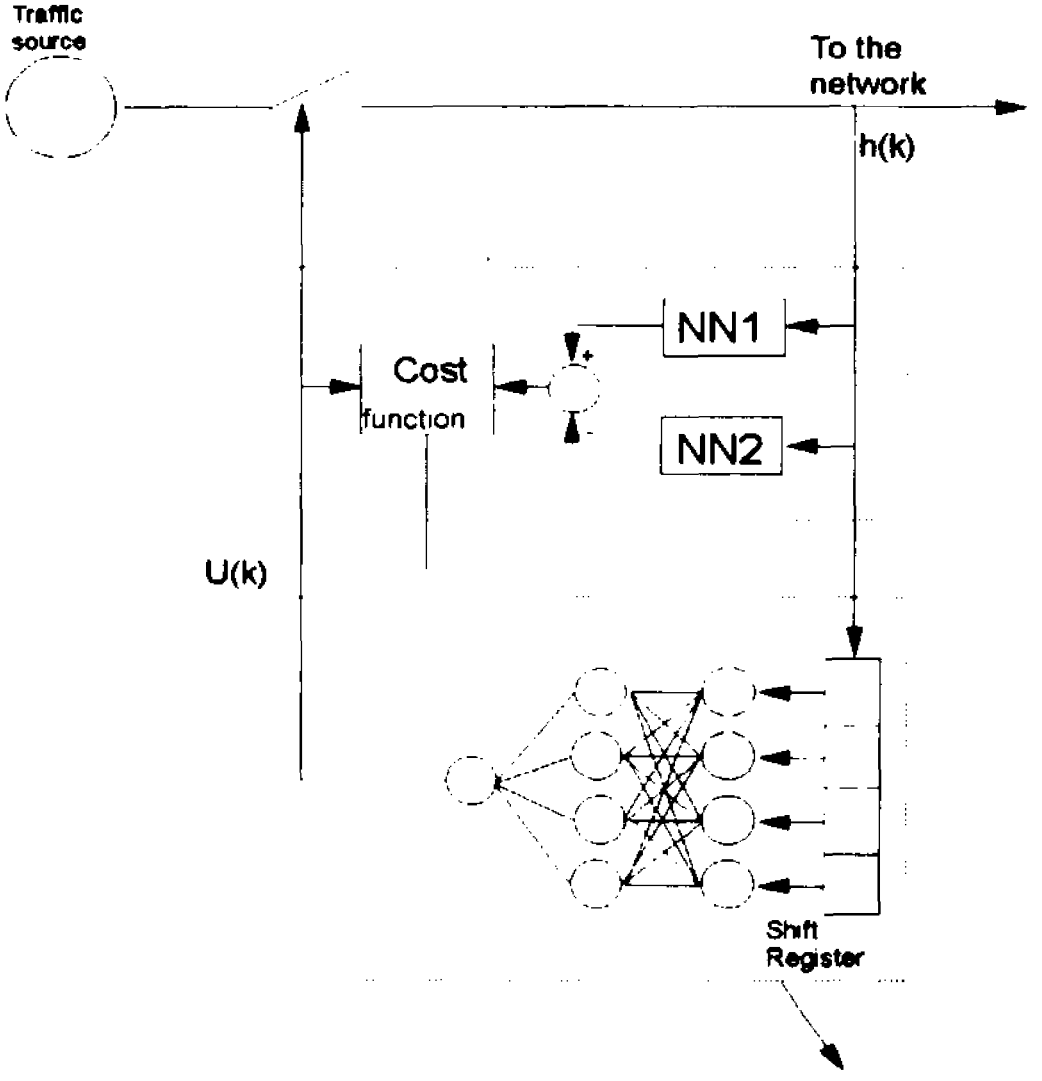


Fig. 3.11 Block diagram of reinforcement learning type controller

Fig. 3.12 Histogram of the count process, no violations

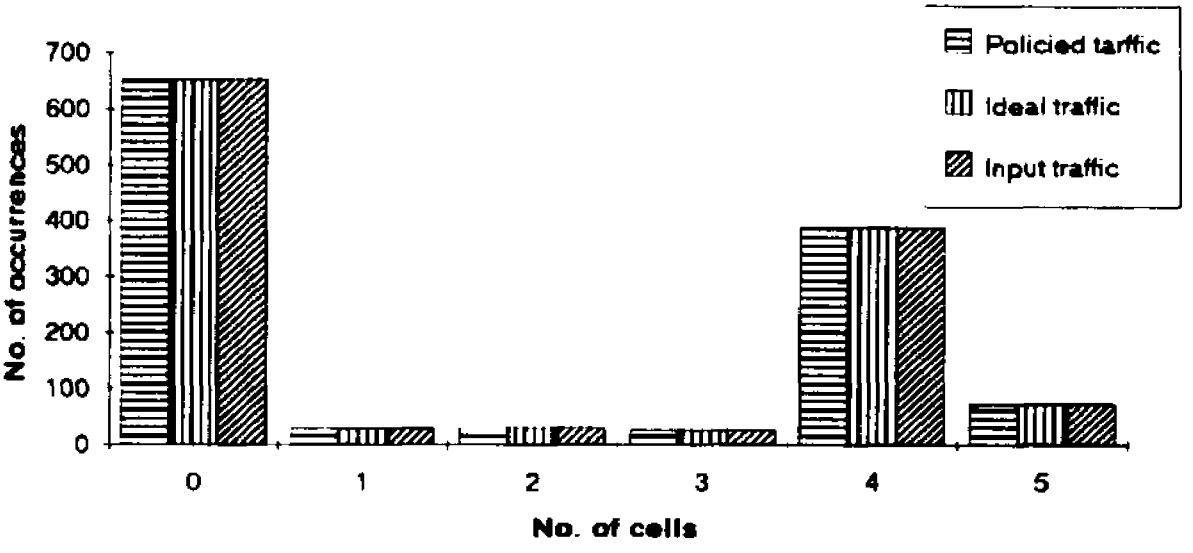


Fig. 3.13 Histogram of the count process, peak and mean violations

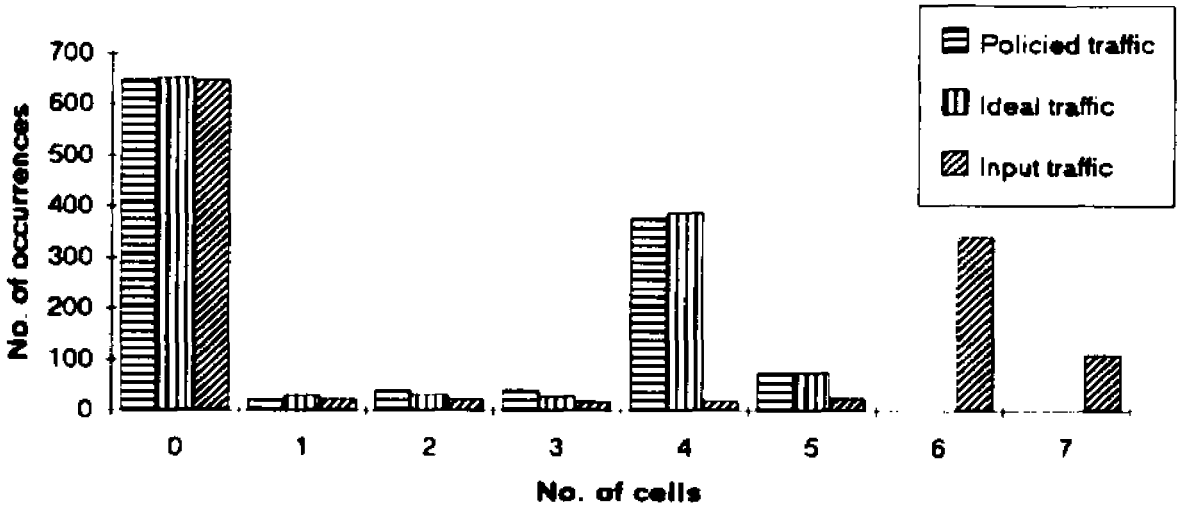


Fig. 3.14 Control signal, peak and mean violations

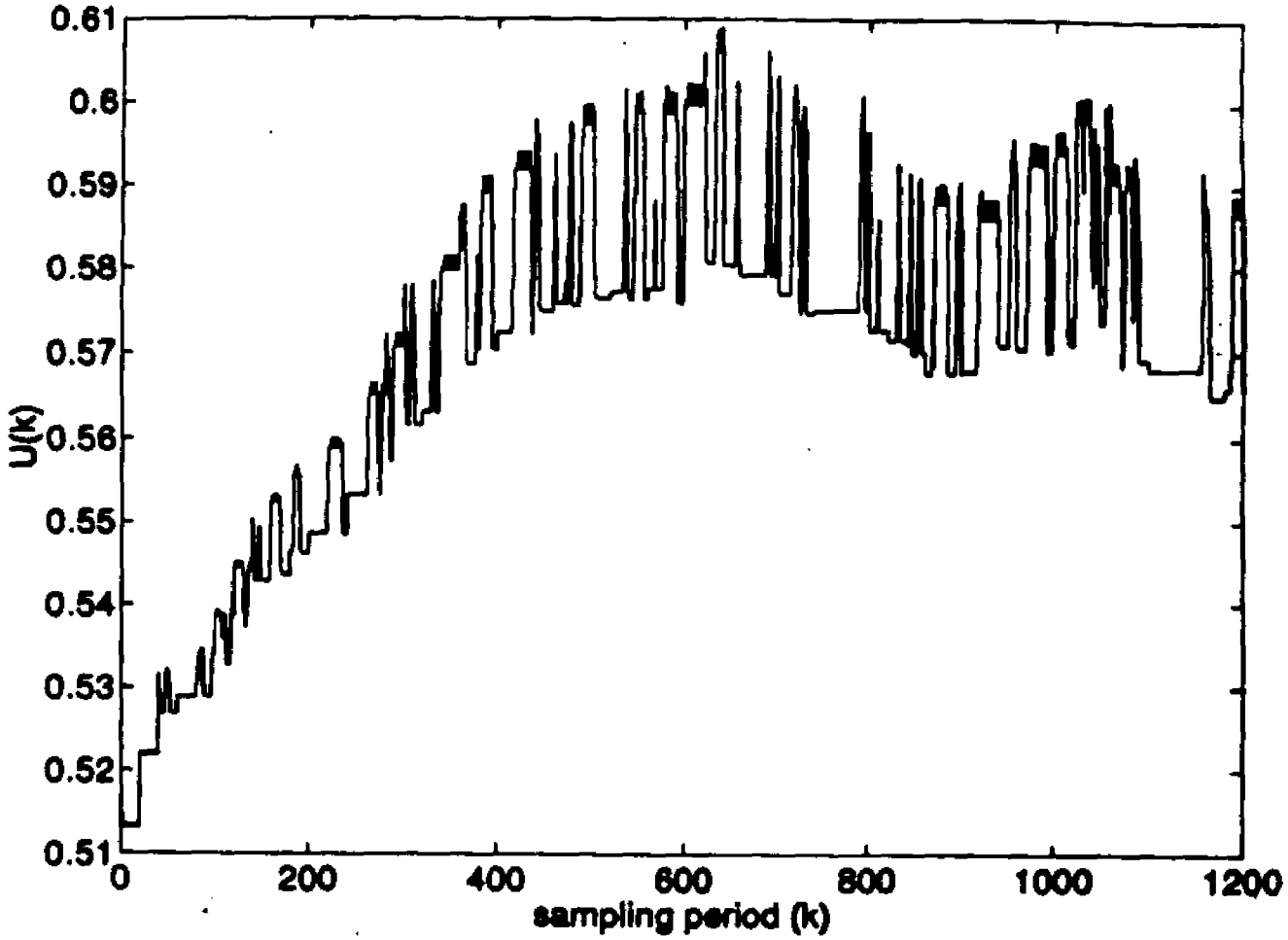


Fig. 3.15 Histogram of the count process, peak violation

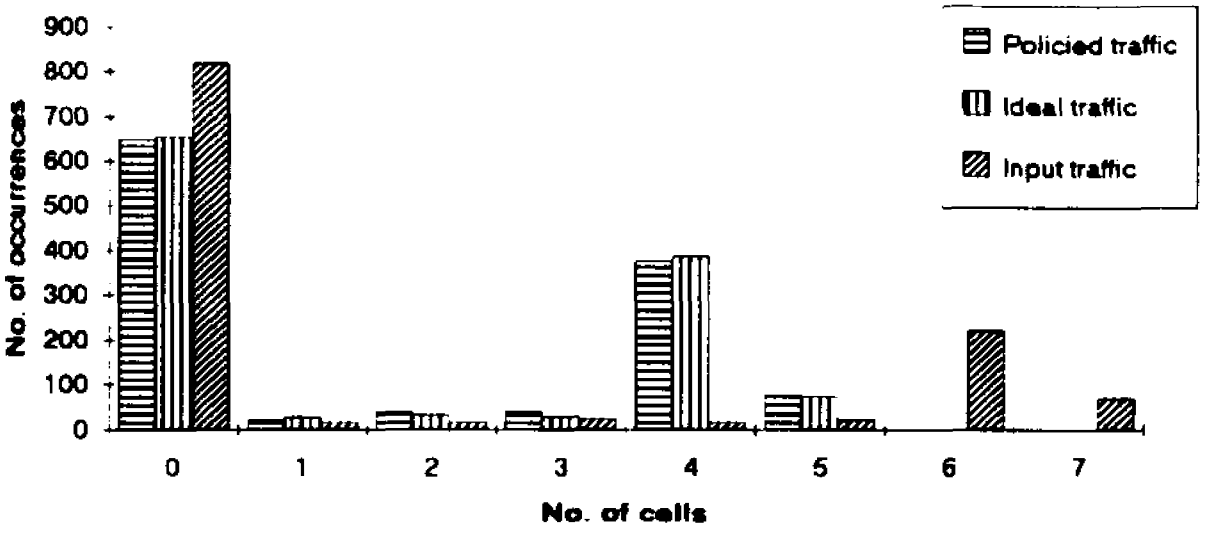


Fig. 3.16 Control signal, peak violation

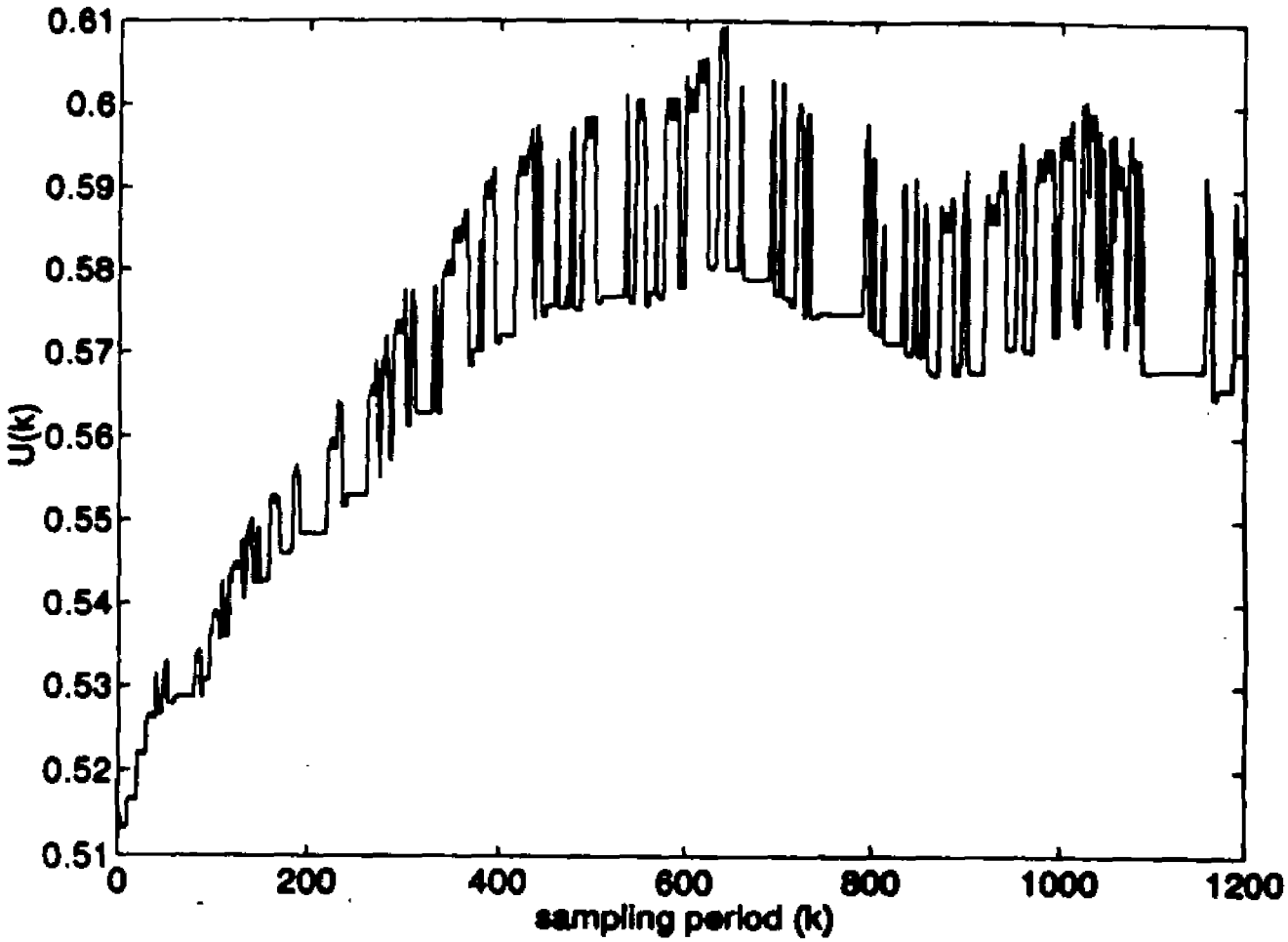
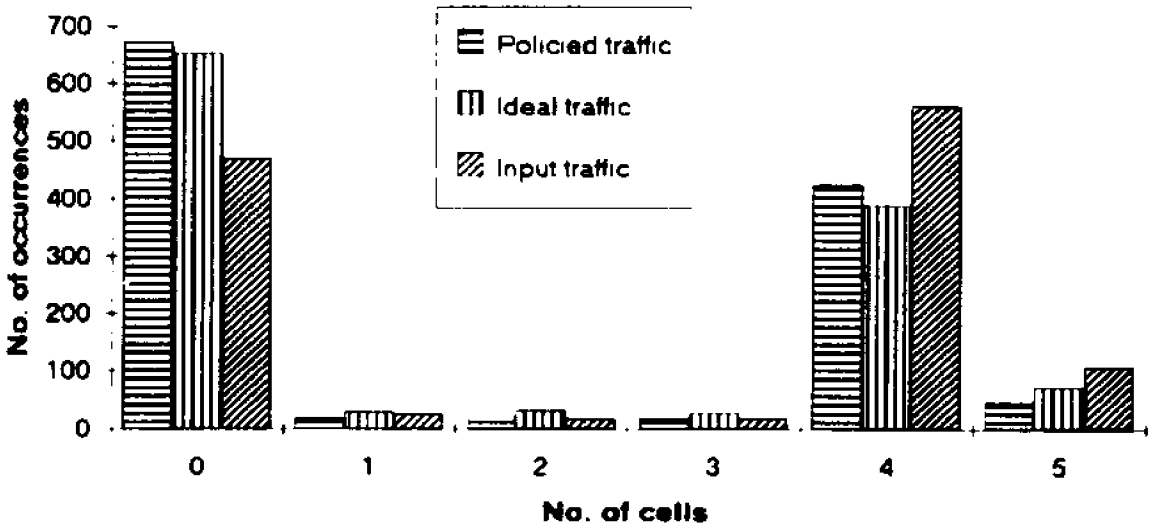


Fig. 3.17 Histogram of the count process, mean violation



IV. Congestion Control for ATM Networks by Reinforcement Learning NN

This chapter explains the function of the NN as congestion control in the ATM networks as shown in figure (1.2). ATM allows for statistical multiplexing at the cell level. Statistical multiplexing occurs when the capacity of an output channel is less than the sum of the peak connection bandwidth, but is larger tan their average total bandwidth requirement. The statistical gain is the factor by which the sum of the peak bandwidth exceeds the output channel's capacity. Statistical multiplexing, therefore, relies on the input channels begin bursty due to variable information transfer rates. Hence, the statistical gain directly depends on the bandwidth utilization and traffic characteristics of the input channels.

Achieving any statistical gain results in finite probability of cell-level overload or congestion. Controls are necessary to engineer the probability of occurrence of congestion and possibility to minimize the impact of congestion on end users. Congestion can be controlled to a limited extent by the provisions of buffers. The buffers will absorb excess information until the sum of the input rates drops below the output rate of the multiplexer. The larger the buffer, the greater the overload that can be absorbed, but at the expense of delay. Buffering can help avoid but can not eliminate congestion, as there is still a finite probability of buffer overflow, unless the duration of the overload is bound by some constraint. The only recourse in the case of buffer overflow is to discard information.

Congestion control in ATM networks is difficult because of the different quality

of service (QOS) requirements (e.g., cell loss rate (CLR), delay, and delay variability) that must be guaranteed for each type of the traffic. For example, some services such as voice, real-time video, and data for real time control have very strict delay requirements, whereas some services such as file transfers have very strict CLR requirements. More over, the high speed links (155 Mbps and over) has increased the propagation-bandwidth product. Which is a measure of the amount of traffic that can be in transient during a propagation delay time. Therefore, reactive form of congestion control based upon end-to-end feedback is inappropriate in the ATM environment.

Broadband ATM networks are subject to the uncertainties about traffic patterns and large variations in the network conditions. Traffic demands from users are often unpredictable. It is becoming increasingly difficult to model the statistical behavior of many types of diverse traffic and QOS requirements. Therefore, congestion control in ATM networks must be able to adapt gracefully to the dynamic behavior of traffic and the time-varying nature of network conditions, especially when new services are being continually introduced after network design and installation. Unlike traditional data network, performance-driven control is vital to congestion control design in ATM networks. Control algorithms should be effective in terms of taking action immediately at the onset of congestion and they should further, preferably, be incorporated into hardware implementation. These ongoing design requirements represent a significant challenge to the next generation of congestion control schemes.

IV.1 Congestion Control

This section presents an adaptive congestion control model using Neural Networks (NNs) which can meet the foregoing requirements of congestion control. This research is motivated by applying the powerful apparatus of NNs to congestion control problems for which conventional approaches have proven ineffective or constructive procedures do not exist for adaptive control. The presented algorithm is a preventive type and is applied at the access node of the network (i.e., at the User Network Interface (UNI)) [26]. The algorithm is a rate-based optimal control one and is implemented by the reinforcement learning NN. The algorithm uses a feedback control signal to throttle the peak bit-rate of the input arrival process to the input statistical multiplexer (see figure (4.1)). The feedback control signal is applied to the input source coder. This signal will control the source rate by decreasing the coding rate (number of bits per sample). The feedback control signal is produced by a NN-based controller. This feedback control signal is an optimal control signal in the sense that it is determined in such way to maximize the system performance. The system performance is measured by a certain performance index which combines two important system performance measures:

1) The CLR (i.e., the input multiplexer buffer overflow).

2) The level of the coding rate of the input source(s) coder(s).

We have solved the optimal control problem using the reinforcement leaning NN method. In this method, the weights of a NN-based controller are tuned continuously in order to maximize the system performance.

IV.2 Congestion Control Algorithm

The main objective of this section is to design a controller that can optimally minimize the input multiplexer buffer overflow while maintaining the coding rate (of the sources) at its maximum values most of the time. Since decreasing the coding rate (via the feedback control signal) could degrade the quality of the information (voice or video). It is evident that the optimal value(s) of the feedback optimal control signal (source throttling signal) that can achieve can not be determined quantitatively. Hence the best approach to the problem is to apply a reinforcement_based learning method which can tune the weights of a NN-based controller in order to achieve an efficient congestion control algorithm capable of maintaining the QOSs of the different users at the required values.

In this section, we present a congestion control algorithm for input voice/video arrival process using NN. The algorithm senses the congestion by measuring the number of the cells in the input multiplexer buffer at the access node to the network, and then take a control action by throttling the input sources's coding rates (see figure (4.1)). The control low is optimal one in the sense that it is determined to optimize the performance of the system. The performance index (cost function) of the system combines two performance measures. The first performance measure is defined in terms of the input multiplexer buffer overflows whereas the second performance measure is defined in terms of the level of the coding rate of the input sources. In this application, the optimal control low is the one that can minimize the performance index of the system by minimizing the first performance measure (buffer overflow) and maximizing the second

performance measure (the level of the coding rate). The importance of minimizing the buffer overflow is to minimize the CLR of the accepted calls during the call progress phase. On the other hand the importance of maximizing the level of the coding rate is to increase the quality of the coding information (voice or video).

It clear now that we the congestion control problem can be formulated as an optimization problem. It is very difficult for the classical optimal control methods to solve this problem because these methods rely upon a very accurate mathematical model for system to be controlled. In our case, these methods need a very accurate model that can characterize the arrival process with the distribution of number of the cells in the input multiplexer buffer. Since this model does not exist, more over if this model exists it should be adaptive for the continuous changes of the characteristic of the arrival process and it involves very long computation time. Hence it will be infeasible to be implemented on the real time. These limitations of using the classical optimal control methods in our application have motivated us to use reinforcement learning NN method to solve the optimal control problem in our application.

The reinforcement learning NN method evaluates the performance of the system in terms of the defined performance index, consequently generates an evaluation signal. This signal is function in the deviation of the system performance from the optimal one. This evaluation signal is used to adjust the weights of a NN controller such that the produced feedback control signal, from the controller, when applied to throttle the coding rates of the input sources, it results in maximization of the system performance (minimization of the performance index). Hence the reinforcement leaning NN method depends mainly on the defined performance index of the system and it always tends to minimize that index even though it might uses very simple mathematical model for the

system.

The proposed congestion control algorithm consists of a critic part and a NN controller part as shown in figure (4.2). The system to be controlled (the environment) composites the input traffic sources and the input multiplexer buffer at the access node to the network (UNI). The cells' arrival rate to the input multiplexer buffer depends upon the coding rate of the input traffic sources while the departure rate of the cells form the input multiplexer buffer depends upon the link capacity (in Mbps). In this application, the link capacity is constant, hence, the number of the cells in the multiplexer buffer is highly depends on the arrival process. The inputs to the control algorithm are taped delay values of the number of the cells in the multiplexer buffer and taped delay values of the feedback control signal. The controller output is the feedback control signal which is fed to the input traffic sources to change their coding rates. The critic part involves the performance index of the system (cost function) to be minimized. According to this cost function the critic part evaluates the system performance and generates an evaluation signal which is function in the deviation of the system performance from the desired optimal one, and is used to change the weights of the of the NN controller.

Figure (4.3) shows the NN scheme used as a controller. It has four neurons in the input layer, four neurons in one hidden layer, and one neuron in the output layer. The input layer neurons are fully connected to the hidden layer neurons via the weight matrix [W], on the other hand, the hidden layer neurons are connected to the output layer neuron via the weight vector [w]

As mentioned before, the reinforcement learning mechanism involves two parts, critic part and control part. The critic part of the system uses defined performance index (cost function) to adjust the weights of the NN controller in the control part. The cost

function has two performance measures, one presents the buffer overflow whereas the other one presents the level of the coding rate of the input traffic sources. The buffer overflow performance measure is the summation of the values of the buffer overflows over the optimization period. These buffer overflows are shown in figure (4.4.a) which illustrates the number of the cells in the input multiplexer buffer compared with the maximum length of that buffer (n_{max}). The buffer overflow performance measure is a summation of the number of the cells that exceeds the maximum buffer length during the overload periods, this summation is carried out over the optimization period. One optimization period might have more than one overload period. The level of the coding rate performance measure is defined by summing the squared differences between the maximum value of the feedback control signal and the instantaneous values of the feedback control signal. Figure (4.4.b) illustrates the feedback control signal versus time compared with the maximum value of that signal. The level of the coding rate of the input traffic sources performance measure is the summation of squared of these differences during the optimization period. these differences are shown in figure (4.4.b).

In other word, the objective of the proposed controller is to minimize the area above the maximum buffer length in figure (4.4.a) and minimize the area under the maximum control signal in figure (4.4.b).

We defined the cost function (J) as:

$$J(P) = \sum_{k=1}^{L} R_n S_n(k+1)\,(n_d(k+1) - n(k+1))^2 + R_u(u_d(k+1) - u(k+1))^2 \quad (4.1)$$

where

P: is the trial number

L: is the length of the optimization interval.

$n(k+1)$: is the number of cells in the buffer at the sample $k+1$

$n_d(k+1)$: is the congestion threshold level within the multiplexer buffer

$$n_d(k+1) \le n_{max} \qquad (4.2)$$

n_{max}: is the maximum length of the input multiplexer buffer

$S_n(k+1)$: is the given by

$$S_n(k+1)=1 \quad \text{if} \quad n(k+1) \ge n_d(k+1)$$

$$S_n(k+1)=0 \quad \text{if} \quad n(k+1) \le n_d(k+1)$$

$S_n(k+1)$ is used to make the cost function defined in terms of the buffer overflow, not just the difference between the number of the packets in the buffer and the buffer length.

R_n: is the weight value on the buffer overflow performance measure

$u(k+1)$: is the feedback control signal at the sample $k+1$

$u_d(k+1)$: is the desired value of the feedback control signal, it is also the maximum value of the feedback control signal.

$$u(k+1) \le u_d(k+1) \qquad (4.3)$$

R_u: is the weight value the on the level of the coding rate performance measure. This term reflects the weight on achieving a certain voice/video quality.

The controlled source coding rate is defined by the equation:

$$C(k) = u(k) * C_o \qquad (4.4)$$

where C_o is the maximum uncontrolled coding rate of the source, and C(k) is the controlled coding rate at sample k, where u(k) is the feedback control signal produced by the controller at sample k. ($u(k) \le 1$). Since $C(k) \le C_o$, therefor, the maximum

value of the control signal is unity. i.e., $u_d = 1$.

The cost function can be rewritten as:

$$J(P) = \sum_{k=1}^{L} R_n S_n(k+1)\,\epsilon_n^2(k+1) + R_u \epsilon_u^2(k+1) \qquad (4.5)$$

The term ϵ_n represents the cell loss and the term ϵ_u represents the deviation of the voice/video quality from its maximum value. Hence the feedback control signal is determined in order to minimize the cell loss rate in the mean time minimize the deviation of the voice/video quality from its maximum value.

The NN of the controller has three layers with no inner feedback loop and no direct connection from the input layer to the output layer as shown in figure (4.3). Both the hidden and the output layers have a sigmoid function [44] to provide the nonlinear mapping capability. The NN output is the environment input, u(k). This NN is expressed by the following equation [50]:

$$u(k) = f\{w^T(P)\, f[W(P)\, I(k)]\} \qquad (4.6)$$

where **w** is the weight vector from the hidden layer to the output layer, **W** is the weight matrix from the input layer to the hidden layer. Function f is the sigmoid function defined by the following equation [48]:

$$f(x) = \frac{1}{1+\exp(-xS)} \qquad (4.7)$$

where x is the input to the sigmoid function and S is the parameter determining its shape. I is the input vector to the controller, expressed as follows:

$$I^T(k) = [n(k), n(k-1), u(k-1), u(k-2),] \qquad (4.8)$$

Both the weight vector, w, and the weight matrix, W, are tuned so as to minimize the cost function, J, which is defined by (1.4). There are many methods to tune the weights of the NN such as steepest descent method, Newton's method, quasi Newton method, and conjugate gradient method [49]. Because the steepest descent method has the advantages of being simple and highly parallel, we used it to tune the weights of the NN as following:

$$w(P+1) = w(p) - \eta \frac{\partial J(P)}{\partial w(P)} \qquad (4.9)$$

$$W(P+1) = W(P) - \eta \frac{\partial J(P)}{\partial W(P)} \qquad (4.10)$$

where η is the parameter determining the convergence speed of the weights tuning. The work reported in [48] and [51] has proved the conversion of the cost function using the weight tuning algorithm given in equations (4.9) and (4.10). In order to realize the reinforcement mechanism shown in (4.9) and (4.10), it is necessary to derive the details of $(\partial J(P)/\partial w(P))$ and $(\partial J(P)/\partial W(P))$ which is given in appendix II.

IV.3 Simulation

The controller, the voice sources and video sources are simulated on SunSparc work station using the C language. A voice source is simulated using the ON/OFF model and a video source is simulated using the first order autoregressive (AR) Markov process.

The coding rate of the voice/video sources will be reduced in steps (corresponding

to number of bit per sample). The controlled source coding rate will take the following values according to the feedback control signal:

$$C(k) = C_o$$

$$C(k) = 0.75 C_o$$

$$C(k) = 0.5 C_o$$

These three values of the coding rate are corresponding to three values of the feedback control signal:

u(k)=1, u(k)=0.75, and u(k)=0.5, respectively.

Hence, the control signal produced by the controller is quantized to those three values before applied to the voice sources.

The number of the cells in the buffer is measured at every sampling period (T_s), and also the control signal is applied at every T_s. At given sampling instant (k) the controller measures the number of the packets in the buffer, and also apply the feedback control signal.

The values of the sampling period T_s, the length of the optimization period L, the weights R_n and R_u, affects the performance of the system in terms of the CLR and The voice/video quality.

several experiments have been performed using different values of the above parameters, and the corresponding of values of the CLR and voice/video quality also have been calculated, to show the performance of the controller.

IV.4 Results

In this section we show some of the experiments using different values of the control parameters. In these experiments, the voice sources are simulated using the ON/OFF model with $1/\beta = 0.350$ sec and $1/\alpha = 0.650$ sec. The video sources are simulated using the first order autoregressive process mentioned above. The parameter used in this model are: a=0.8781, b=0.1108, the mean of G(n) is 4.3 Mbps, and the variance of G(n) is unity.

Experiment (4.1):

In the first experiment, we considered infinite length multiplexer buffer, and packets from 150 voice sources were statistically multiplexed, with link capacity of 1.53 Mb/sec. We assumed in ,this experiment only, that the coding rate of a voice source can be changed smoothly from C_0 to zero, i.e. no quantization is made to the feedback control signal. The value of the control parameters were as follows:

$R_u=0.01$, $R_u=10$, $n_d=50$, $\zeta=0.1$, $T_s=0.01$sec. $L=10T_s$.

Figure (4.5) shows the feedback control signal and the uncontrolled aggregate arrival process resulted from the 150 voice sources. The feedback control signal starts at about 0.1 and grows toward 1. During this grows, the control signal gets some drops in its value at a specific times. These drops are because the input process has experienced some high burstiness periods. This indicates that the controller has successfully sensed the increased in the burstiness in the arrival process and consequently has dropped its output feedback control signal, to throttle the coding rate of the arrival process. Figure (4.6) shows the number of the cells in the buffer versus time and figure (4.7) shows the

cost function versus time. The cost function starts with high value and converges to zero and gets some increases at the moments of high burstiness of the arrival process, whoever it converges to zero very rapidly. This ensures the conversion of the weight tuning algorithm given in equations (4.9) and (4.10).

Experiment (4.1):

Starting from this experiment, we used a finite buffer length of 50, to limit the delay to 125 μsec. 150 voice sources were involved with the link capacity of 1.53 Mb/sec (about 110% utilization). The control parameters were:

R_n=0.4, R_u=10, n_d=30, ζ=0.2, T_s=0.01sec. L=10T_s .

Figure (4.8) shows the feedback control signal and the number of the cells in the buffer versus time. The control signal oscillates between 0.5, 0.75, and 1, this corresponding to coding rate of o.5C_0, o.75C_0, and C_0. Figure (4.9) shows the histogram of the number in the cells in the buffer compared with that for the uncontrolled case, it demonstrates the effect of the controller in reducing the buffer overflow. The CLR in this experiment is 10^{-4}. Figure (4.10) shows the histogram of the control signal, from which it is clear the most of the time the sources are operated at the 0.75 of the maximum coding rate (i.e., at 0.75C_0), and the corresponding voice quality are 0.72 of the maximum voice quality. Figure (4.11) shows the cost function versus time, it starts with some high number and converges very rapidly and it experiences some perturbations due to the increase in the burstiness in the input process, whoever it converges.

Experiment (4.3):

In this experiment, we show the efficacious of the controller in decreasing the CLR and increasing the voice quality. The control parameters were:

R_n=0.2, R_u=10, n_d=30, ζ=0.2, T_s=0.01sec. L=5T_s .

Figure (4.12) shows the CLR and the voice quality versus time, where the CLR decreases with the time and also the voice quality gets better with time. This because that the weights of the NN controller start with arbitrary random values, and the critic part evaluates the performance of the system via the cost function and then tunes the weights of the NN controller. In this experiment, the weights were updated at every L ($=5T_s$) and after 600 sec the controller has achieved a CLR of 10^{-4} and a voice quality of 0.8 of the maximum voice quality.

Experiment (4.4):

This experiment demonstrates the effect of the parameters R_n, and R_u on the performance of the controller.The control parameters were:

$n_d=30$, $\zeta=0.2$, $T_s=0.01$sec. $L=5T_s$. The ratio R_n/R_u are changed from 0.01 to 0.1. Figure (4.13) shows the CLR and the voice quality versus R_n/R_u. As shown, as the weight of the CLR-part of the cost function, R_n increases the CLR gets better (decreases) and the voice quality decreases and vice versa. This justifies the tarde off between decreasing the CLR and increasing the voice quality, whoever, one might increase the weight of either one to achieve certain required performance. Hence the proposed controller allows the network operator to choose the ratio of satisfaction of the required QOS and voice quality. for example, he might choose to decrease the CLR in the expense of decrease the voice quality, or vice versa, or choose to maintain the CLR low and also maintain the voice quality at an acceptable level.

Experiment (4.5):

This experiment shows the effect of the length of the optimization interval (L). The control parameters were:

$n_d=30$, $\zeta=0.2$, $T_s=0.01$sec. The ratio R_n/R_u were changed from 0.01 to 0.1, and L

takes the values of $2T_s$, $5T_s$, $10T_s$, $20T_s$. As can be shown from the group curves given in figure (4.14), for low value of L, we get better performance in terms of decreasing the CLR and increasing the voice quality as shown in the curve for $L=2T_s$. Whoever, for high values of L we get low CLR and low voice quality, this because that the weights of the NN controller are updated, by the critic part, at every L. Hence, by increasing L, there will be a considerable time elapsed between the increase of the burstiness of the arrival process and take the swift action (i.e. tune the weights of the controller in order to accommodate with this increase in the burstiness).

Experiment (4.6):

In this experiment, we show the CLR versus load for both the controlled system and the controlled system, we also show the voice quality versus time for the controlled system. The control parameters were:

$R_n=0.2$, $R_u=10$, $n_0=30$, $\zeta=0.2$, $T_s=0.01$sec. $L=5T_s$.

Figure (4.15) shows the CLR versus load for both the controlled system and the uncontrolled one. It is clear the effectiveness of the controller in maintaining the CLR low while it is continuously increases with the load for the uncontrolled system. Figure (4.16) shows the voice quality versus load. It decreases as the load increases, whoever it is still at an acceptable value.

The following experiments were performed to test the proposed algorithm with the video arrival process. 16 video sources were statistically multiplexed at the input multiplexer buffer. We used a finite buffer of length of 20 cells with the link capacity of 62.4 Mbps to limit the delay to 125 μsec.

Experiment (4.7):

In this experiment, the control parameters were:

$R_n=0.2$, $R_u=10$, $n_d=15$, $\zeta=0.5$, $T_s=0.001$sec. $L=10T_s$.

Figure (4.17) shown the feedback control signal and the number of the video cells in the buffer versus time. Figure (4.18) shows the histogram of the number of the video cells in the buffer under the control of the proposed algorithm compared with the histogram of that number for the uncontrolled case. The controller has reduced the CLR to 10^{-6} in the mean time maintained the video sources operating at their maximum coding rate at most of the experiment time, as shown in Figure (4.19). It shows the histogram of the feedback control signal, the correspond video quality is 0.889 from its maximum value.

Experiment (4.8):

In this experiment, we show the effect of the parameter R_n in determining the value of the obtained CLR. The control parameters were:

$R_n=0.3$, $R_u=10$, $n_d=15$, $\zeta=0.5$, $T_s=0.001$sec. $L=10T_s$.

The value of R_n is increased in this experiment compared with the last experiment ($R_n=0.2$). Figure (4.20) shows the histogram of the number of the cells in the buffer and the corresponding CLR is 10^{-7}. Figure (4.21) shows the histogram of the feedback control signal, from which it is clear that the control has maintained the video sources operating at 0.75 of their maximum coding rate at most of the time, and the corresponding video quality is 0.85 of the maximum value. It is clear form these results that by putting more weight on decreasing the buffer overflow (by increasing R_n), we get less CLR in the expense of decreasing the video quality. This experiment assures the sensitivity of the proposed algorithm to the relative values of the parameters R_n and R_u. Hence, these two parameters can be used by the network operator to achieve certain performance.

Experiment (4.9):

In this experiment, we show the CLR versus load for both the controlled system and the

controlled system, we also show the video quality versus time for the controlled system. The control parameters were:

R_n=0.2, R_v=10, n_d=15, ζ=0.2, T_s=0.001sec. L=10T_s .

Figure (4.22) shows the CLR versus load for both the controlled system and the uncontrolled one. It is clear the effectiveness of the controller in maintaining the CLR low while it is continuously increases with the load for the uncontrolled system. Figure (4.23) shows the video quality versus load. It is decreased as the load increases, whoever it is still at an acceptable value.

From all of the above experiments we conclude that the proposed congestion control algorithm is capable of controlling the voice arrival process and the video arrival process as well. It gives the maximum control for the CLR and the required quality of the coded voice and video information.

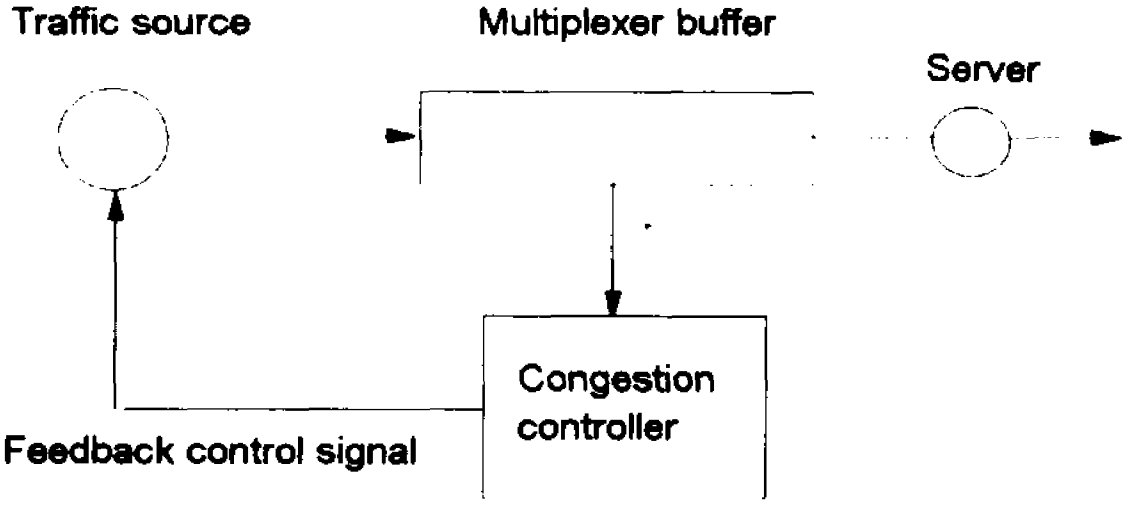


Fig. 4.1 Congestion controller

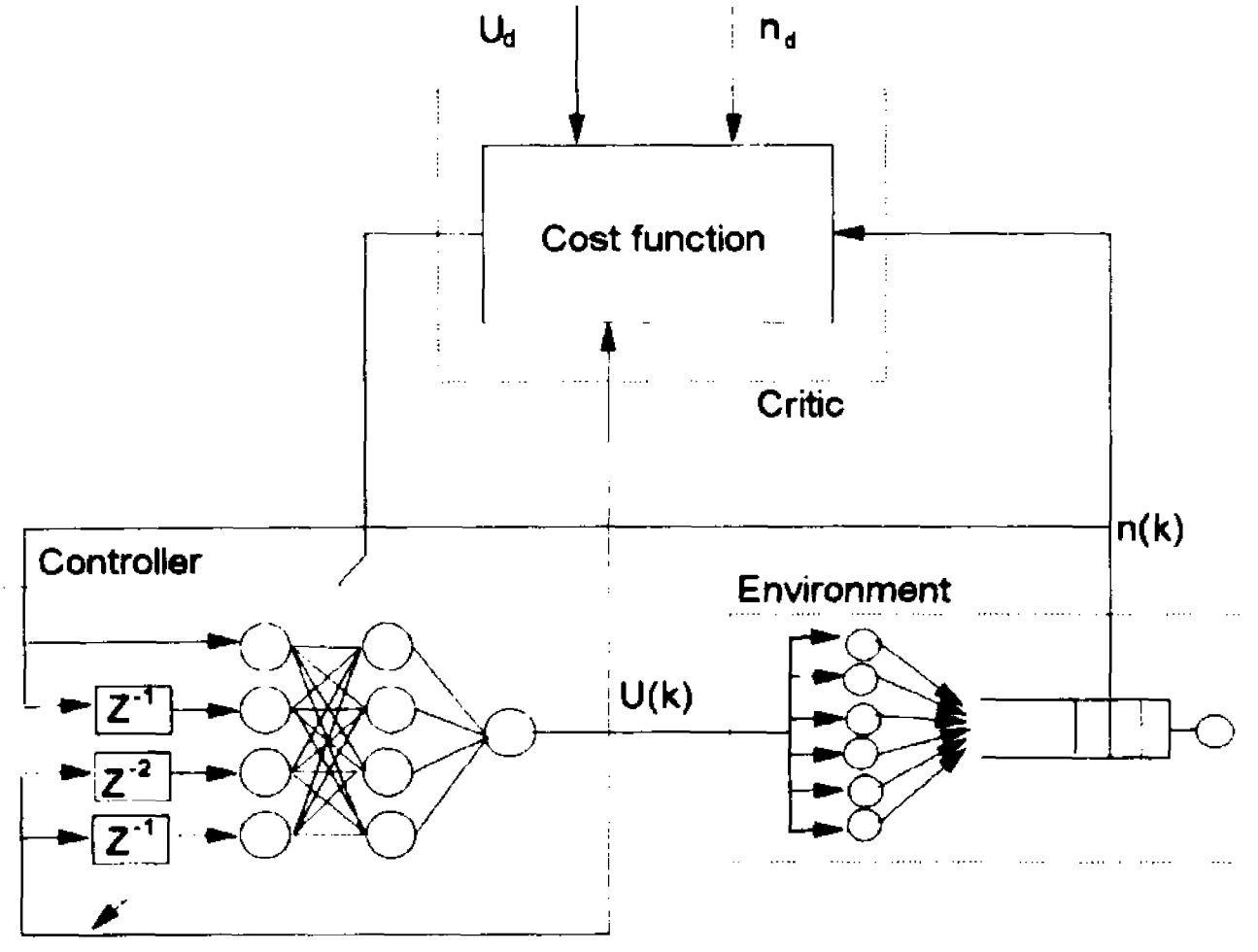


Fig. 4.2 Neural Network congestion controller

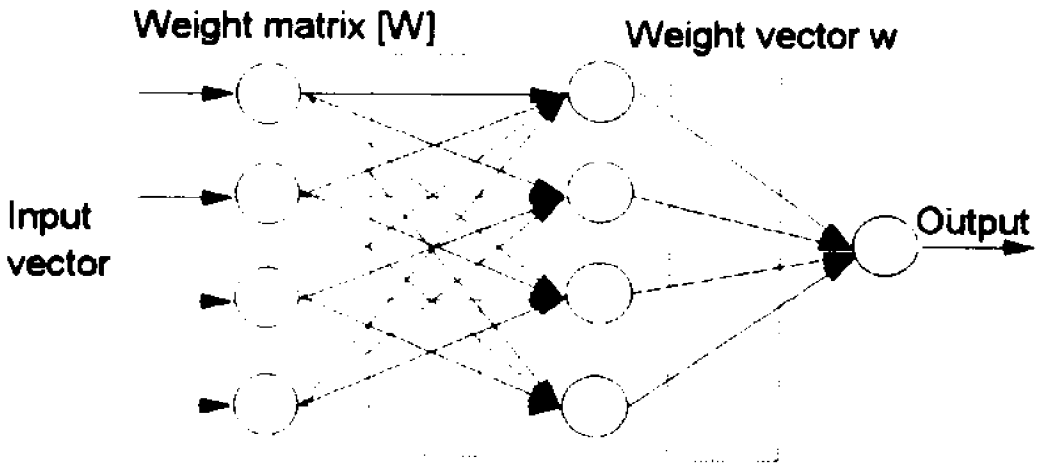


Fig. 4.3 Neural Network Controller

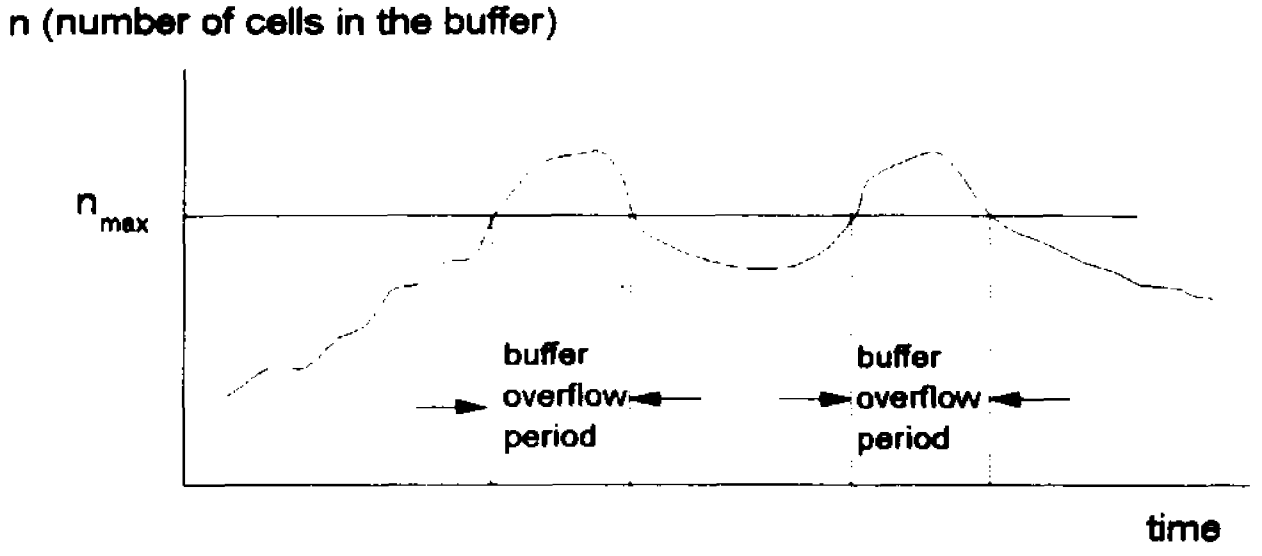


Fig. 4.4a number of cells in the buffer versus time

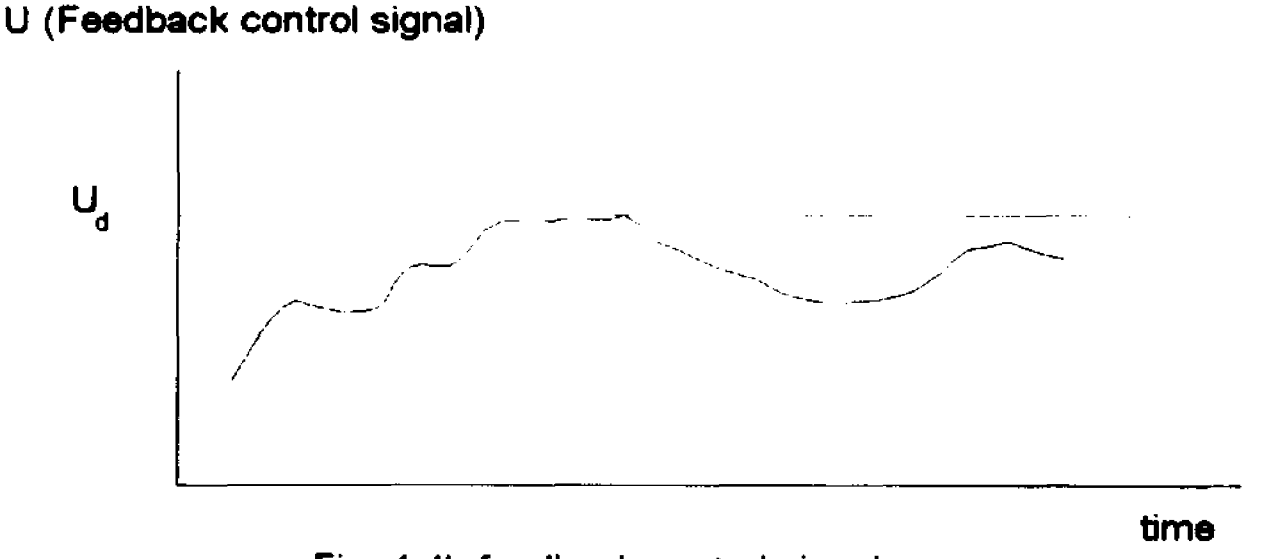
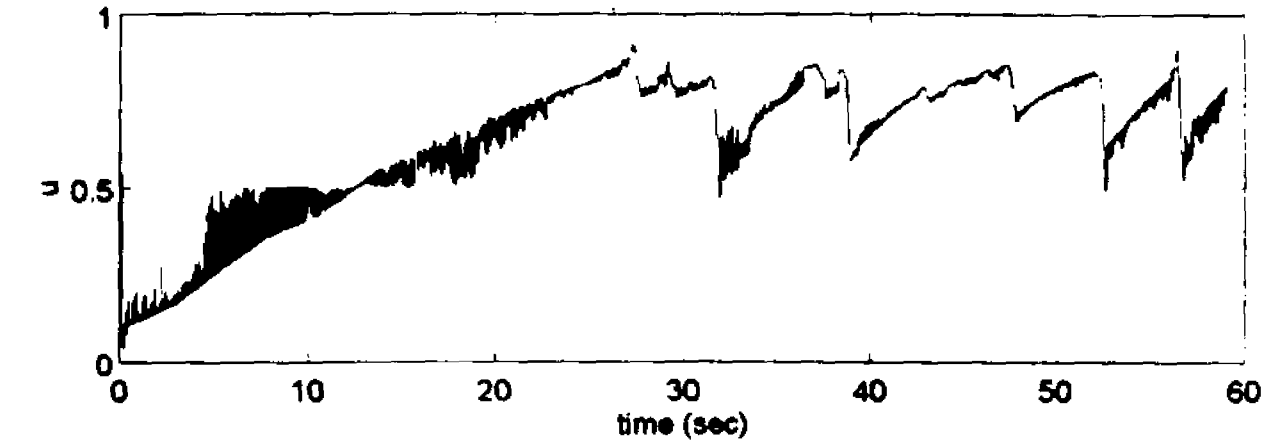


Fig. 4.4b feedback control signal versus time

Fig. 4.5 Feedback control signal



The input arrival process

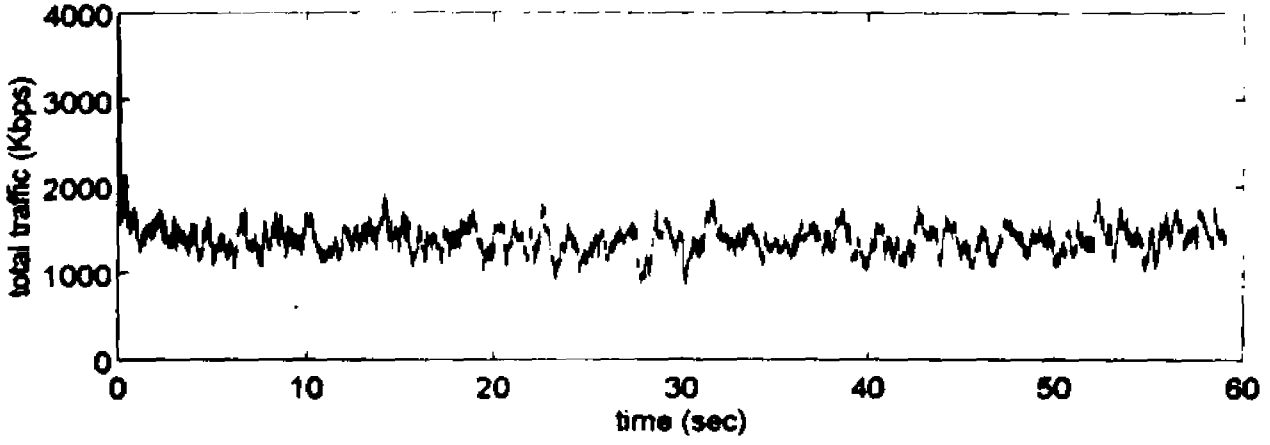


Fig. 4.5 "cont." Feedback control signal

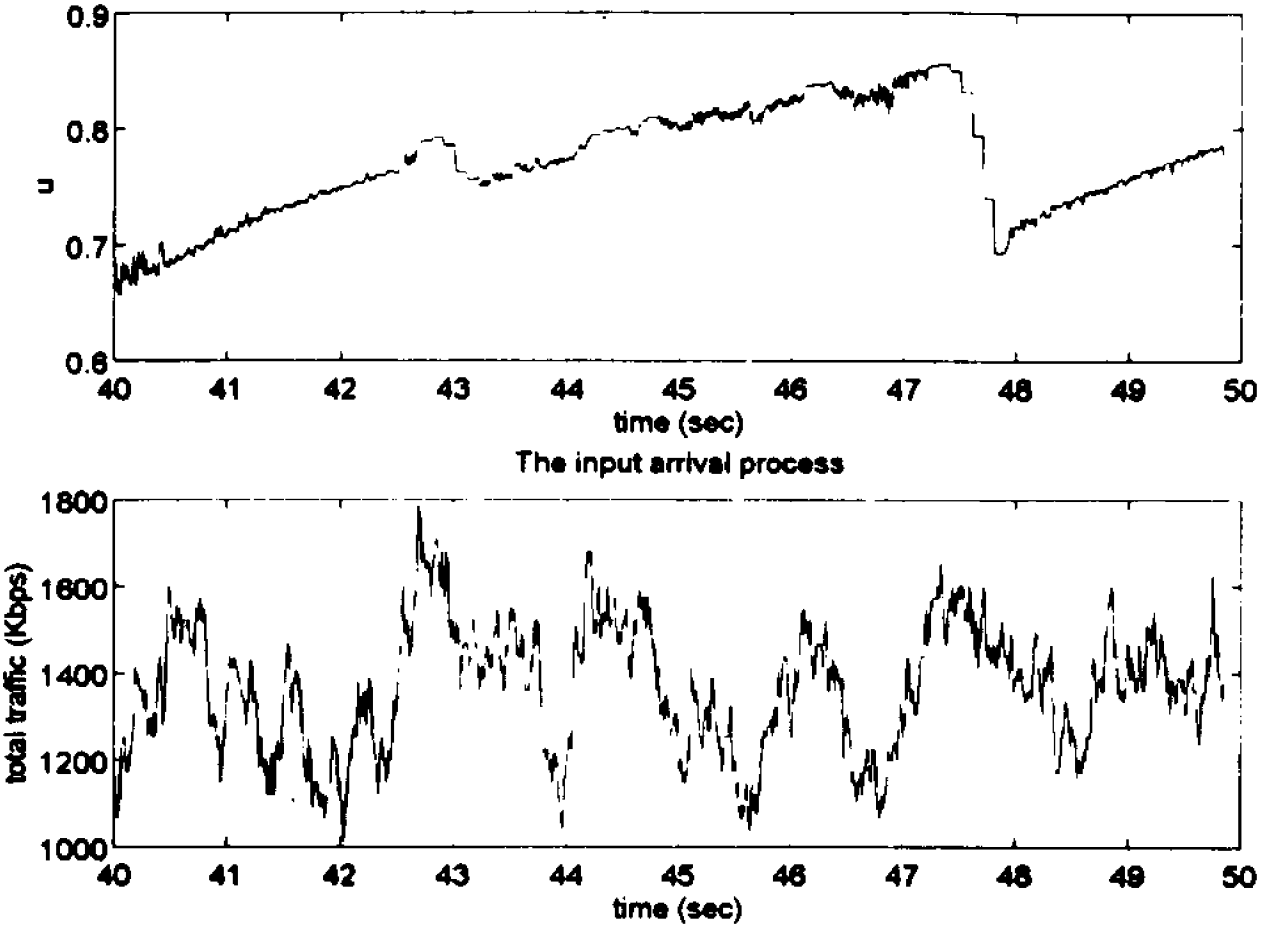


Fig. 4.6 Feedback control signal

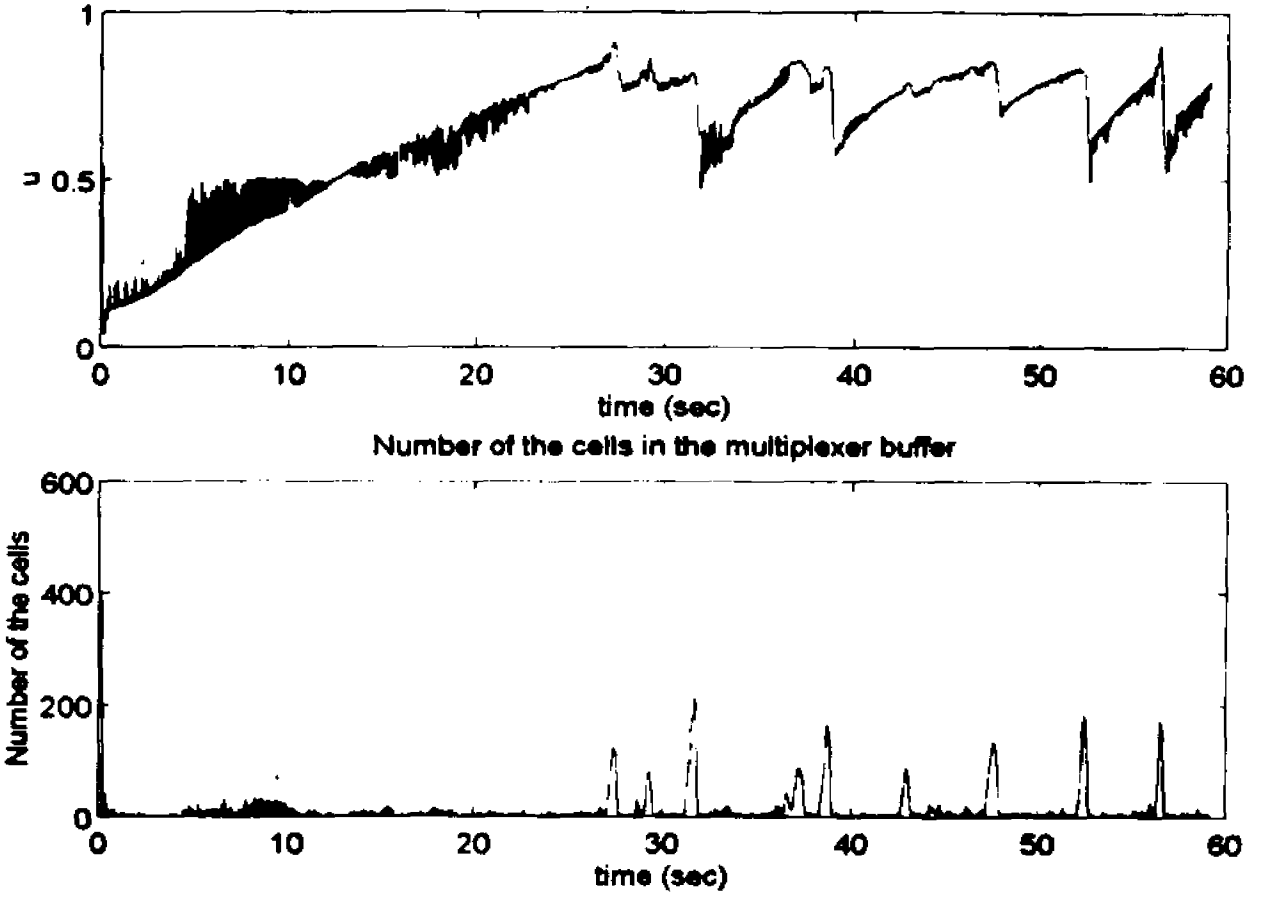


Fig. 4.7 Feedback control signal

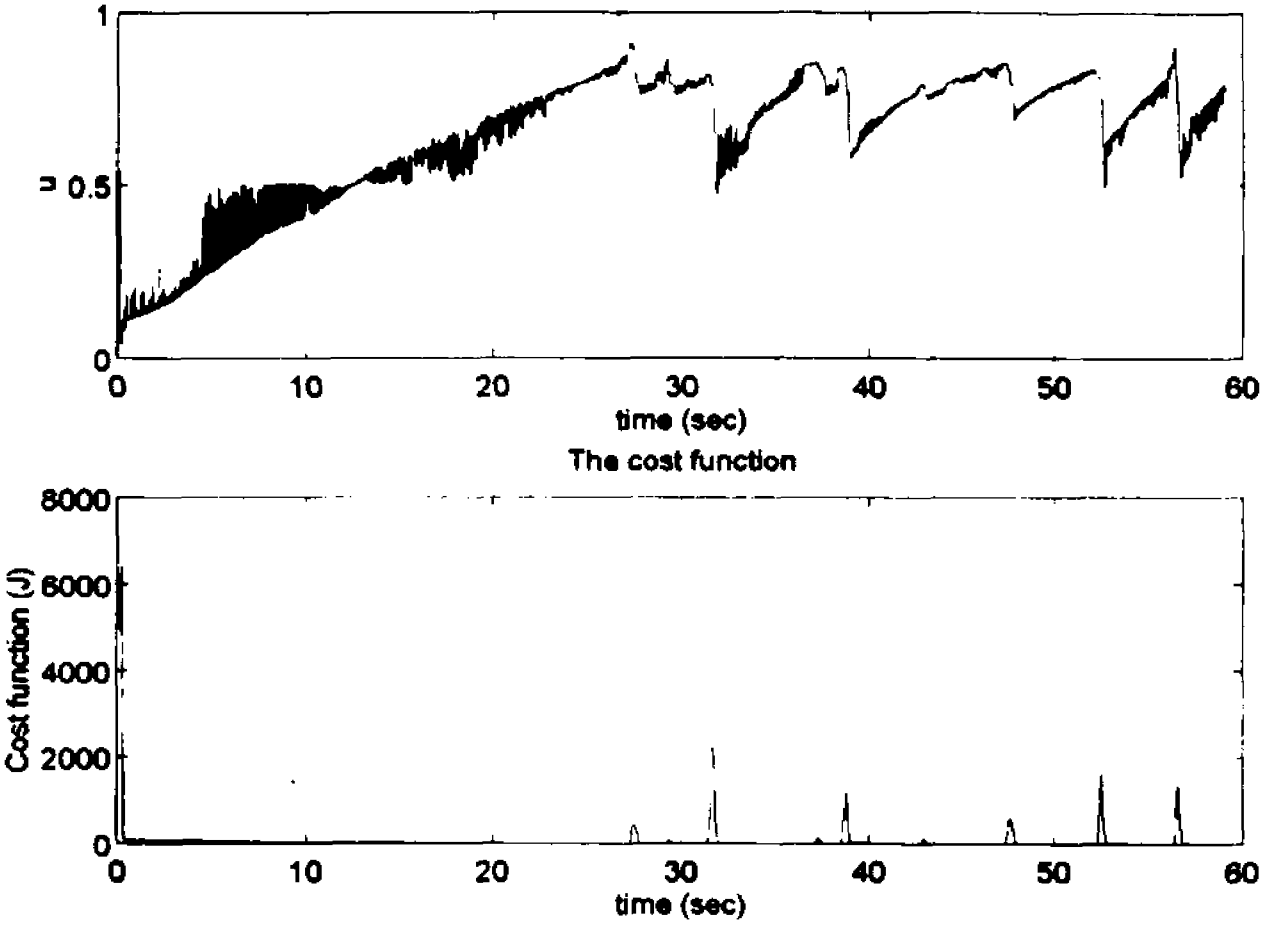


Fig. 4.8 Feedback control signal

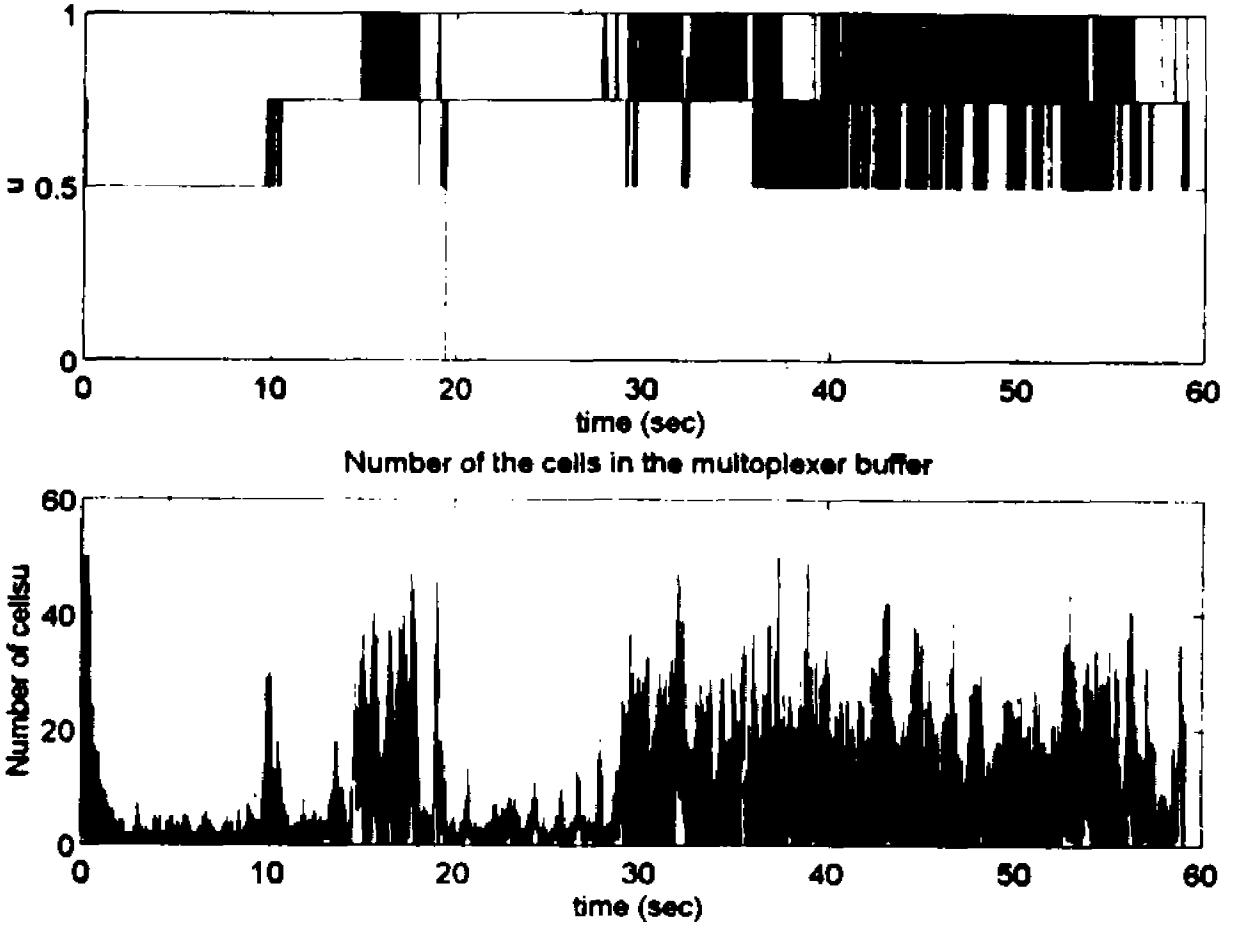


Fig. 4.9 Histogram of the number of voice cells in the buffer

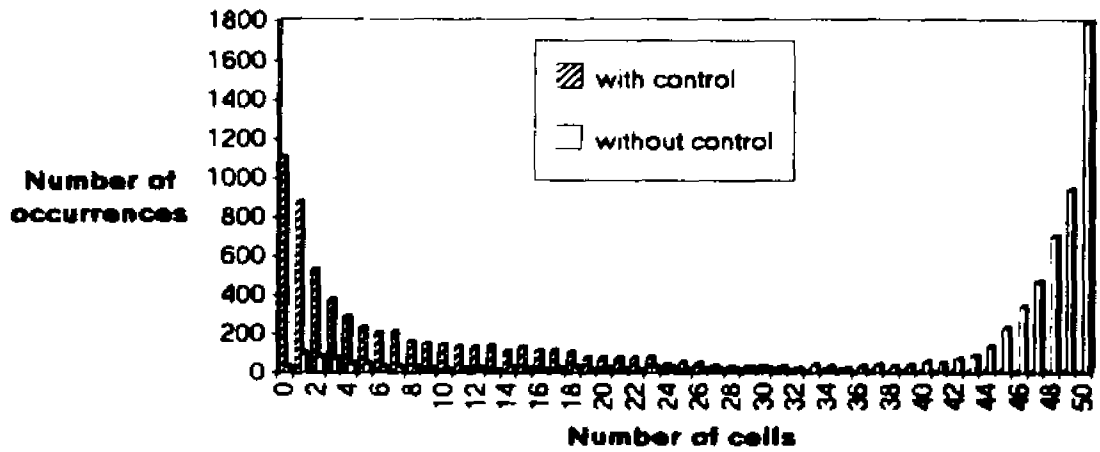


Fig. 4.10 Histogram of the feedback control signal

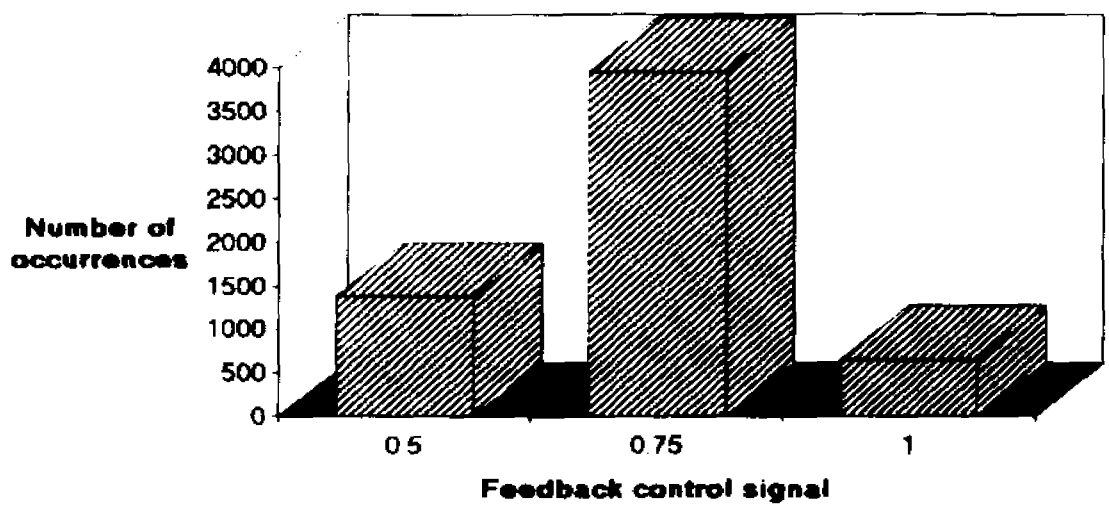


Fig. 4.11 The cost function

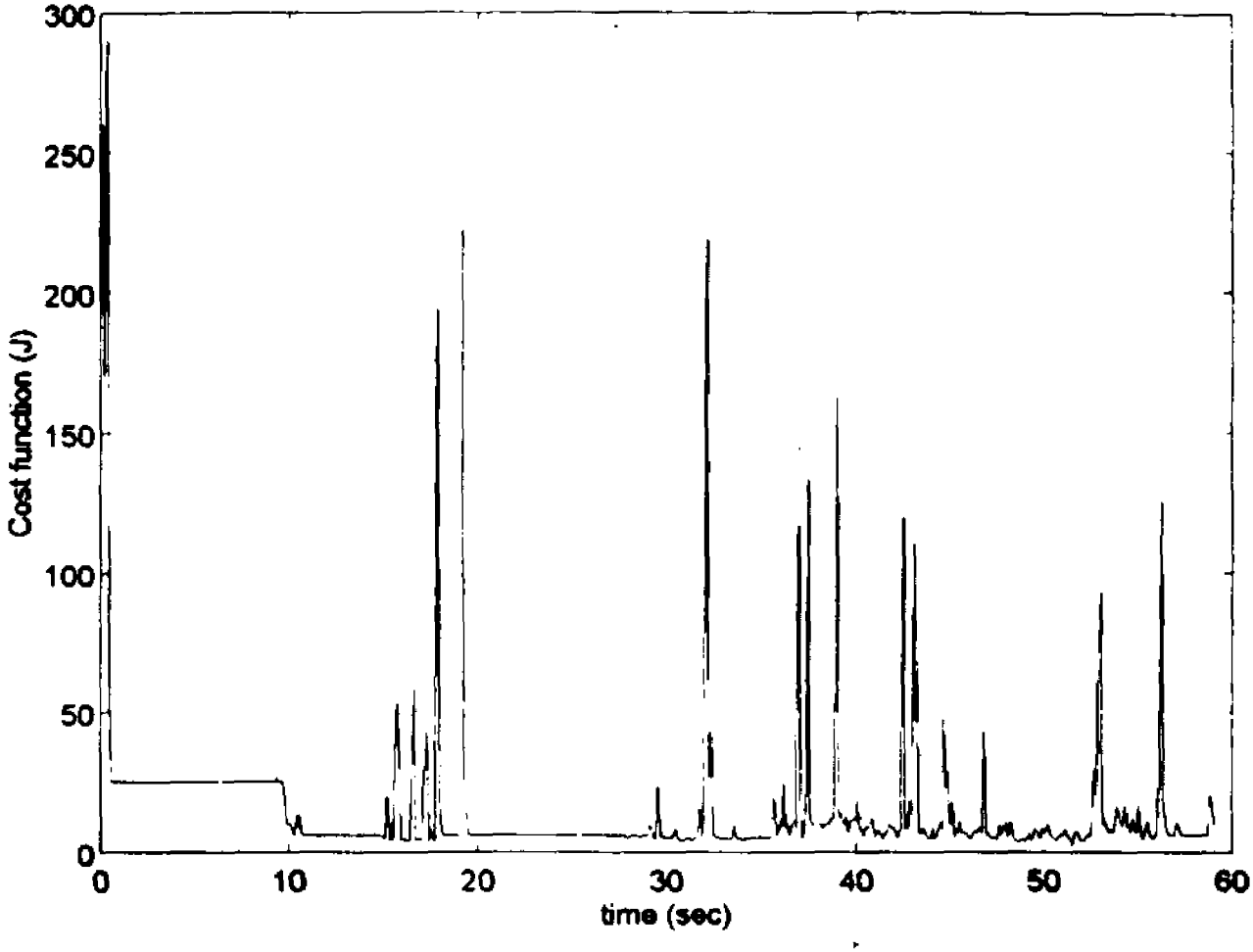


Fig. 4.12 Cell loss rate and voice quality

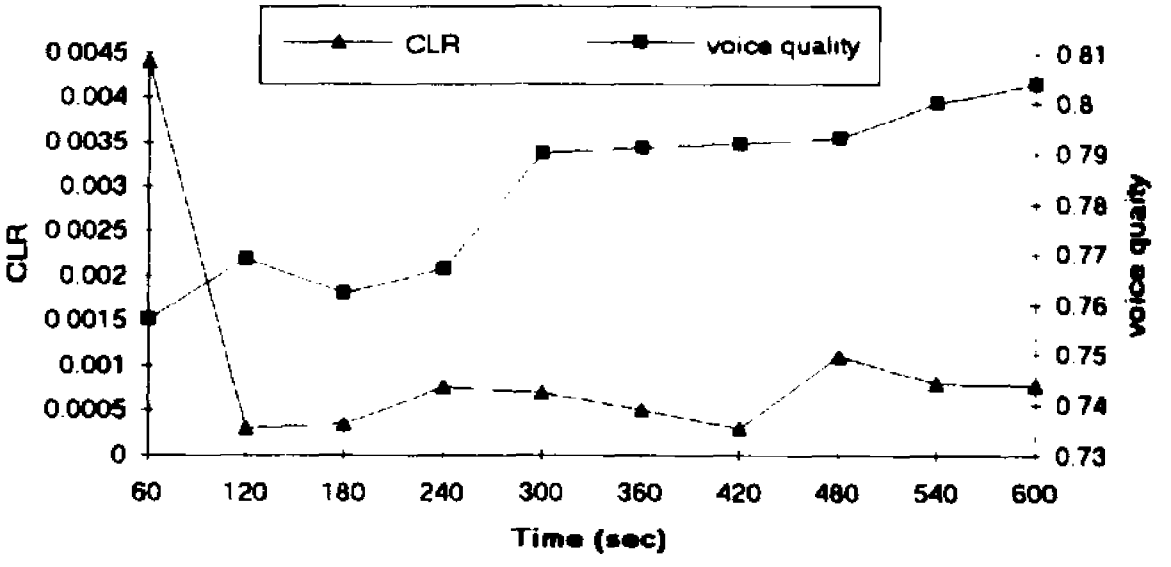


Fig. 4.13 Cell loss rate and voice quality versus Rn/Ru

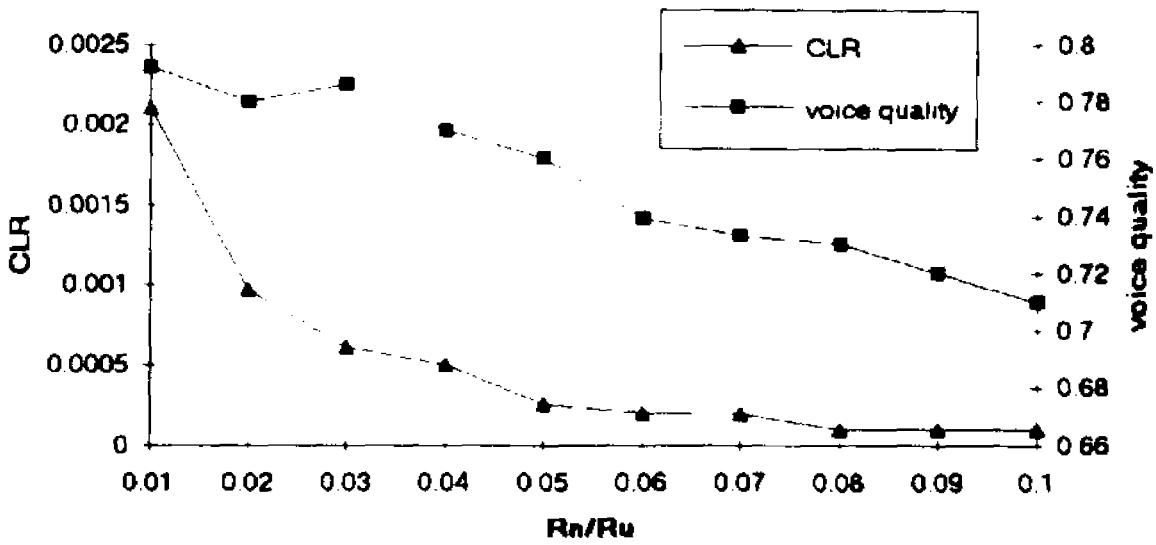


Fig. 4.14a Cell loss rate and voice quality versus Rn/Ru
L= 2Ts

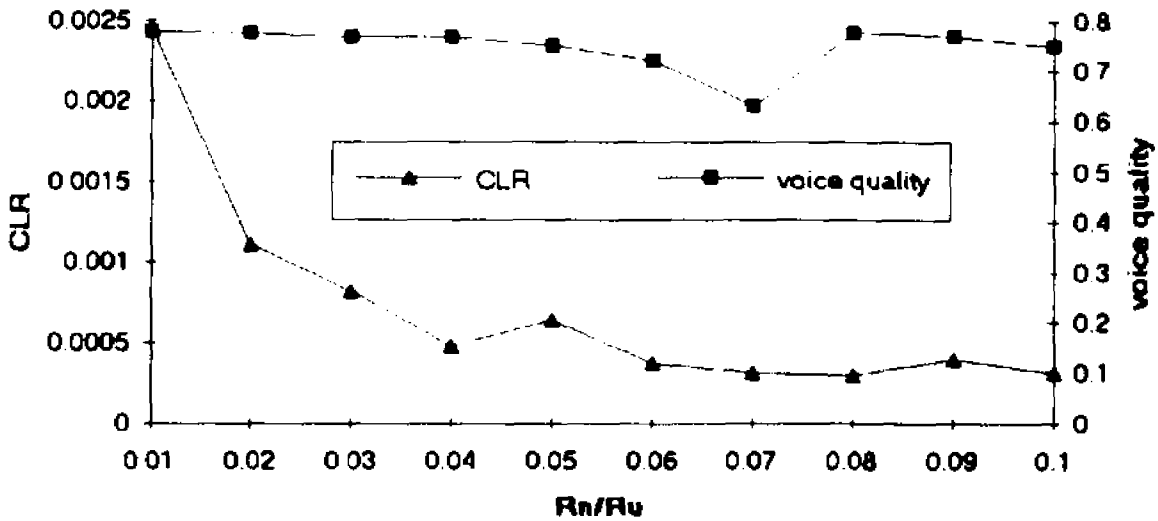


Fig. 4.14b Cell loss rate and voice quality versus Rn/Ru

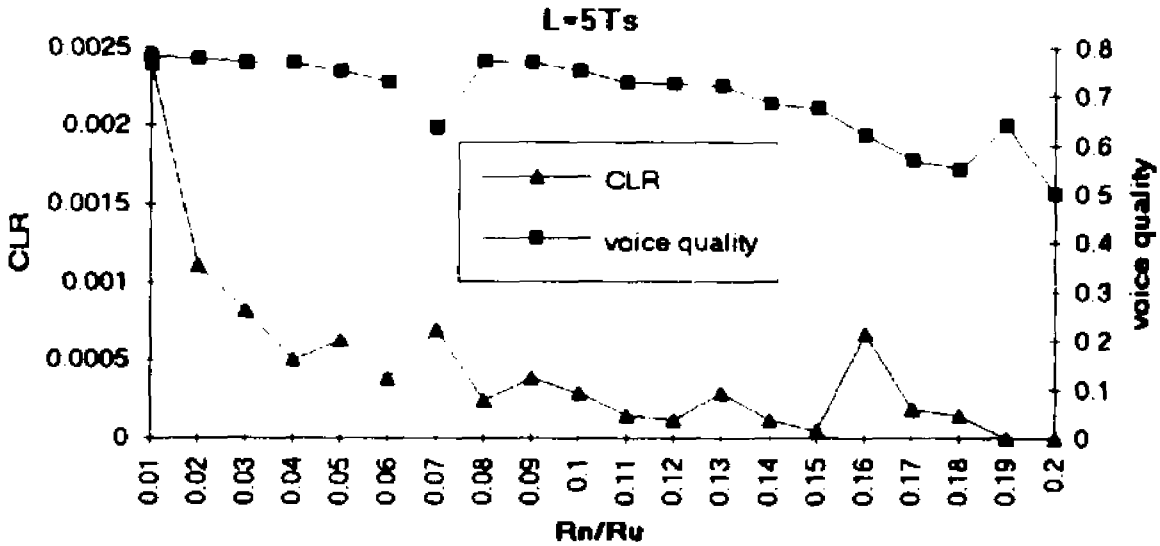


Fig. 4.14c Cell loss rate and voice quality versus Rn/Ru

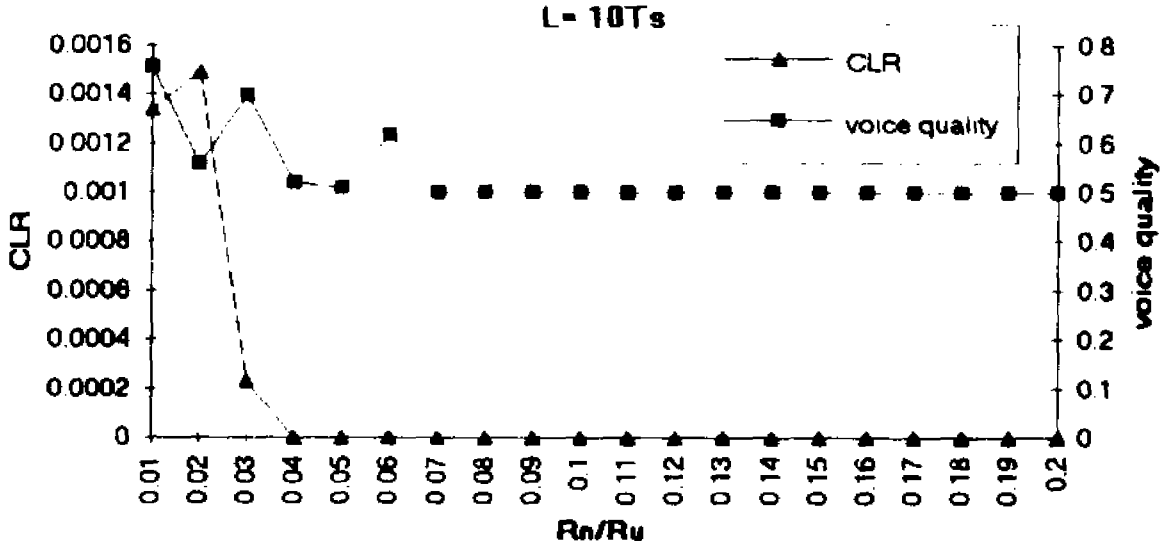


Fig. 4.14d Cell loss rate and voice quality versus Rn/Ru

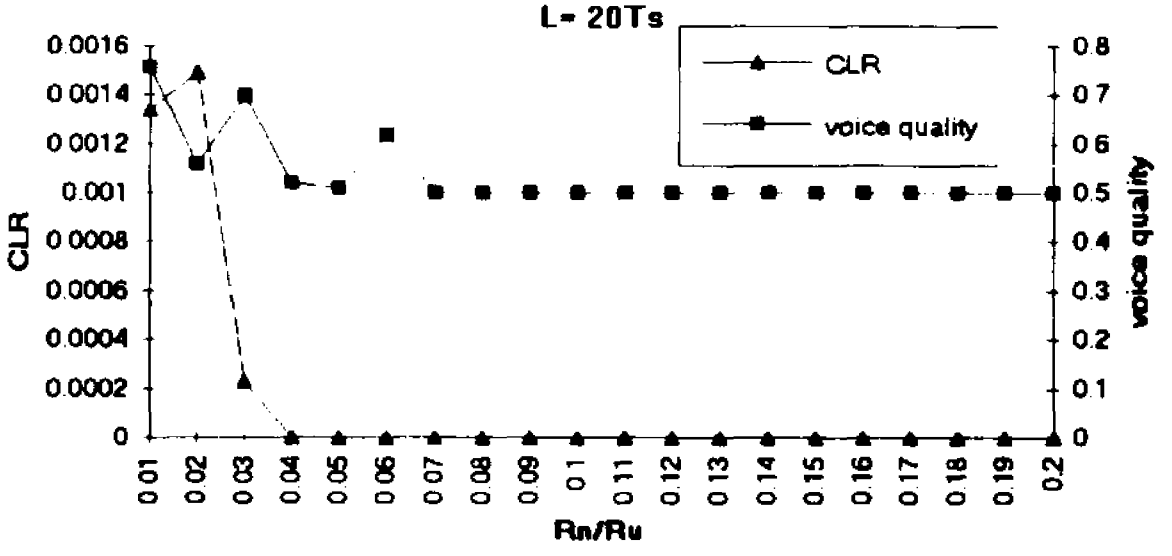


Fig. 4.15 CLR versus the utilization

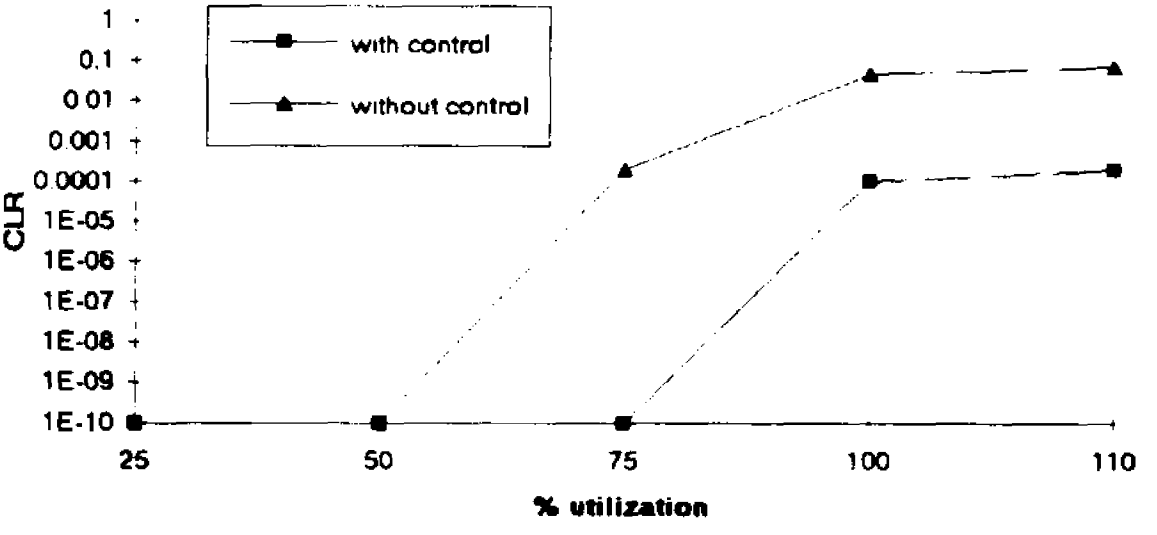


Fig. 4.16 CLR and voice quality versus the utilization

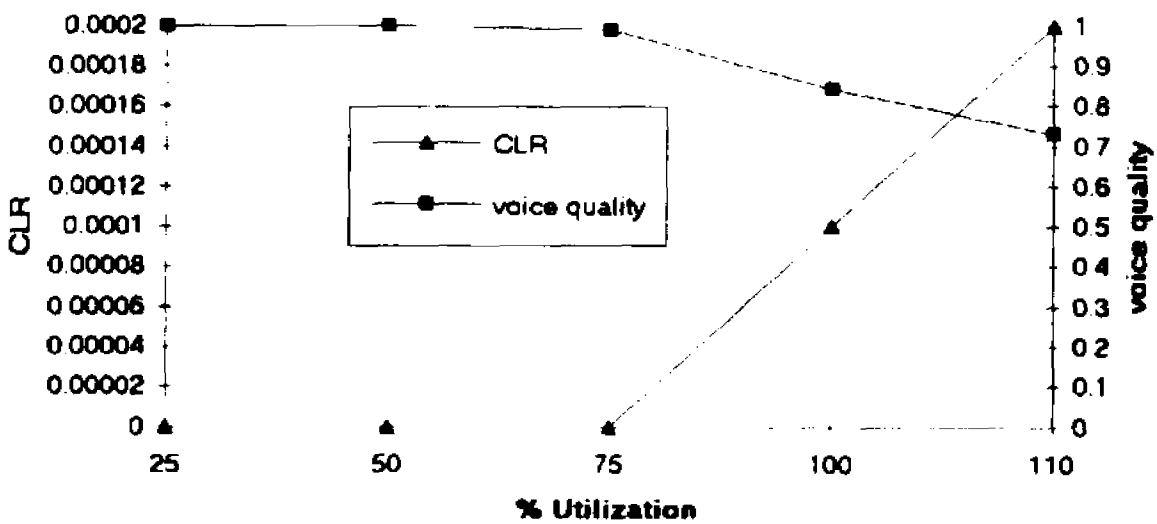


Fig. 4.17 Number of the video cells in the buffer

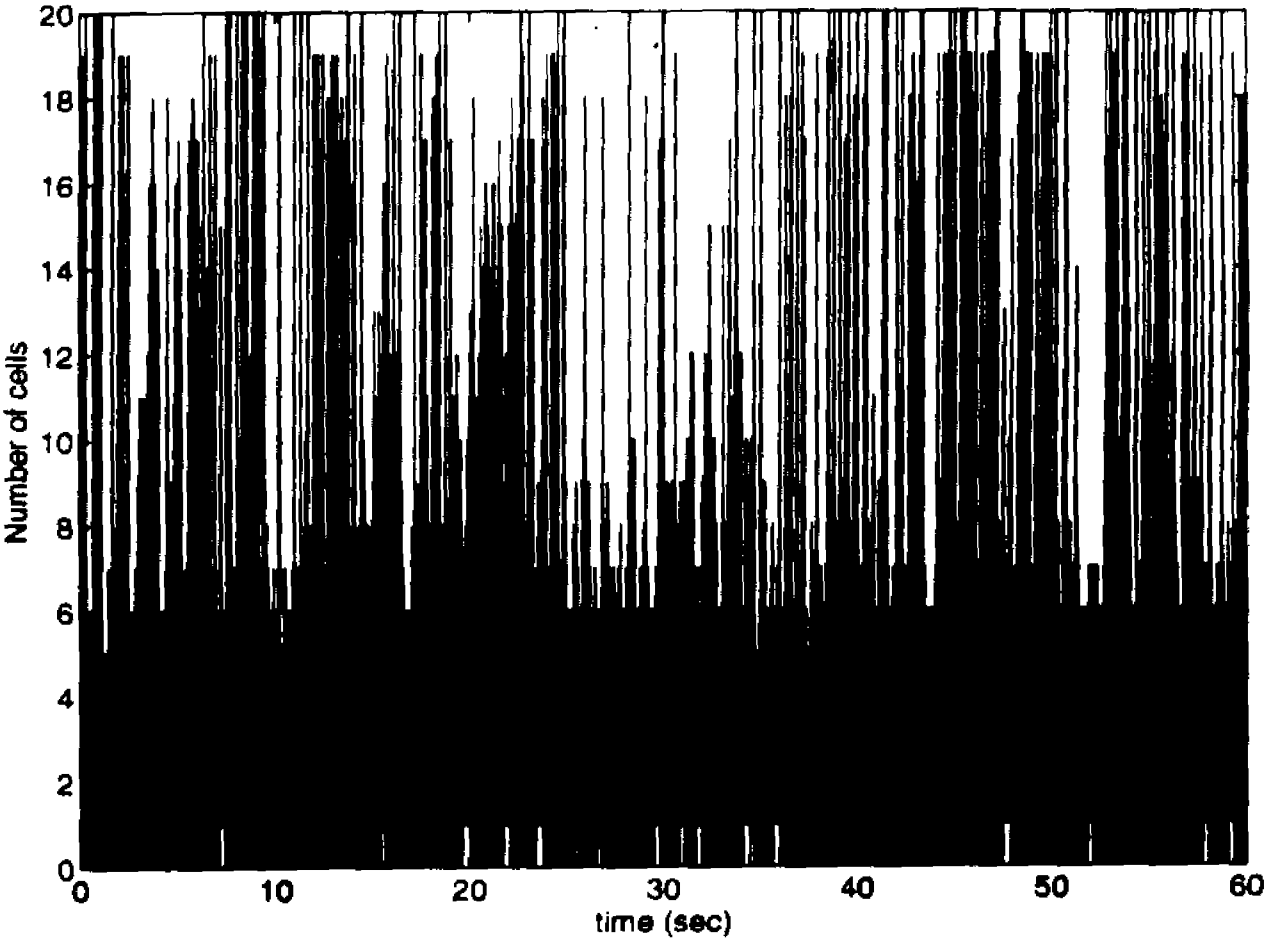


Fig. 4.18 Histogram of the number of the video cells in the buffer

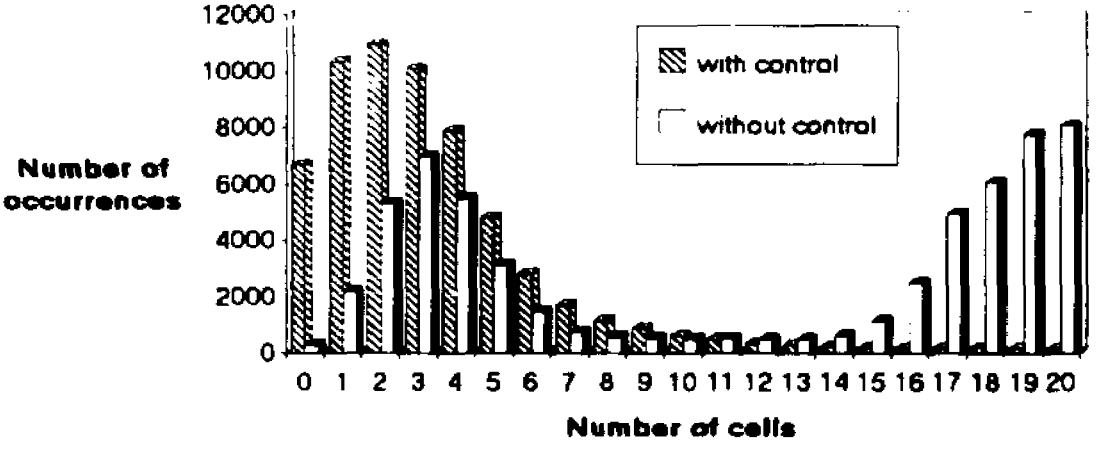


Fig. 4.19 Histogram of the feedback control signal

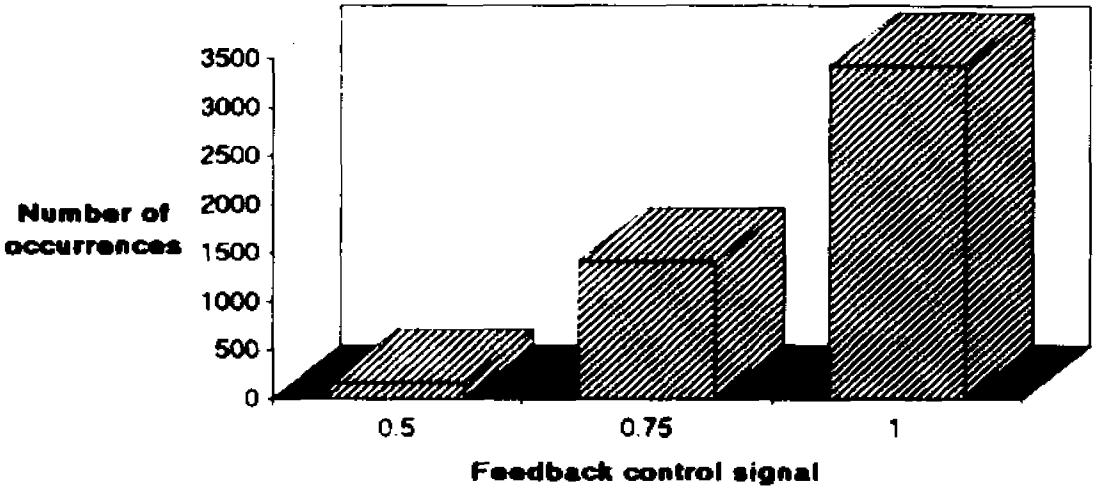


Fig. 4.20 Histogram of the number of the video cells in the buffer

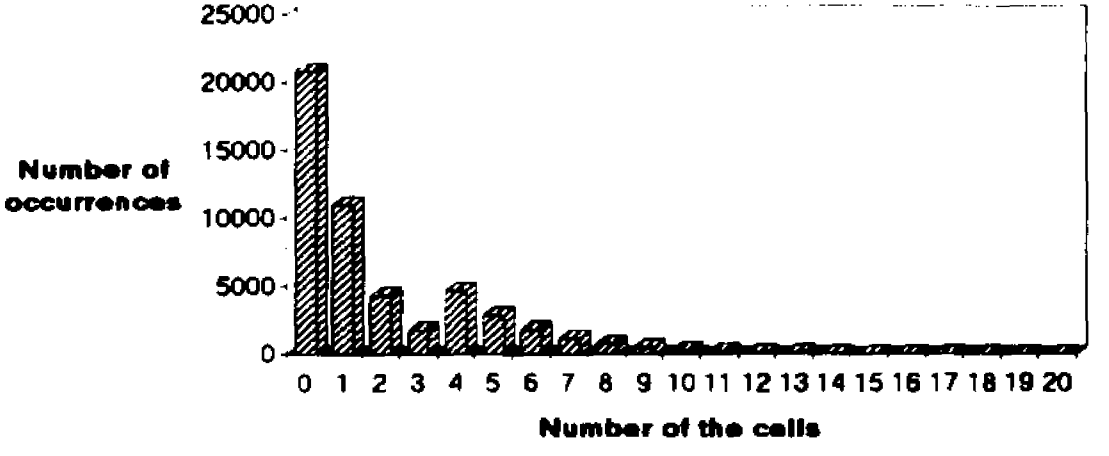


Fig. 4.21 Histogram of the feedback control signal

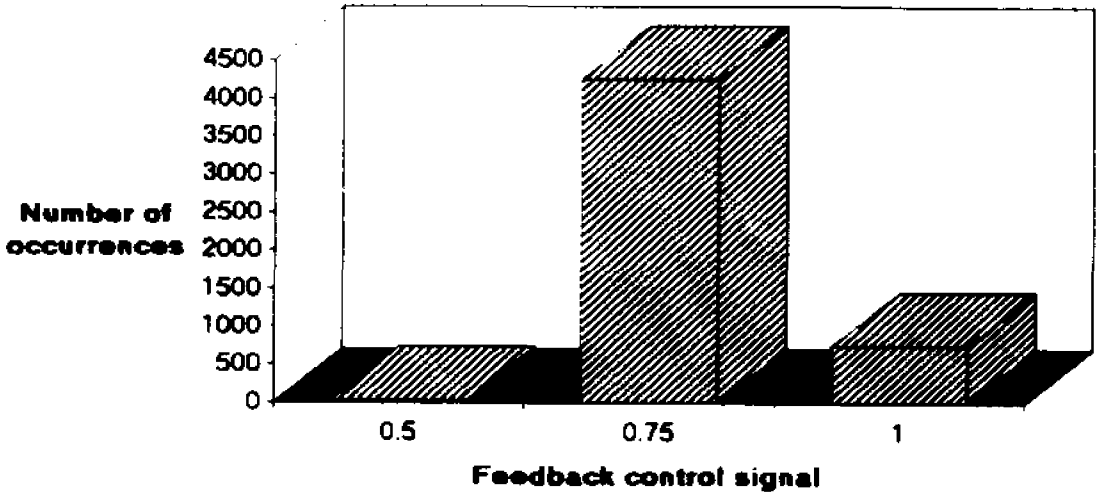


Fig. 4.22 CLR versus the utilization

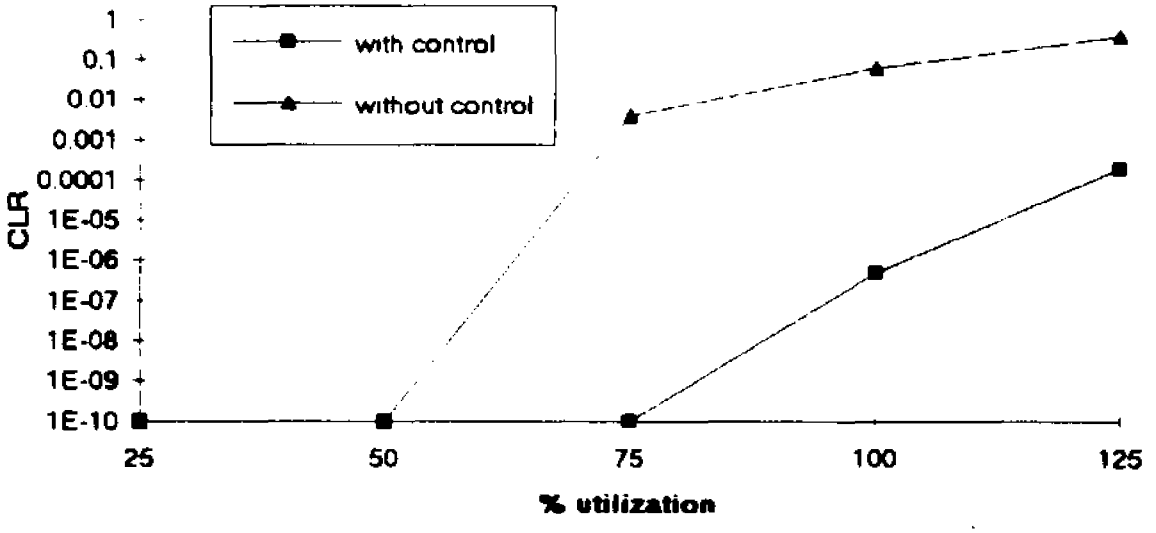
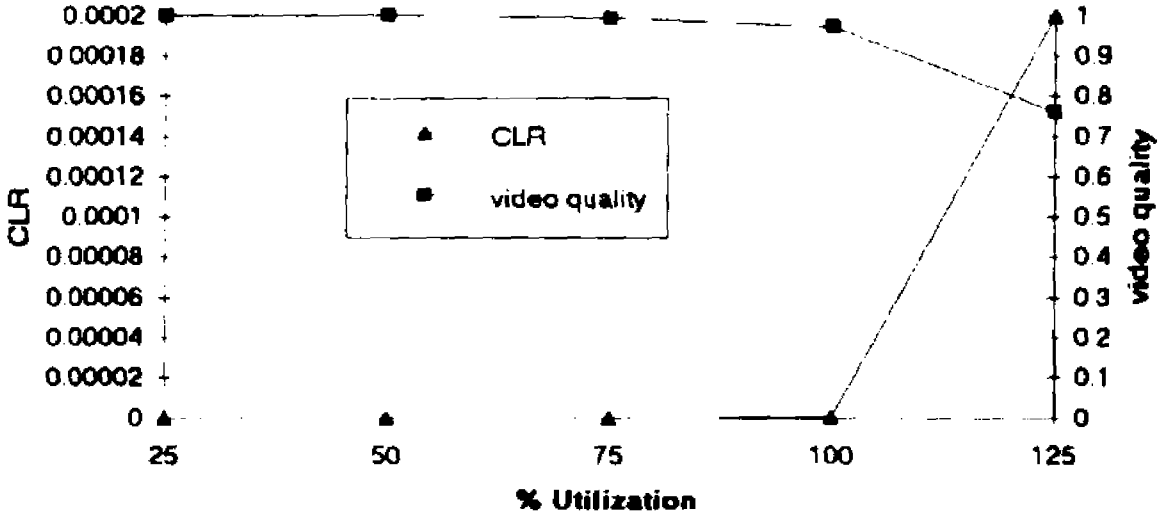


Fig. 4.23 CLR and video quality versus the utilization



V. Conclusions and Future work

In this research, we have introduced a scheme for ATM traffic management using NNs. The scheme is composed from cell level control functions, call level control function and network level control function. This research is mainly concentrated with the cell level control functions. The cell level control functions includes a traffic measurements function, traffic enforcement function and a feedback congestion control function, all implemented by NNs.

Based upon the obtained results for the traffic measurements using NNs, we conclude that a suitable NN architecture can be chosen to characterize the multimedia traffic. After completing the training phase of the NN, it will learn the pdf of the offered traffic. Hence, the NN can be used as an effective traffic descriptor. It describes the traffic by its actual pdf (instead of the approximated simple parameters such as the peak and mean bit-rates). The proposed model using the NNs is suitable for implementing an effective ATM traffic descriptor, since it can adaptively predict the traffic by learning the relationship between the past and the future traffic variations. The potential of the proposed model is demonstrated by its efficacious to predict the packet arrival process of: the superposition of M voice sources, the superposition of N video sources, and the superposition of N video sources and M voice sources. The results show that NNs can be trained to learn the pdf of the arrival process hence it can function as an adaptive predictor.

The traffic enforcement function of the cell level traffic management scheme includes a pdf-based NN traffic enforcement mechanism (NNTEM). In this mechanism,

a set of two inter-connected backpropagation NNs are used to characterize and predict any violations of the traffic pdf. The proposed NNTEM is suitable for the highly variable multimedia traffic enforcement in ATM-based networks. This is achieved by utilizing one NN to capture the actual pdf of the ideal "non-violating" traffic, while the other one is trained to adaptively characterize and predict any type of traffic violations by learning the relationship between the past and future traffic variations. The NNTEM polices the actual pdf of the traffic which includes all the statistical properties of the traffic. The error signal produced by the NNTEM can detect the individual contractual parameter violations as well as any combinations of the contractual parameter violations. This error signal is used to drop the violated cells.

As extension for the NNTEM, a Neural Network controller using reinforcement learning method is introduced. This mechanism is composite of critic and control functions, that are used to characterize, predict, and police any violations in the traffic pdf. The role of the critic function is to produce an evaluation signal which is function in the traffic violations. The critic function uses the NNTEM to produce an error signal function in the violations of the actual pdf of the traffic which includes all the statistical properties of the traffic. A reinforcement learning method defines a cost function in terms of traffic violations. The method uses the error signal to tune the weights of the NN in the controller part in order to achieve a control signal capable of optimally policing the traffic violations. The proposed system solves the fundamental problem challenging most of the existing flow enforcement mechanics which is the difficulty in policing both the mean and the peak bit-rates, simultaneously. The system can be easily configured and trained to incorporate any types of multimedia traffic by a simple modification of its input vector to include the learning of the pdf of the new traffic types. Although the

training phase requires extensive numbers of trials, it does not affect the performance since it is done "off-line" and before the actual "on-line" or production phase of the NN.

The feedback control function involves an adaptive congestion control model using NNs. The presented algorithm is a preventive type and is applied at the access node of the network. The algorithm is a rate-based optimal control implemented by the reinforcement learning NN. The algorithm uses a feedback control signal to throttle the peak bit-rate of the input arrival process to the input statistical multiplexer. The feedback control signal is applied to the input source coder. This signal will control the source rate by decreasing the coding rate (number of bits per sample). The feedback control signal is produced by a NN-based controller. This feedback control signal is an optimal control signal in the sense that it is determined in such way to maximize the system performance. The system performance is measured by a certain performance index which combines two important system performance measures:

1) The input multiplexer buffer overflow (this will reflect the CLR for the accepted calls)

2) The level of the coding rate of the input source(s)

We have solved the optimal control problem using the reinforcement leaning NN method. In this method, the weights of a NN-based controller are tuned continuously in order to maximize the system performance. Since the proposed algorithm is a rate-based optimal control implemented by the reinforcement learning NN. Therefore, it has the advantages of the rate-based control method, the optimal control method, and the reinforcement learning method. The obtained results of that algorithm have assured its potential and effectiveness in maintaining a low CLR in the mean time maintaining a high voice and video quality.

Although the proposed traffic management scheme shows excellent results with the simulated system (i.e., input voice and video sources and input multiplexer buffer), it is expected that the proposed scheme works successfully with the real time system (as indicated by experiment (2.10)). Hence, we suggest to build a real time test-bed system which should be composed from variable bit-rate voice and video sources, dropping switches and input multiplexer buffer. The proposed NN cell level traffic management scheme can be implemented using hardware or software to generate the cell dropping signal (s) that control the operation of the dropping switch (s) and the feedback control signal that changes the coding rate of the input voice and video sources.

We also suggest to implement a call level traffic management scheme, and a network level management scheme using NNs and combine these two schemes with our cell level traffic management scheme to obtain the total NN traffic management solution illustrated in figure (1.2).

Appendix I

This appendix explains the detailed derivation of (∂J(P)/∂W(P)) and (∂J(P)/∂w(P)) used in equations (3.5) and (3.6) respectively. We start with (∂J(P)/∂W(P)):

From Equation (3.1), with Q=0.5, (∂J(P)/∂W(P)) is expressed as follows:

$$\frac{\partial J(P)}{\partial W(P)} = -\sum_{k=1}^{L} (h_d(k) - h(k)) \frac{\partial h(k)}{\partial W(P)} = -\sum_{k=1}^{L} \epsilon(k) \frac{\partial h(k)}{\partial u(k)} \frac{\partial u(k)}{\partial W(P)} \quad (A1)$$

$$\frac{\partial J(P)}{\partial W(P)} = -\sum_{k=1}^{L} \epsilon(k) \frac{\partial u(K)}{\partial W(P)} h_i(K) \quad (A2)$$

Define x1 and x2 as follows:

$$x1 = w^T(P) f(W(P) I(k)) = w^T(P) f(x2) \quad (A3)$$

$$x2 = W(P) I(K) \quad (A4)$$

Then, from equation (3.2), u(k) can be defined as :

$$u(K) = f(x1) \quad (A5)$$

From equation (A5), (∂u(k)/∂W(P)) can be expressed as:

$$\frac{\partial u(K)}{\partial W(P)} = \frac{\partial f(x1)}{\partial x1} \frac{\partial x1}{\partial W(P)} \quad (A6)$$

Where (∂x1/∂W(P)) is defined as:

$$\frac{\partial x1}{\partial W(P)} = w^T(p) \frac{\partial f(x2)}{\partial x2} \frac{\partial x2}{\partial W(P)} \quad (A7)$$

It Follows that:

$$\frac{\partial x1}{\partial W(P)} = w^T(P)\,\frac{\partial f(x2)}{\partial x2}\,I(K) \qquad (A8)$$

Hence from (A6) and (A8), we can get:

$$\frac{\partial u(k)}{\partial W(P)} = \frac{\partial f(x1)}{\partial x1}\,w^T(P)\,\frac{\partial f(x2)}{\partial x2}\,I(k) \qquad (A9)$$

Therefore

$$W(P+1) = W(P) + \eta \sum_{k=1}^{L} \epsilon(k)\,h_i(k)\,\frac{\partial f(x1)}{\partial x1}\,w^T(P)\,\frac{\partial f(x2)}{\partial x2}\,I(k) \qquad (A10)$$

Similarly $(\partial J(P)/\partial w(P))$ can be expressed as:

$$w(P+1) = w(P) + \eta \sum_{k=1}^{L} \epsilon(k)\,h_i(k)\,\frac{\partial f(x1)}{\partial x1}\,f(x2) \qquad (A11)$$

Appendix II

This appendix explains the detailed derivation of (∂J(P)/∂W(P)) and (∂J(P)/∂w(P)) used in equations (4.9) and (4.10) respectively. We start with (∂J(P)/∂W(P)):

From equation (4.5) (∂J(P)/∂W(P)) is expressed as follows:

$$\frac{\partial J(p)}{\partial W(p)} = -\sum_{k=1}^{L} 2R_n S_n(k+1)\epsilon_n(k+1)\frac{\partial n(k+1)}{\partial W(p)} + 2R_u\epsilon_u(k+1)\frac{\partial u(k+1)}{\partial W(p)} \quad \text{(A12)}$$

$$\frac{\partial J(p)}{\partial W(p)} = -\sum_{k=1}^{L} 2R_n S_n(k+1)\epsilon_n(k+1)\frac{\partial n(k+1)}{\partial u(k)}\frac{\partial u(k)}{\partial W(p)} + 2R_u\epsilon_u(k+1)\frac{\partial u(k+1)}{\partial W(p)} \quad \text{(A13)}$$

$$\frac{\partial J(p)}{\partial W(p)} = -\sum_{k=1}^{L} 2R_n S_n(k+1)\epsilon_n(k+1)h^*(k)\frac{\partial u(k)}{\partial W(p)} + 2R_u\epsilon_u(k+1)\frac{\partial u(k+1)}{\partial W(p)} \quad \text{(A14)}$$

where $h^*(k)$ is the amount of the uncontrolled traffic (measured in number of cells) emitted from the traffic sources during one sample period (from k to k+1).

Define x1(k) and x2(k) as follows:

$$x1(k) = w^T(P)\, f(W(P)\, I(k)) = w^T(P)\, f(x2(k)) \quad \text{(A15)}$$

$$x2(k) = W(P)\, I(K) \quad \text{(A16)}$$

Then similarly as we did in appendix I, W(P+1) can be expressed as:

$$W(P+1) = W(P) + \eta \sum_{k=1}^{L} \{2R_n S_n(k+1)\epsilon_n(k+1)h^*(k) \frac{\partial f(x1(k))}{\partial x1(k)} w^T(P) \frac{\partial f(x2(k))}{\partial x2(k)} I(k) + 2R_u \epsilon_u(k+1) \frac{\partial f(x1(k+1))}{\partial x1(k+1)} w^T(P) \frac{\partial f(x2(k+1))}{\partial x2(k+1)} I(k+1)\}$$

Similarly $(\partial J(P)/\partial w(P))$ can be expressed as:

$$w(P+1) = w(P) + \eta \sum_{k=1}^{L} \{2R_n S_n(k+1)\epsilon_n(k+1)h^*(k) \frac{\partial f(x1(k)}{\partial x1(k)} f(x2(k)) + 2R_u \epsilon_u(k+1) \frac{\partial f(x(k+1))}{\partial x1(k+1)} f(x2(k+1))\} \quad \text{(A18)}$$

Bibliography

[1] CCITT Recommendations, I series (B-ISDN), July 1992.

[2] Special section on neural networks, IEEE control sys. Mag., Apr. 1988.

[3] Bronoko soucek "Neural and concurrent real-time systems," the six generation, John Wiley & sons, Inc., 1989.

[4] A. Hiramatsu, "Integration of ATM call admission control and link capacity control by distributed neural networks," IEEE JSAC, Vol. 9, No. 7, Sept. 1991.

[5] T. Takahashi and Hiramatsu, " Integrated ATM traffic control by distributed neural networks.", in Proc. ISS'90, Stockholm, Sweden, May 1990, vol. III, PP. 54-65.

[6] A. Hiramatsu, "ATM communications network control by neural networks," IEEE Trans. neural networks, vol 1, PP. 122-130, 1990.

[7] B. Aazhang et al "Neural networks for multiuser detection in code-division multiple access communications," IEEE Trans. Comm, Vol. 40, NO. 7, July 1992.

[8] R. Lippmann "An introduction to computing with neural nets," IEEE Assp Mag., April 1987.

[9] R. Nielsen "Neurocomputing," Addison wesly publishing comp., 1989.

[10] D. E. Rumelhart, J. L. McClellend, and PDP Research Group, "Parallel Distributed Processing," Vol I. Cambridge, MA: M.I.T. Press, 1986.

[11] ATM User-Network Interface Specification Version 2.0, June 1992.

[12] K. Siriram, and W. Whitt "Characterizing superposition arrival processes in packet multiplexers for voice and data,"IEEE JSAC, Vol. SAC-4. NO. 6, Sept 1986.

[13] H. Heffes, and D. Lucantoni, "A markov modulated characterization packetized voice and data traffic and related statistical multiplexer performance," IEEE JSAC, Vol. SAC-4, NO. 6, Sept.1986.

[14] H. Yamada, and S. Sumita "A traffic measurement method and its application for cell loss probability estimation in ATM networks," IEEE JSAC, Vol 9, NO. 3, April 1991.

[15] I. Habib, and T. Saadawi "Multimedia Traffic characteristics in broadband networks," IEEE Comm. Mag. July 1992.

[16] K. Siriram, and D. Lucantoni "Traffic smoothing effects of bit dropping in a packet voice multiplexer," IEEE Trans. Comm, Vol. 37, No. 7, July 1989.

[17] H. Satio, and H. Yamada "Analysis of statistical multiplexing in ATM transport network," IEEE JSAC Vol. 9, NO. 3, April 1991.

[18] J Diagle, and J Langford "Models for analysis packet voice communications systems," IEEE JSAC, Vol. SAC-4, NO. 6, Sept. 1986.

[19] A. Baiocchi et al "Loss performance analysis of an ATM multiplexer loaded with high-speed on-off sources," IEEE JSAC, Vol. 9, NO. 3, April 1991.

[20] E. Rathgeb "Modeling and performance comparison of policing mechanisms for ATM networks," IEEE JSAC, Vol. 9, No. 3, April 1991.

[21] F. Denissen, E. Desmet, and G. H. Petit "The policing function in an ATM network," in Proc. 1990 Int. Zurich Sem. Digit. Commun., Zurich, MAR. 1990, PP. 131-144.

[22] A. Tarraf, I. Habib, and T. Saadawi "Characterization of packetized voice traffic in ATM using neural networks," IEEE Globcom'93.

[23] A. Tarraf, I. Habib, and T. Saadawi "Neural network model for packetized voice traffic in ATM networks," HPCS'93, Sept. 1993

[24] Ernst Nordstrom "A Hybrid Admission Control Scheme for Braodband ATM Traffic." Proceeding of the International Workshop on Apllication of Neural Network to Telecommunication, Princeton 1993.

[25] L. Dittmann, S. Jacohsen, and K. Moth "Flow enforcement algorithms for ATM networks," IEEE JSAC, Vol. 9, NO. 3, April 1991.

[26] I. Habib and T. Saadawi "Access flow control algorithm for voice-multiplexers in ATM networks," Proceedings of IEEE Milcom'92.

[27] M. Butto', E. Cavallero, and A. Tonietti "Effectiveness of the leaky bucket policing mechanism in ATM networks," IEEE JSAC, Vol. 9, NO. 3, April 1991.

[28] H. Chao "Design of leaky bucket access control schemes in ATM networks," ICC 1991.

[29] H. Chao "Architecture design for regulating and scheduling user's traffic in ATM networks," ICC 1991.

[30] I. Habib and T. Saadawi "Congestion control in video multiplexers in broadband networks," Journal of computer comm. June 1992, also in part in the proceedings of IFIP third conference on High Speed Networks, Berlin, March 91.

[31] T. Aoyama, I. Tokizawa, and K. Sato "ATM VP-based broadband network for multimedia services," IEEE comm. Mag., April 1993.

[32] S. Yazid and H.T. Mouftah "Congestion control methods for BISDN," IEEE Comm. Mag. July 1992.

[33] I. Habib and T. Saadawi "Controlling flow and avoiding congestion in broadband ATM networks," IEEE Comm. Mag., October 1991.

[34] M. De Prycker "Asynchronous transfer mode," Ellis horwood Ltd. 1991.

[35] A. Tarraf, I. Habib, and T. Saadawi "A novel neural network traffic enforcement mechanism for ATM networks," IEEE JSAC, special issue on Network Management.

[36] A. Tarraf, I. Habib, and T. Saadawi "A novel neural network controller using reinforcement learning method for ATM traffic policing," Submitted to IEEE JSAC, special on intelligent signal processing in communications.

[37] Ogata "Introduction to digital control systems," Addison-Wesly publishing comp., 1988.

[38] J. Proakis, and G. Manolakis "Introduction to digital signal processing," Macmillan publishing comp., 1990.

[39] J. Little and L. Shure "Signal Processing Toolbox, for use with MATLAB," The MathWork, Inc, August 1988.

[40] HNC Inc. "HNC ExploreNET release 2.0 operating manual," 1991.

[41] R. Grunenfelder et al "Characterization of video codecs as autoregressive moving average processes and related queuing system performance," IEEE JSAC, Vol 9, No. 3, april 1991.

[42] M. Nomura, T. Fujii, and N. Ohta "Basic characteristics of variable rate video coding in ATM environment," IEEE JSAC, Vol. 7, No. 5, June 1989.

[43] H. Soon et al "Statistics od video signals for viewphone-type pictures," IEEE JSAC, Vol 7, No. 5, June 1989.

[44] B. Maglaris et al "Performance models of statistical multiplexing in packet video communications," IEEE Trans. Comm, Vol. 36, No. 7, July 1988.

[45] P. Sen et al "Models for packet switching of variable-bit-rate video sources," IEEE JSAC, Vol. 7, No. 5, June 1989.

[46] D. Le Gall "MPEG: A Video Compression standard for multimedia applications Communications of the ACM," 34(4):305-313, April 1991.

[47] T. Urabe rt el "MPEGTool: An X Window Based MPEG Encoder and Statistics Tool," Univ. of Pennssylvania.

[48] J. Hertz et el "Introduction to the theory of neural computation," Addison-Wesly publishing comp., 1991.

[49] W. Thomas et el "Neural Networks for control," The MIT Press, Cambridge, MA, 1990.

[50] T. Yamada and T. Yabuta "Neural network controller using autotuning method for nonlinear function," IEEE Trans. on Neural Networks, Vol. 3, No. 4, July 1992.

[51] K. Narendra and M. Thathachar "Learning automata," Prentice Hall, Englewood Cliffs, NJ, 1989.

Biography

Ahmed A. Tarraf received the B.Sc. and M.Sc. degrees in electrical engineering from Ain Shams university, Cairo, Egypt, in 1986 and 1990 respectively.

From 1986 to 1991, he was a research/teaching assistant in the department of electronics and computer engineering, Ain Shams University, Cairo, Egypt. During the same period, he was a senior design engineer working in the area of Data Acquisition and Control Systems. His research interests are in the applications of Artificial Neural Networks in the area of Access Flow Control and Congestion Avoidance in Broadband Networks.