NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

APPLICATION OF NEURAL NETWORKS TO ISDN OPTIMAL ACCESS CONTROL POLICIES

Luc Vouligny

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

February 1991

ISBN 0-315-64725-6

Canada

ABSTRACT

Application of Neural Networks to ISDN Optimal Access Control Policies

Luc Vouligny, February 1991

The problem of determining optimal access policies for integrated circuit-switched and packet-switched traffic types for communication networks is addressed. The network is assumed to support K classes of calls where each class is determined by a fixed route and a bandwidth requirement in the case of circuit-switched traffic or a packet arrival rate in the case of packet-switched traffic. A certain grade of service is allowed in every class of calls. The optimal access policy determines the decision to accept or reject a call based on maximizing the utilization of a link. A Semi-Markov Decision Process (SMDP) approach is employed to extend the existing results for circuit-switched traffic to an integrated environment which includes packet-switched traffic also. Then the SMDP is mapped as a Linear Programming algorithm. Finally, the optimization process is solved using the parallel computing power found in Neural Networks. Simulation results demonstrate the superior utilization of a link when packet-switched traffic transmission is allowed compared to a link where only circuit-switched traffic is permitted.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude to my supervisor Dr. M.K. Mehmet Ali for his precious help, support, and guidance during the entire development and preparation of this thesis.

I also gratefully acknowledge the support of Dr. J.F. Hayes and Dr. M.K. Mehmet Ali for the opportunity to work on their research project which provided much support for the completion of the master's program.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_k	Action to be taken on a class-k call
$\alpha(t)$	Number of arrivals in time interval $(0,t)$
$\alpha_{xa}^{(k)}$	Frequency constraint corresponding to the class-k calls specification when in state x given that action a is chosen
$\vec{A}$	Vector of objective function coefficients
b_k	Bandwidth requirement of the class-k calls in number of channels
β	Constraint specified by the class-k calls
B	Action space
B_x	State dependent action space
C	Capacitance value
C_n	Number of circuit-switched classes of calls supported by link n
δ	Incremental time interval
$\delta(t)$	Number of departures in time interval $(0,t)$
∂	Partial derivative
Δt	Short time interval
D	Diode value
D_{ij}	Weight corresponding to the i^{th} restriction amplifier and to the j^{th} variable amplifier
e_k	Vector of zeroes, except for the k^{th} component
$E[\ ]$	Expected value function
$\in$	Belongs to
$\notin$	Does not belong to

η_k	Mean packet arrival rate for class-k calls in packets per second per call
Σ	Summation symbol
$f, \mathbf{f}$	Function associated with the variable amplifiers in the neural network
$f_{xa}(\pi)$	The long-run fraction of decision epochs at which the process is in state x and action a is chosen when policy π is used
Ψ	Output of the restriction amplifiers used in the neural network implementation
$g, \mathbf{g}$	Function associated with the inequality restrictions amplifiers in the neural network
G	Function used to ensure that inequality restrictions will be respected
$h, \mathbf{h}$	Function associated with the equality restrictions amplifiers in the neural network
H	Function used to ensure that equality restrictions will be respected
J_0	Set of states where an action is possible
K	Total number of classes of calls
λ_k	Arrival rate of class-k calls in calls per second
Λ	Set of all possible network states
m_n	Capacity of link-n in number of channels
m_{xc}	Number of channels occupied by circuit-switched calls
m_{xp}	Number of channels occupied by packet-switched calls
$1/\mu_k$	Expected service duration taken by a class-k call in seconds
M	Momentum value of a neural network
$N(t)$	Number of packets in the system at time t
ω	Mean service rate of packets per channel per second
ρ_{xc}	Utilization of a circuit-switched channel in state x
ρ_{xp}	Utilization of a packet-switched channel in state x

π	A network access control policy
$\pi_a(x)$	Probability to be in state x and to choose action a
P_n	Number of packet-switched classes of calls supported by link n
P_j	Probability of having j packets in the system
$Pr\{\ \}$	Probability function
$\tilde{P}_{xay}$	Transition probability from state x to state y given action a chosen (for the Markovian Decision Process case)
P_{xay}	Transition probability from state x to state y given action a chosen (for the Semi-Markovian Decision Process case)
$r(x)$	Overall utilization of a link in state x
$\mathbf{R}, \mathbf{r}_j$	Resistance value
$\overline{T}_i$	Vector normal to the i^{th} hyperplane
$\tau(x,a)$	Expected time until a new state is entered from state x given action a is chosen
$\mathbf{U}$	Potential value
$\mathbf{V}_j$	Potential value at the output of the variable amplifiers
$\overrightarrow{V}$	Vector of objective function variable
x	Element of the state space
x_k	Number of class-k calls connected in state x
γ	Scaling of the penalizing function
z_{xa}	Probability to be in state x and choosing action a

CHAPTER 1

INTRODUCTION

With the advent of todays'wideband transmission facilities such as optical fibers, demands are created for integrated multimedia communication networks like the Broadband Integrated Services Digital Network or B-ISDN. As stated in the International Consultative Committee for Telephone and Telegraph (CCITT) recommendations, B-ISDN is an all purpose digital network supporting a wide range of audio, video, and data applications in the same network. The network capabilities include support for variable-bit-rate information transfer from Kbps to Gbps for both bursty and continuous traffics, as well as for dialog and broadcast services. A conceptual view of the B-ISDN multimedia communication features is depicted in Figure 1.1.

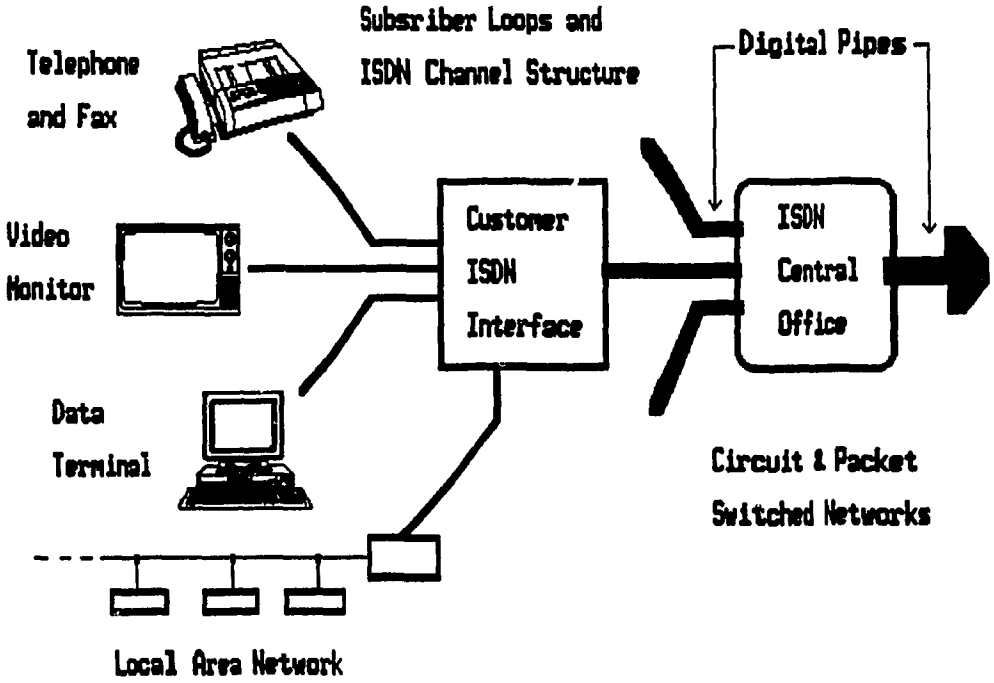

Figure 1.1 Conceptual View of the B-ISDN Connection Features.

A given user will have all its ISDN communicating devices attached to a single Customer ISDN Interface. The Customer ISDN Interface will be connected to the ISDN Central Office together with other Customer ISDN Interfaces. The digital pipe between the Central Office and the ISDN user will be used to carry a number of communication channels. The capacity of this digital pipe and therefore the number of channels available will vary from user to user. The basic channel structure of the presently available Narrowband ISDN (N-ISDN) for instance consists of two full-duplex 64 Kbps B channels and a full-duplex 16 Kbps D channel. The B channels can handle both circuit-switched and packet-switched traffic types. A circuit-switched call is equivalent to the switched digital service available in the current Plain Old Telephone Services (POTS). The user places a call and a circuit-switched connection is established with another network user. Circuit-switching technology refers to the transfer of continuous information such as voice (telephone call) or video. In the packet-switched traffic type, the user is connected to a packet-switching node and data is exchanged with other users via a communication protocol such as the X.25. Packet-switching is used whenever a bursty traffic type of call occurs, like in electronic mail or terminal to mainframe computer communications. The designation of 64 Kbps as the standard user channel rate was chosen to be compatible with digital voice. The main purpose of the D channel is to carry any signalling information to control circuit-switched calls on the associated B channels at the user interface.

The digital pipes interconnecting the ISDN Central Offices will be able to support many more channels in order to efficiently accommodate all the customers using the network. The present standard for the N-ISDN network consists of 24 channels of 64 Kbps each, per digital pipe. In the near future though users will request transmission speeds higher than those currently offered in the N-ISDN T1 standard of 1.5 Mbps. These higher speeds (from

the already established T3 rate of 45 Mbps to rates in the range of 150 to 600 Mbps) will be required in order to realize Local Area Network-to-LAN interconnections, High Definition Television (HDTV) distribution, and so on. The preferred transport method in a fully evolved B-ISDN multimedia communication network is the ATM, or Asynchronous Transfer Mode. The ATM transport method is suitable for a multimedia traffic environment because it offers a great flexibility in bandwidth allocation through the assignment of fixed length packets, called cells, to virtual connections on a demand basis. Each ATM cell consists of a header and an information field. Although CCITT has agreed that ATM will be based on fixed-length cells (since they are easier to process at higher speeds), the length of the cells and the content or length of the cell headers have not yet been agreed upon. Currently a header field length of 5 bytes and an information field length of 64 bytes appear to be favored by the US. A layered view of the ATM transport method is given in Figure 1.2. The traffic flows originating at the service layer are converted into ATM format by the ATM adaptation layer, and then transferred within cells via multiplexing and/or switching by the ATM transport layer which is supported by a physical layer. In order to maximize the information transfer speed, the traffic will be best controlled at the point of access to the ATM transport network (which corresponds to the ISDN Central Office in Figure 1.1) in order to avoid additional per-node buffering and processing delays.

The ATM transport method is expected to handle a wide variety of traffic types ranging from circuit-like connections of fixed size and guaranteed bandwidth to highly bursty data services. Integrating the various traffic types is an important task in B-ISDN. Therefore, the schemes for packet multiplexing, bandwidth allocation and congestion control should be investigated to ensure fairness and efficient resource usage. It is also important to identify and understand some of the traffic types likely to be integrated in B-ISDN. They may have

differences in tolerance to queueing delay, buffering, and bandwidth requirements. It can therefore be useful to characterize the following B-ISDN traffic types:

(1) *Delay-sensitive high-bandwidth services*: This traffic type requires a fixed large bandwidth for the duration of a call and demands real-time service. Examples of this traffic type are conference video, real-time image processing, document retrieval, and local area network interconnects.

(2) *Delay-insensitive high-bandwidth services*: This traffic type includes bulk information transport services. The user can designate the extent of delay which is tolerable (specified, for instance, as minutes, hours, or overnight). Examples of this traffic type include delay-tolerent document, image, and video delivery services.

(3) *Delay-sensitive low-bandwidth statistically multiplexed services*: This traffic type includes calls which are delay sensitive with end-to-end delay requirements ranging from a few tens to a few hundreds of milliseconds. Examples of such traffic are packetized voice, interactive data, and enquiry-response messages.

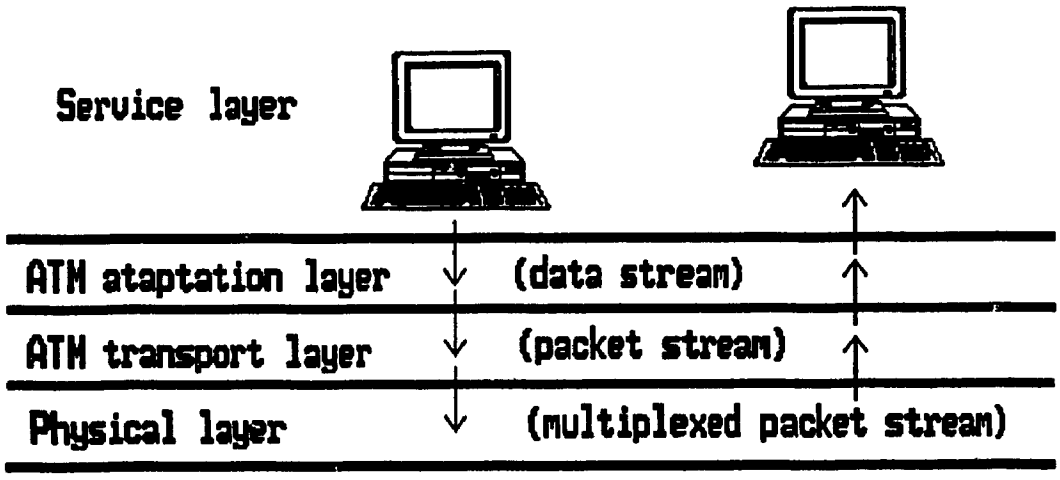

Figure 1.2 Layered View of the ATM Protocol Model.

In this work, B-ISDN traffic will be assumed to fall into two categories; circuit-switched and packet-switched traffic. Circuit-switched traffic may also be referred to as non-queueable since a call of this type cannot be stored or queued in a buffer. Packet-switched traffic on the other hand may be referred to as queueable since a packet can be queued in a buffer until the transmission line becomes free. Each of these traffic types are further divided into a number of classes of calls where each class has its own call arrival rate and call duration. The circuit-switched classes of calls are moreover classified with their bandwidth requirements in number of channels and the packet-switched classes of calls are classified with their packet arrival rate per call per second. To handle such a wide range of traffic flow characteristics and to make efficient use of the associated digital pipe, close control of the access mechanism in the ISDN Central Office will be required.

1.1 Access Control Policies

In order to accept the transmission of a call in the digital pipes between the ISDN Central Offices, the access controller must ensure that the specific requirements associated with the call are available. As an answer to a circuit-switched type of call for instance, a given number of free channels ought to be dedicated from the source ISDN Central Office to the destination ISDN Central Office during the entire communication time. If the bandwidth requirement is available then a virtual circuit will be established between the sending and the receiving central offices. Packet-switched calls on the other hand are usually carried out by sending the packets constituting a message from one central office to the next in a store-and-forward fashion. A packet-switched call may hence request a given time limit on the delay which can occur in the communication transfer at every central office. An access control policy is therefore required to provide users with a fair share of the resources

available while maximizing the utilization of the node. Access control policies fall into one of the following categories: *complete sharing, complete partitioning, preemptive priority* and *optimal policy*. The first three access control categories are suboptimal in the sense that they do not yield optimal utilization of the bandwidth available. This work is concerned with the optimal policy which maximizes the use of a link while ensuring the requirements for all classes of calls. In what follows the suboptimal access policies are introduced together with examples and then the optimal access policy is described.

1.1.1 Suboptimal Access Control Policies

The easiest access scheme to implement is complete sharing. Complete sharing accepts a call whenever sufficient bandwidth exists on the communication link to accommodate the call. Complete sharing, however, suffers from fairness between users when it is used to control the access in a communication network where many classes having different requirements are allowed. For example an increase in the arrival rate of a given class of call will decrease the share of the resources available to the other classes of calls in the network. No control is given in complete sharing to allocate a fair share of the link to the other classes of calls by blocking a call from the first class, since a call is allowed on the link as long as there is sufficient bandwidth available.

Complete partitioning is an access scheme used in order to avoid conflicts between the different classes of calls. Complete partitioning separates the available bandwidth in such a way that each class of calls possesses a portion of the link. Using this access control policy a call is accepted as long as sufficient bandwidth exists in the reserved area of the link for that specific class of calls. The drawback with this technique is the possible waste

of link capacity. For example, some area of a link may very well be unused while other area can suffer from congestion. Complete partitioning will not allow a call corresponding to the congested area to access the unused portion of the link since calls of each class are only accepted if there is enough bandwidth in their restricted area of the link.

The preemptive priority access control scheme is a technique specially useful when both circuit-switched and packet-switched traffic types are integrated on the same link. In this access scheme a fixed number of channels is reserved for the use of circuit-switched traffic. A packet-switched call is allowed in the idle portion of the link allocated to circuit-switched calls with the risk of being preempted whenever a circuit-switched call requests it. The packet-switched traffic is thus able to use the idle portion of the link whenever a sudden increase in its arrival rate occurs, which is a great advantage. Due to the fixed boundary limit for circuit-switched calls, this technique suffers from a lack of flexibility. The boundary position cannot be optimally set to provide the best possible utilization of the communication link at all times since the arrival rates of circuit-switched and packet-switched calls vary. Some actions must further be taken to avoid packet loss when packets are preempted from the circuit-switched reserved area of the link.

Several studies have already been conducted to address the above suboptimal access policies and their variations showing how they would perform under a variety of integrated traffic environments. For instance, the implementation of the preemptive priority access control strategy (shown in Figure 1.3) was applied to the control of wide-band non-queueable traffic type (WB) and narrow-band queueable traffic type (NB) [1]*. Another study [2]

* Numbers in brackets designate references at the end of the thesis.

reports the performance analysis of a system under different control strategies, namely, First In First Out (Complete Sharing) and Preemptive Priority, in the case where only all-queued traffic types are assumed to share the same broadband channel (or digital pipe). From these studies, the best access scheme to be used heavily depends on the traffic allowed in the communication network. One of the key issues found is that fairness to the users may be achieved only at some expense of reduced bandwidth utilization.

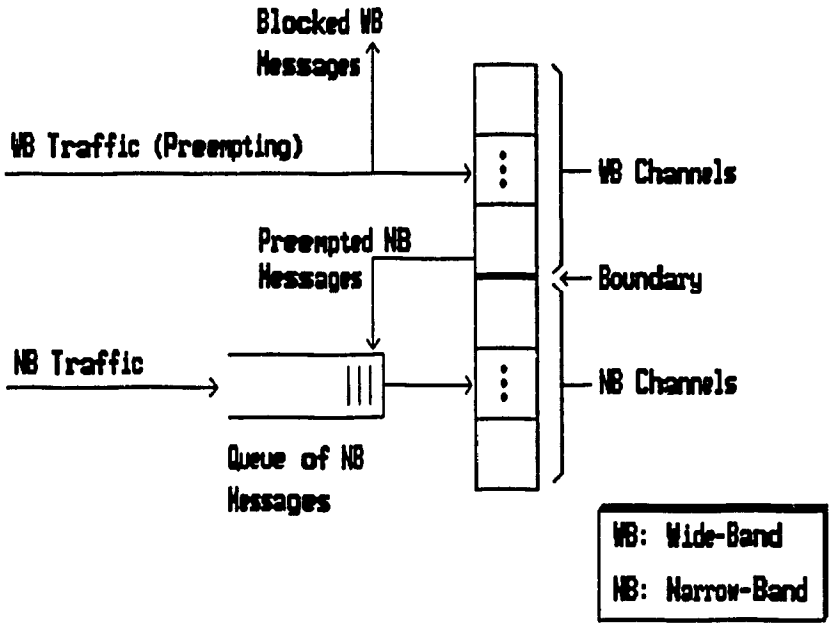

Figure 1.3 Preemptive Priority (PP) Access Control Strategy.

Another study [3] analyses the utilization of an integrated system where voice, video, and data communications share a single link under the preemptive priority scheme. A certain grade of service is assumed for circuit-switched calls (video and voice) and a minimum delay is allowed for packet-switched calls. The system proposed permits the location of a boundary maximizing the utilization of a link while keeping the user fairness to the best level possible. This system permits the maximum utilization of a link under the preemptive

access control strategy, as long as the traffic parameters (such as the arrival rate) of the different classes of calls do not vary too much in time. This thesis is concerned with the access control of an integrated communication network where the traffic parameters are fast changing.

1.1.2 Optimal Access Control Policies

The decision to accept or reject a call in the suboptimal access policies depends only on the present state of the communication network. The state of a communication network corresponds to the number of calls currently using a link for every allowed class of calls. The accept/reject decision can be taken on a real time basis in the case of suboptimal access policies since upon arrival of a call the access controller only needs to know the amount of free space available in the link and the requirements related to the class of the call in order to accept or reject the call. These policies maximize the utilization of the link in each state independently of one another and they are qualified as suboptimal for this reason. An optimal access policy on the other hand maximizes the overall utilization of a link by taking into consideration all possible transitions from the states. For example even though there might be enough bandwidth left in the link for a given class of call, the optimal access controller could reject the access to a call from that class in order to accept another call requesting more bandwidth later on, if this action results in an increase in the overall link utilization. The optimal access policy operates like a dynamic movable boundary in an integrated environment. Every time a circuit-switched call is accepted, the capacity of the link to serve packet-switched calls will decrease accordingly. At any time, the channels unused by the circuit-switched calls will serve packet traffic on a First-Come-First-Served (FCFS) queueing discipline. In order to guarantee a certain grade of service for packet-switched

calls, the optimal policy may allocate a minimum capacity to packet-switched calls based on their present load at any time. In fact the optimal access policy provides fairness in the access to the resources since all classes of calls are able to specify their own requirements. These requirements might be a maximum blocking probability for a circuit-switched class of calls or a maximum probability that the packet delay in a packet-switched call transmission will not be longer than a specified maximum. The optimal policy could be very useful as the access control strategy for the ATM transport method for instance. In a multimedia environment it will be critical that the ATM transport method satisfies the cell delay and loss performance requirements adequate for all applications supported [4]. One of the main reasons which makes the optimal access policy the best choice as an access controller in an integrated communication system is its adaptability to guarantee all users requirements in a varying environment.

The optimal access policy technique has been studied lately in various papers [5][6]. The problem of controlling the access to a communication link consisting of a heterogeneous mix of circuit-switched traffic is usually formulated as a Semi-Markov Decision Process (SMDP) to find the optimal access policy. Unfortunately even though the optimal policy gives the best possible utilization of the resource, one is required either to solve a Linear Programming problem or to use a value iteration algorithm to determine the optimal access policy. These techniques eliminate the possibility of a real time computation process to determine the accept/reject decision in all but the simplest cases using the present uniprocessor sequential computer technology since the time required to solve the problem increases exponentially with the size of the problem. An optimal access controller for instance might need several seconds to compute the best decision upon a call arrival, which should be

answered in a very short time delay (a few milliseconds) [7]. If this processing time problem could be solved then the optimal access scheme could be effectively implemented to maximize the utilization of a link.

1.2 Research Contributions and Scope of the Thesis

The first major contribution of this thesis is the extension of the optimal access scheme to maximize the utilization of a link for circuit-switched traffic to include packet-switched traffic as well. This new concept allows us to find the optimal access policy corresponding to a multimedia ISDN communication system, for example. In order to fully integrate packet-switched traffic and circuit-switched traffic types on the same link, the concept of a dynamic movable boundary is used. A dynamic movable boundary ensures that both circuit-switched and packet-switched traffic requirements are always satisfied, except when the capacity of the resource is pushed to the limit (in overload situation). With the dynamic movable boundary technique, the resource is fairly shared between the packet-switched and circuit-switched traffic types, which may not be the case when ordinary boundary access schemes are used as discussed above. The results allow specification of user requirements for both circuit-switched and packet-switched traffic.

As a second major contribution of this thesis, we tackled the processing time problem required to find the optimal access policy corresponding to a given communication system. This processing time problem may be solved using the fast parallel computing power found in *Neural Networks*. A neural network is a highly interconnected network of simple analog processors (called neurons) which can collectively compute good solutions to difficult optimization problems. The type of neural network considered here is a modified version

of the neural network first proposed by D.W. Tank and J.J. Hopfield which was specially designed to solve Linear Programming problems [8]. This type of neural network has also been studied for other optimization applications such as in high-speed packet switch control [9], or in traffic routing and flow allocation [10]. Another type of neural network has recently been studied to be used as an access controller in an integrated services communication system [11]. The proposed ATM communication network access controller is implemented using neural networks with back-propagation in order to learn the relations between offered traffic and service qualities. Once trained this neural network takes a decision upon accepting or rejecting a call much faster than a uniprocessor computer would do. Since the learning model used to train the neural network is a variation of complete sharing, the utilization of the resource under this technique would however still be lower than what could be provided by an optimal access policy.

This thesis is organized as follows:

Chapter 2 introduces the reader to the neural network computation approach. After an overview of the different neural network models, the neural network used to solve constrained optimization problems is fully described. Topics such as the momentum (or energy) function corresponding to a neural network, and the implementation of equality and inequality restrictions are discussed. The physical implementation of the neural network from its momentum function using basic electrical circuit components is also covered. Care is taken in the physical implementation to allow easier VLSI fabrication of the neural network although the VLSI implementation itself is not provided here.

Chapter 3 first demonstrates how the optimal access policy problem for a communication network integrating both circuit-switched and packet-switched traffic types can be

modelled using a Semi-Markov Decision Process (SMDP). Then the mapping of the SMDP communication model to a Linear Programming problem is described. Examples of optimal access policy computation problem for both constrained and unconstrained communication system are also given. In an unconstrained communication system, the optimal policy does not take the user requirements into consideration. Finally the queueing model techniques used to compute the packet delay probability which allows packet-switched classes of calls to set their service requirements is fully described.

Chapter 4 is devoted to the simulation results obtained from the optimal access policy. The possible implementation of an access controller based on the optimal access policy is given first. Then the program used to compute the different parameters required by the Linear Programming neural network, and the Linear Programming simulation program emulating the action of the neural network itself are described. The simulation results obtained from an unconstrained communication system where the access scheme is based on the optimal access policy, and where only circuit-switched traffic is allowed are first given first. The simulation results provide the utilization of a link under various parameters such as the arrival rate of the different classes of calls, the capacity of the link in number of channels, the number of channels taken by each class of calls, and so on. Then simulation results show the obvious advantage of integrating packet-switched traffic with circuit-switched traffic on the utilization of a link. The last simulation results demonstrate that the utilization of a link decreases as a certain grade of service is allowed to a given class of call.

Finally a summary of the main findings and results of this thesis are given and suggestions for further research are provided in Chapter 5.

CHAPTER 2

THE NEURAL NETWORK MODEL

2.1 Introduction

Intelligent behavior in a person seems to emerge from interaction involving a huge number of neurons -each of which is quite limited in its processing capabilities (i.e., with regard to its speed, the information it acts upon, and the information it produces). Similarly with a neural-network approach, information is processed through the interaction of a large number of simulated neurons. In the first part of this chapter the neural network approach is introduced in general terms. The neural network specifically used to solve Linear Programming problems is then described. The physical implementation of the neural network using basic electrical circuit components is fully covered.

2.2 The Neural Network Approach

The neural network models have so far been used in different areas such as pattern recognition, artificial intelligence problems, and approximations to large optimization problems. The basic element of any neural network is the neuron. An artificial neuron has four main components: (see Figure 2.1)

- *synapses* (input connections), through which the neuron receives activation from other neurons.
- a *summation function* which combines the various input activations into a single activation.

- a *threshold function* which converts this summation of input activation into output activation.
- *axonal paths* (output connections) by which a neuron's output activation arrives as input activation to other neurons in the system.

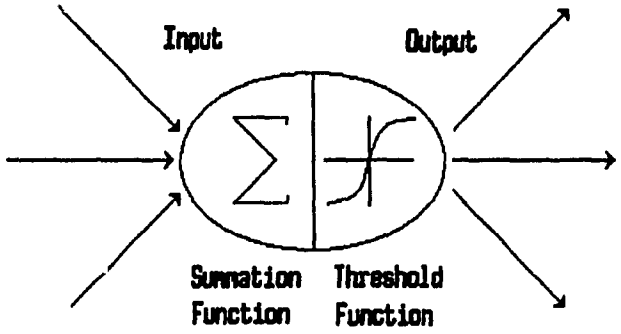

Figure 2.1 A Simulated Neuron.

A neural network consists of a large number of neurons communicating together using their inter-neuron connections, that is, the synapses and the axonal paths. An inter-neuron connection in a neural network is typically assigned a weight value which modulates the activation passing through the connection. If the connection from neuron A to neuron B has a weight w_{BA} for example then the activation output of neuron A is multiplied by this value to determine the activation actually received by B. The absence of a connection between A and B can be represented by simply assigning w_{BA} a 0 value. An inhibitory relation between A and B is modeled by giving w_{BA} a negative value. A neural network is driven by the numerically-valued activation which passes from neurons to other neurons. The knowledge of a neural network lies in its inter-neuron connections and their corresponding weights.

There are two popular models of neural networks; the feedback model and the feed-forward model. In the feed-forward model (shown in Figure 2.2) the neurons are arranged into layers so that the flow of information propagates from the input layer towards the output layer. The operation of the feed-forward model is similar to that of a combinational circuit where the inputs propagate and interact in one direction to produce the output. The feed-forward model is appropriate to solve problems such as pattern recognition for example.

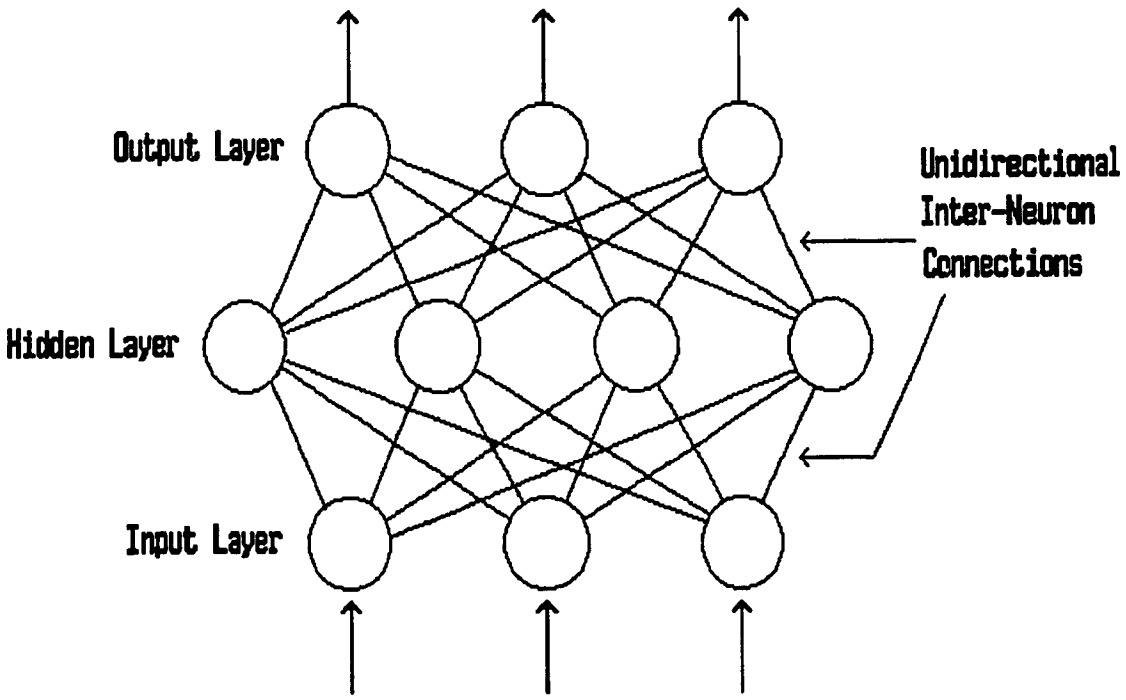

Figure 2.2 Feed-forward Neural Network.

The architecture of the feedback neural networks can be described as an undirected graph (see Figure 2.3) since the flow of information propagates bidirectionally between the neurons. The operation of the feedback model is closer to that of sequential circuit where the system is initialized to a given state and evolves in time to a final state. The computation time in feed-forward neural network is simply given as the time required for the signals to

propagate from the input layer to the output layer. The evolution of the feedback model in time is more complex and may be analyzed with the help of a momentum (or energy) function. Under favorable conditions the momentum function decreases monotonically as the neural network evolves from one state to the next.

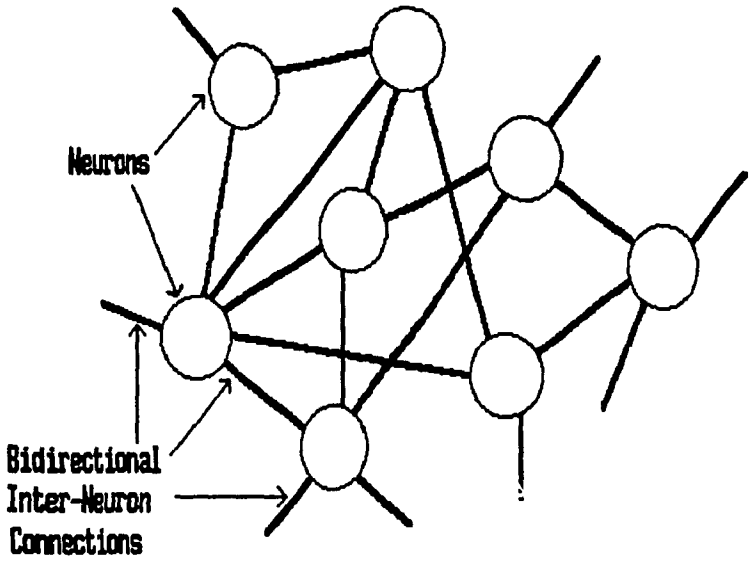

Figure 2.3 Feedback Neural Network.

Optimization problems are typically implemented on a feedback neural network. The feedback neural network is therefore suitable to determine an optimal access policy since a Linear Programming optimization process can be used to find a solution to this problem. One famous example of optimization process is the Traveling Salesman Problem (TSP) in which a salesman is supposed to tour N cities and desires to minimize the total distance of the tour. There are N! possible solutions and this number increases rapidly with the problem size. This task is computationally intensive on a present-generation digital computer. Using the original feedback neural network model as proposed by Hopfield and Tank a very good solution could be found in a relatively short time, the problem being distributed among all

neurons. In the remainder of this chapter, the physical implementation of a neural network used to solve a Linear Programming problem is described. The Linear Programming optimization technique will be used later on to find the optimal access policy corresponding to a given communication system.

2.3 Linear Programming Neural Networks

The behavior of a neural network using the feedback approach is best described by its so called momentum (or energy) function. The momentum function is chosen in such a way to have its minimum activation once the optimum solution is found. A Linear Programming problem can be stated as the attempt to maximize an objective function such as

$$\Pi = \vec{A} \cdot \vec{V} \tag{2.1}$$

where "•" refers to the dot product operation and $\vec{A}$ is an N-dimensional vector of coefficients for the N variables which are the components of $\vec{V}$, that is,

$$\Pi = \begin{bmatrix} A_1 \\ A_2 \\ . \\ . \\ A_N \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ V_N \end{bmatrix} \tag{2.2}$$

This maximization process is to be accomplished subject to a set of I linear inequality constraints and E linear equality constraints among the variables:

$$\vec{D_i} \cdot \vec{V} \leq B_i, \quad i = 1,2,..,I \tag{2.3}$$

$$\vec{D_e} \cdot \vec{V} = B_e, \quad e = I+1,..,I+E \tag{2.4}$$

for

$$\vec{D_j} = \begin{bmatrix} D_{j,1} \\ D_{j,2} \\ . \\ . \\ D_{j,N} \end{bmatrix}, \quad j = 1,2,..,I+E \tag{2.5}$$

where the $\vec{D_j}$, for each j, contains the N variable coefficients in a constraint equation and B_j are the bounds.

In order to better understand how the neural network proceeds to find an optimum solution to an optimization problem, one can introduce the following analogy. The components of the variable vector $\vec{V}$ can be treated as the coordinates of a point in a Euclidean space of N dimensions. Call this point the objective point. The objective function (equation 2.1) is continuous and single valued everywhere within this space and therefore can be used to define a gradient vector, grad Π. If $\vec{u}_1, \vec{u}_2, .., \vec{u}_N$ are unit vectors in the directions of the coordinate axes, then

$$\text{grad}\,\Pi = \overline{\Pi} = \sum_{k=1}^{N} \frac{\partial \Pi}{\partial V_k} \vec{u}_k = \sum_{k=1}^{N} A_k \vec{u}_k. \tag{2.6}$$

The vector $\overline{\Pi}$ is everywhere constant, normal to the hyperplanes of equal Π, and in the direction of steepest ascent.

The i^{th} inequality restriction of equation 2.3 represents a hypervolume in the N-space bounded by the hyperplane $\vec{D_i} \cdot \vec{V} = B_i$. This bound is called the i^{th} edge, and corresponding to this edge one may also define a vector $\vec{T_i}$ normal to the i^{th} hyperplane. An edge separates

the region of the space in which an inequality is satisfied from the region in which it is violated. The space in which all of the given inequalities are satisfied is called the allowed region. The allowed region is closed in the sense that the objective function cannot increase indefinitively without entering a forbidden region. An example of this representation is shown in Figure 2.4.

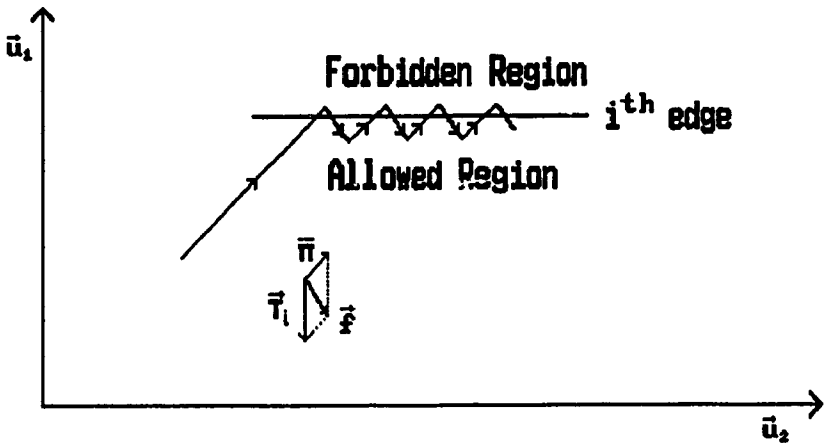

Figure 2.4 Motion of Objective Point Near a Restriction.

Assuming only inequality restrictions one can imagine a point (the objective point) moving through the allowed region along $\vec{\Pi}$ until it reaches a restriction such as the i^{th} edge (see Figure 2.4). Then the motion is determined by two vectors; $\vec{\Pi}$, and $\vec{T_i}$ which is multiplied by a factor z corresponding to the distance between the point and the i^{th} edge. If $z\vec{T_i}$ is greater than $\vec{\Pi}$ then the point will be ejected from the forbidden region (along the resultant vector $\vec{f}$). The actual rebound is infinitesimal in magnitude as the point moves back to the allowed region. The influence of vector $\vec{T_i}$ then vanishes, and the gradient $\vec{\Pi}$ causes the motion to reverse until the point again enters the forbidden region. This way, the objective point moves along the restriction boundary in the direction of the projection of the vector grad Π on the i^{th} hyperplane [12].

2.4 Physical Implementation of the Linear Programming Neural Networks

As a Linear Programming problem solver the neural network is required to maximize the objective function (equation 2.1) while ensuring that all equality restrictions as well as all inequality restrictions are satisfied (equations 2.3 and 2.4). The momentum function for such a neural network can be stated as [13]

$$M = -(\vec{A} \cdot \vec{V}) + \sum_{i=1}^{I} G(\vec{D_i} \cdot \vec{V} - B_i) + \sum_{e=I+1}^{I+E} H(\vec{D_e} \cdot \vec{V} - B_e) \tag{2.7}$$

where $\vec{A} \cdot \vec{V}$ is the objective function to be maximized, and $G(z)$ and $H(z)$ are the functions which will ensure that the inequality restrictions as well as the equality restrictions are respectively met. Notice here that only inequality restrictions were allowed in the original Hopfield and Tank model. As the neural network tries to find a valid optimal solution, the restrictive functions $G(z)$ and $H(z)$ will penalize the system whenever an inequality or an equality restriction is violated. The value of $G(z)$ should hence be high only when an inequality restriction is not met while $H(z)$ should penalize proportionally to the divergence of the objective point from a given equality restriction. Two such penalizing functions could be given as follows:

$$G(z) = \left\{ \begin{matrix} 0 & \text{if } z \le 0 \text{ (allowed region)} \\ \frac{\gamma z^2}{2}, \gamma > 0 & \text{if } z > 0 \text{ (restricted region)} \end{matrix} \right\} \tag{2.8}$$

and

$$H(z) = \left\{ \frac{\gamma z^2}{2}, \gamma > 0 \right\}. \tag{2.9}$$

The variable z here can be interpreted as the distance between the objective point position and a boundary restriction. The coefficient γ is a positive number used to adjust the scaling of the penalizing function. A value too low would not "penalize" the objective point motion enough while a value too high could cause divergence, and hence oscillation in the neural network solution. These penalizing functions were chosen because of the facility with which their physical implementations can be obtained as it will be demonstrated later on. $G(z)$ has no effect in the momentum function as long as the objective point respects every inequality restriction. For the equality penalizing function, it is assumed that the objective point is always in the forbidden region. $H(z)$ thus has a value proportional to the distance between an equality boundary and the neural network solution, so that the objective point will be forced towards this boundary. Furthermore one may obtain the circuit equation of the neural network from its momentum function (equation 2.7). To do so, the negative of the partial derivative of the momentum function is taken with respect to $\vec{V}$ (where each component of $\vec{V}$ will represent the output voltage of a neuron) as follows:

$$-\frac{\partial M}{\partial V_k} = C\frac{dV_k}{dt} = A_k - \sum_{i=1}^{I} D_{i,k} g(\vec{D_i} \cdot \vec{V} - B_i) - \sum_{e=I+1}^{E+I} D_{e,k} h(\vec{D_e} \cdot \vec{V} - B_e),\ k = 1,2,..,N \tag{2.10}$$

where

$$g(z) = \frac{dG(z)}{dz} \text{ and } h(z) = \frac{dH(z)}{dz}. \tag{2.11}$$

The schematic diagram of such a neural network is shown in Figure 2.5. The inputs of the neural network are the components of vector $\vec{A}$ (see equation 2.2); the boundary values of the inequality restrictions B_i, $i = 1, 2, ..., I$; and the boundary values of the equality restrictions B_e, $e = I+1, ..., I+E$. The outputs of the neural network correspond to $\vec{V}$. In

Figure 2.5, **f** represents the variable amplifiers which provides the neural network solution to the Linear Programming problem once stabilization is achieved (the optimum being found). The circuit equation of the variable amplifiers is given by equation 2.10. The amplifiers **g** and **h** in Figure 2.5 represent the inequality and the equality restriction amplifiers respectively, and their circuit equations are given by

$$\Psi_i = g(\vec{D_i} \cdot \vec{V} - B_i) \text{ and } \Psi_e = h(\vec{D_e} \cdot \vec{V} - B_e). \tag{2.12}$$

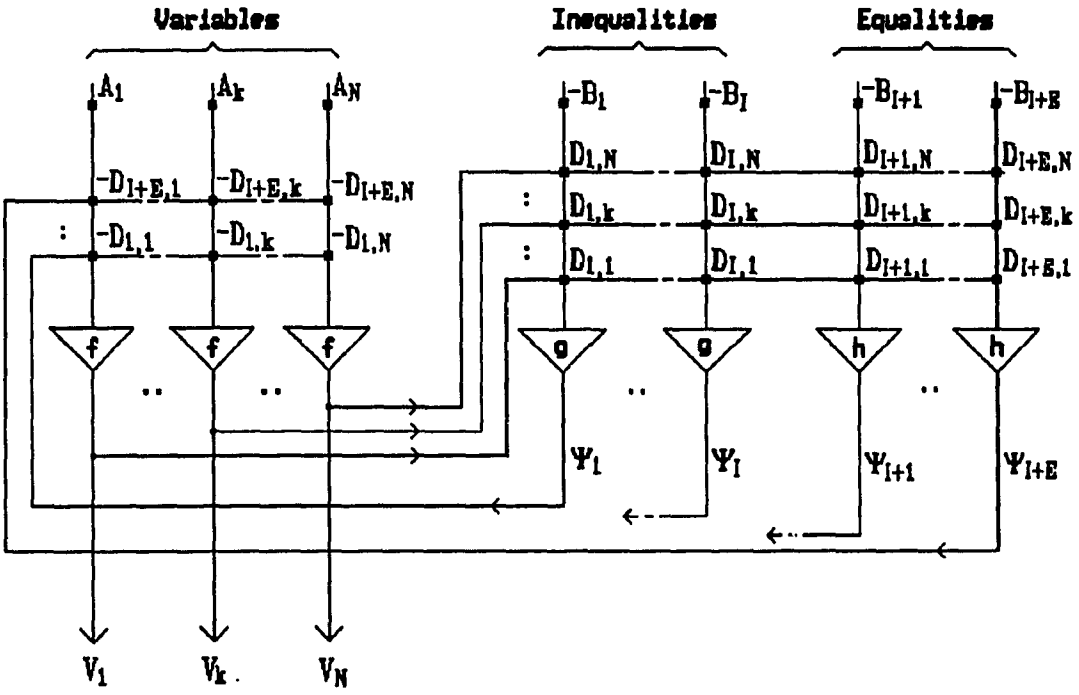

Figure 2.5 Linear Programming Neural Network.

To physically implement the variable and restriction amplifiers, one can use the following four basic electrical circuits which are shown in Figures 2.6, 2.7, 2.8, and 2.9, that is, the summing integrator, the summing adder, the precision amplifier, and the inverter

respectively. The characteristic equation corresponding to the summing integrator can be given as

$$V_{int} = -\frac{1}{C}\int \left(\frac{U_0}{r_0}+\frac{U_1}{r_1}+\frac{U_2}{r_2}+\ldots+\frac{U_L}{r_L}\right) dt + U_c \tag{2.13}$$

where U_c is the initial voltage stored in capacitor C (as shown in Figure 2.6), whereas the characteristic equation corresponding to the summing adder circuit can be written as follows:

$$V_{add} = -R_f\left(\frac{U_0}{r_0}+\frac{U_1}{r_1}+\frac{U_2}{r_2}+\ldots+\frac{U_L}{r_L}\right) \tag{2.14}$$

The functions of both the inverter and the precision rectifier circuits are to negate the potential given at their respective inputs. However the precision rectifier does not respond to a positive potential, that is,

$$V_{out} = \max(-V_{in}, 0). \tag{2.15}$$

Each of the basic circuit elements provides an almost ideal linear characteristic function for the operating range required by the neural network.

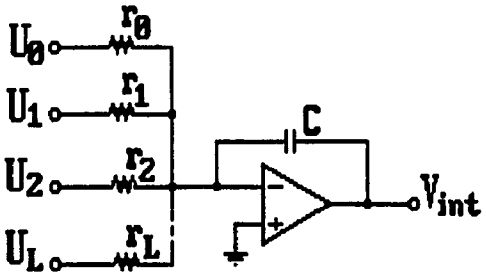

Figure 2.6 Summing Integrator.

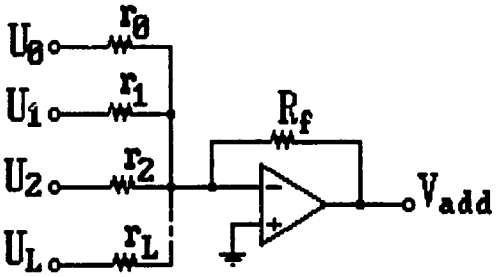

Figure 2.7 Summing Adder.

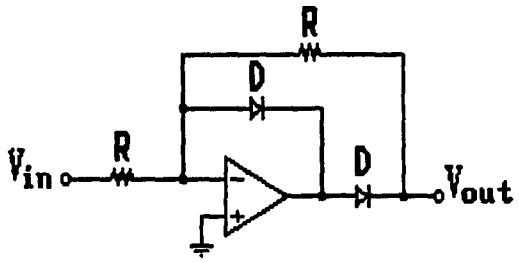

Figure 2.8 Precision Rectifier.

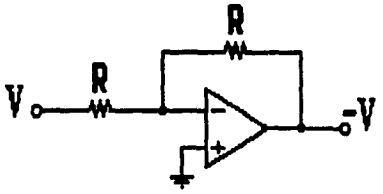

Figure 2.9 Inverter.

The neural network main circuit equations (see equation 2.10) required in order to solve a Linear Programming problem can be integrated to give a relation with the output voltage of the k^{th} variable amplifier as follows:

$$V_k = \frac{1}{C}\int \left(A_k - \sum_{i=1}^{I} D_{i,k}\Psi_i - \sum_{e=I+1}^{I+E} D_{e,k}\Psi_e \right) dt, \quad k = 1,2,..,N \tag{2.16}$$

where

$$\Psi_i = g(\overrightarrow{D_i} \cdot \overrightarrow{V} - B_i) \text{ and } \Psi_e = h(\overrightarrow{D_e} \cdot \overrightarrow{V} - B_e). \tag{2.17}$$

In most Linear Programming applications however there is a set of inequality constraints which are used to ensure that the range of values for the decision variables are kept positive. One may thus further reduce the complexity of the overall neural network physical implementation if a simple modification is done to equation 2.16 in order to restrict the range of the function to allow only positive output values as follows:

$$V_k = \max\left\{ \frac{1}{C}\int \left(A_k - \sum_{i=1}^{I} D_{i,k}\Psi_i - \sum_{e=I+1}^{I+E} D_{e,k}\Psi_e \right) dt, 0 \right\}, \quad k = 1,2,..,N. \tag{2.18}$$

Next, the physical implementation of variable amplifiers, equality and inequality restriction amplifiers will be discussed.

2.4.1 Variable Amplifier Implementation

In order to implement the modified variable amplifiers, three of the basic circuit elements are required, that is, the summing integrator, the precision rectifier (in order to provide the modification brought into equation 2.18) and the inverter. The inverter is used to allow a negative output for the variable amplifier. The weights ($D_{j,k}$; $k = 1, 2, ..., N$; $j = 1, ..., I+E$) found in the neural network (see Figure 2.5) are implemented using resistors. Unfortunately, one cannot have negative resistors corresponding to negative weights. Instead the inverse of the potential which was supposed to be applied to the resistor will be used in such a case. The following figure (Figure 2.10) presents an electrical circuit used to implement the variable amplifiers. As it may be seen, it consists of a summing integrator, a precision rectifier and an inverter. Next, the choice of the parameters in this circuit to express equation (2.18) will be explained.

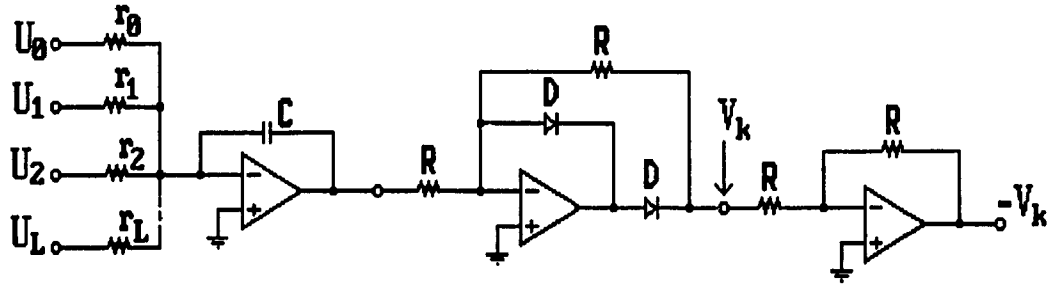

Figure 2.10 Implementation of the Variable Amplifier.

For example, let us begin by assuming that the first input in Figure 2.10 has been used to implement A_k in equation 2.18. One can then set U_0 to plus or minus one volt, given the sign of A_k as follows:

$$U_0 = \frac{A_k}{|A_k|}, \tag{2.19}$$

and r_0 can thus be set as the inverse of the magnitude of A_k so that the following equality is met:

$$\frac{U_0}{r_0} = A_k, \tag{2.20}$$

Each of the other inputs may be used to generate the summation terms in equation 2.18. For example one can set r_j to the inverse of the magnitude of the corresponding weight; namely,

$$r_j = \frac{1}{|D_{j,k}|}, \quad j = 1,2,..,I+E, \tag{2.21}$$

and U_j can now be set to the corresponding output of the restriction amplifier (taking care of the sign of the weight at the same time):

$$U_j = \frac{-\Psi_j D_{j,k}}{|D_{j,k}|}, \quad j = 1,2,..,I+E, \tag{2.22}$$

so that the following requirements are met:

$$\frac{U_j}{r_j} = -\Psi_j D_{j,k}, \quad j = 1,2,..,I+E. \tag{2.23}$$

2.4.2 Equality Restriction Amplifier Implementation

An equality restriction amplifier may be physically implemented using a summing adder and an inverter as shown in Figure 2.11. The circuit equation driving an equality restriction amplifier can be obtained from equations 2.9, 2.11, and 2.12 as follows:

$$\Psi_e = \gamma \cdot \left(-B_e + \sum_{k=1}^{N} D_{e,k} V_k \right), \quad e = I+1,..,I+E. \tag{2.24}$$

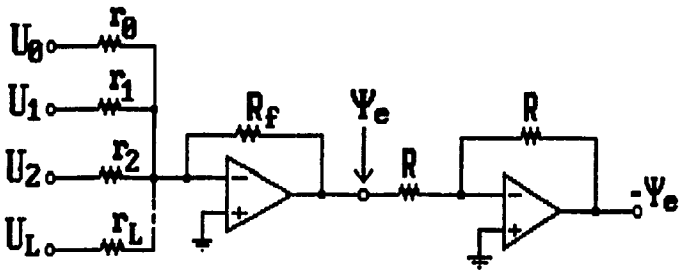

Figure 2.11 Implementation of the Equality Restriction OpAmp.

The first input in this case corresponds to an equality boundary condition and can be easily implemented much like the first input of the variable amplifiers. One can let the potential U_0 to be 1 or -1 volt following the sign of the corresponding boundary condition:

$$U_0 = \frac{B_e}{|B_e|}, \tag{2.25}$$

and r_0 can then be set to the inverse of the magnitude of the boundary condition so that the following equalities are met:

$$\frac{U_0}{r_0} = B_e. \tag{2.26}$$

Each of the other inputs should correspond to one of the outputs coming from the variable amplifiers. Hence, one can set r_k to the inverse of the magnitude of the corresponding weight:

$$r_k = \frac{1}{|D_{e,k}|}, \; k = 1,2,..,N, \tag{2.27}$$

and U_k can now be set to the output corresponding to the variable amplifier (taking care of the sign of the weight at the same time):

$$U_k = \frac{-V_k D_{e,k}}{|D_{e,k}|}, \; k = 1,2,..,N \tag{2.28}$$

so that the following equations are met:

$$\frac{U_k}{r_k} = -V_k D_{e,k},\ k = 1,2,..,N. \tag{2.29}$$

One final step required here is to adjust the scaling of the penalizing function by setting the resistance R_f to the appropriate value, that is,

$$R_f = \gamma. \tag{2.30}$$

2.4.3 Inequality Restriction Amplifier Implementation

Finally one can implement the inequality restriction amplifiers with the use of a precision rectifier as shown in Figure 2.12. The circuit equation driving an inequality restriction amplifier can be obtained from equations 2.8, 2.11, and 2.12 as follows:

$$\Psi_i = \gamma \cdot \max\left(-B_i + \sum_{k=1}^{N} D_{i,k} V_k,\ 0\right),\ i = 1,2,..,I \tag{2.31}$$

The first input in this case corresponds to an inequality boundary condition. Here again, one can let the potential U_0 to be 1 or -1 volt following the sign of the corresponding boundary condition:

$$U_0 = \frac{B_i}{|B_i|},\ i = 1,2,..,I, \tag{2.32}$$

and r_0 can be set to the inverse of the magnitude of the boundary condition so that the following equalities are met:

$$\frac{U_0}{r_0} = B_i, \tag{2.33}$$

Each of the other inputs should correspond to one of the outputs of the variable amplifiers. For example, one can set r_k to the inverse of the magnitude of the corresponding weight:

$$r_k = \frac{1}{|D_{i,k}|}, k = 1, 2, .., N, \tag{2.34}$$

and U_k can now be set to the output corresponding to the variable amplifier (taking care of the sign of the weight at the same time):

$$U_k = \frac{-V_k D_{i,k}}{|D_{i,k}|}, k = 1, 2, .., N \tag{2.35}$$

so that the following equations are met:

$$\frac{U_k}{r_k} = -V_k D_{i,k}, \; k = 1, 2, .., N. \tag{2.36}$$

Another step also required here is to adjust the scaling of the penalizing function by setting the resistance R_f to the appropriate value, that is,

$$R_f = \gamma. \tag{2.37}$$

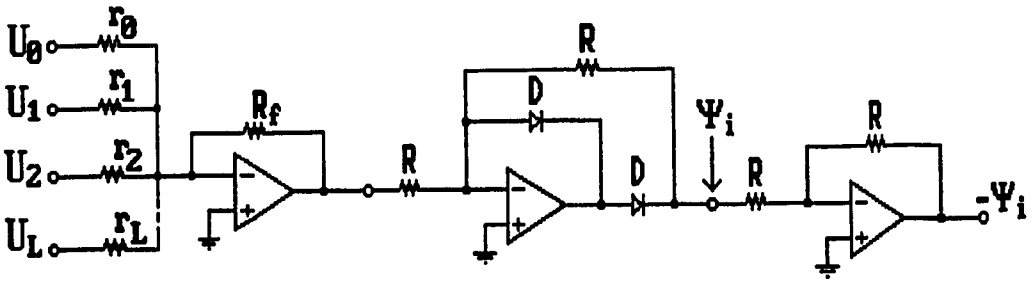

Figure 2.12 Implementation of the Inequality Restriction OpAmp.

For the Linear Programming neural network physical implementation to work properly, the response time of the restriction amplifiers must however be negligible compared to that of the variable amplifiers. Oscillation could otherwise occur in which case the neural

network outputs would not stabilize. Although the VLSI implementation of the Linear Programming neural network is not covered in this work some characteristics may be outlined here. In the procedure given above, one may notice that only resistor values are required to set the weights in the neural network. This will help the implementation of the Linear Programming neural network on VLSI chips where the resistor values can be set much like the way memory locations are set in the memory chips. The neural network implementation will hence be able to solve different Linear Programming problems by simply reprogramming its resistor value. To simulate the absence of connection between two neurons a VLSI implementation will also have to provide the possibility of an open circuit between two neurons. As it will be seen in the next chapter the minimum number of outputs required to get the optimal access policy solution from the Linear Programming neural network corresponds to the number of classes of calls allowed in the communication network.

CHAPTER 3

THE OPTIMAL ACCESS CONTROL POLICIES

3.1 Introduction

The main goal of this chapter is to show how to express the optimal access policy corresponding to a multimedia communication network using the neural network developed in the previous chapter. The problem of determining the optimal access policy for a circuit-switched communication network which supports traffic classes with varying bandwidth requirements is first addressed. A Markov Decision Process approach is employed to obtain the optimal access policy as it was demonstrated in [6]. This approach is then extended here so that packet-switched traffic can be allowed in the communication system. This extension is quite general and allows different classes of packet-switched traffic to the network. The mapping of the Markov decision process to a Linear Programming problem is then introduced in order to implement the corresponding optimal access policy using a neural network. Examples of neural network implementations for both unconstrained and constrained optimal access policies will be given afterwards. Only the optimal utilization of the link is important in an unconstrained optimal access policy, that is, the policy will not take into account the various call requirements in this case. On the other hand the constrained policy allows users to specify maximum blocking probability or delay that they may be subjected to in the network. Finally the queueing models which can be used to compute the packet delay in the system, allowing a certain grade of service for packet-switched users, will be described.

3.2 The communication Network Model

As it may be recalled from Chapter 1, the future B-ISDN multimedia communication network will consist of interconnected Central Offices, or CO (see Figures 1.1 and 3.1). Each of these Central Offices will be able to carry the transmitted information from the source CO to the destination CO through the use of wideband communication links such as fiber optic cables. A communication link n will be characterized by its bandwidth in number of channels (m_n) which are supported by the link. For example a transmission rate of 1.5 Mbps is required in order to handle the 24 channels of 64 Kbps each as it is currently specified in the *T1* telephone network standard for digital communication. One may now proceed with the description of the communication network model.

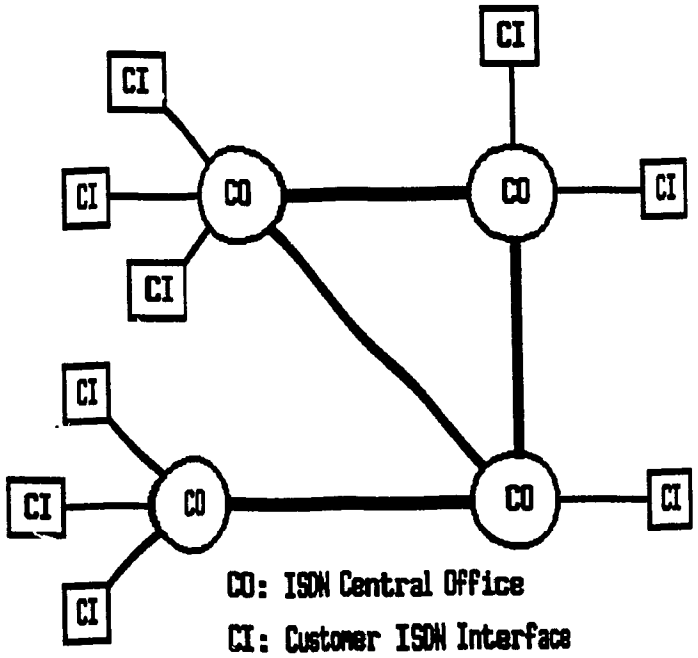

Figure 3.1 B-ISDN Communication Network.

In the following C_n circuit-switched classes and P_n packet-switched classes of calls will be allowed in link n, for a total of $C_n + P_n = K$ classes of calls. Each class of calls will be identified with their Poisson process call arrival rate and their exponentially distributed call duration, that is,

$$\lambda_k \text{ calls per second}, \quad k = 1, 2, .., K \tag{3.1}$$

$$\frac{1}{\mu_k} \text{ seconds per call}, \quad k = 1, 2, .., K. \tag{3.2}$$

Furthermore a circuit-switched class of call will also be identified with its bandwidth requirement, that is,

$$b_k \text{ channels per call}, \quad k = 1, 2, .., C_n, \tag{3.3}$$

and a packet-switched class of call will generate packets with exponentially distributed length following a Poisson process during the duration of a call with an arrival rate of

$$\eta_k \text{ packets per call per second}, \quad k = C_n + 1, .., C_n + P_n. \tag{3.4}$$

Finally each channel will be characterized by a service rate of ω packets per channel per second.

The state of a link at any time may be given by $x = (x_1, x_2, ..., x_k, ..., x_K)$ where x_k represents the number of class-k calls which are carried out by the link at that time. The capacity of the link is assumed to be divided into two parts with a dynamic movable boundary. Let

m_{xc} = the number of channels occupied by the circuit-switched calls when the system is in the state x, and

m_{xp} = the number of channels occupied by the packet-switched calls when the system is in the state x.

Hence $m_{xc} + m_{xp} = m_n$ where m_n is the capacity of the link in number of channels at any time. The utilization of a channel used for circuit-switched purposes being always one in any given state x, that is, $\rho_{xc} = 1$, the number of channels occupied by the circuit-switched calls may be computed as follows:

$$m_{xc} = \sum_{k=1}^{C_n} x_k b_k. \tag{3.5}$$

Clearly $m_{xp} = m_n - m_{xc}$. Note here that m_{xc} and m_{xp} will change dynamically. It is assumed that the packet arrival due to all packet-switched calls will form a single queue. Clearly this packet arrival process will be a Poisson process. The packets in the queue will be served according to the FCFS queueing discipline by the m_{xp} channel servers. The utilization of a channel in a given state x used for packet-switching purposes may thus be computed as follows:

$$\rho_{xp} = \sum_{k=C_n+1}^{C_n+P_n} \frac{x_k \eta_k}{m_{xp}\omega}. \tag{3.6}$$

Hence $x_k\eta_k$ refers to the total arrival rate of packets per second in the link from the class-k packet-switched calls and $m_{xp}\omega$ represents the service rate of the system in number of packets per second. Notice here that this ratio (equation 3.6) should always be kept below one $(\rho_{xp} < 1)$ to avoid infinite packet delay. Now, it is possible to express the overall utilization of the link for a given state x as

$$r(x) = \sum_{k=1}^{C_n} x_k b_k + \sum_{k=C_n+1}^{C_n+P_n} x_k \eta_k/\omega. \tag{3.7}$$

The state space, that is, the space of all possible states, may be derived from equation 3.7 as follows:

$$\Lambda = \left\{ x: x \geq 0;\ \sum_{k=1}^{C_n} x_k b_k + \sum_{k=C_n+1}^{C_n+P_n} x_k \eta_k/\omega \ \leq\ m_{xc} + m_{xp} = m_n \right\} \tag{3.8}$$

A state is thus allowed in the state space as long as the bandwidth requirement for that state is not greater than the capacity of the link itself. To every state of the state space corresponds a state dependent action space. The state dependent action space determines the decision to permit or deny access to every class of call in the link. The set of all possible actions (the action space) is related to the number of classes of calls (K), and may be expressed as follows:

$$B = \{ (a_1, a_2, ..., a_K): a_k \in \{0, 1\} \} \tag{3.9}$$

where

$$a_k = \begin{Bmatrix} 0, & \text{a class-}k \text{ call access will be rejected} \\ 1, & \text{a class-}k \text{ call access will be accepted} \end{Bmatrix}.$$

Clearly, the optimal access policy assigns a given action to every state of the state space in such a way that the overall utilization of the link is optimized. Let us assume for example that only 2 classes of calls are allowed in the link. The action space can then be enumerated as:

$$B = \{(0,0)(0,1)(1,0)(1,1)\}, \tag{3.10}$$

that is, reject both classes of calls, accept only a class-2 call, accept only a class-1 call, or accept both classes of calls respectively. If action $a = (a_1, a_2) = (0,1)$ is chosen in a given state then only a class-2 call will have access to the link. Actually only a subset of these actions (the state dependent action space) are possible for a given state. If the capacity of the link in state x is fully utilized for instance then the only possible action would be to

deny access to every class of calls, that is, $B_x = \{(0,0)\}$, or if the link is idle then the chosen action should accept at least one class of calls. The departure of a call (that is, no further need of the link from a particular call) being always accepted, the state dependent subset of the action space can thus be represented as follows for a given state x:

$$B_x = \{a \in B: a_k = 0 \text{ if } x + e_k \notin \Lambda\} \tag{3.11}$$

where

e_k = a vector of zeroes except for the k^{th} component.

Hence the optimization process required to provide the optimal access policy must find the best action to choose from the state dependent action space for every state of the state space in order to maximize the utilization of the communication link over all states. Finally it may be noticed that the cardinality of the state space, that is, the number of elements in the state space, is given as $|\Lambda|$, whereas the cardinality of the action space (equation 3.9) is 2^K. One may compare the technique used in this work with another technique [5] where the cardinality of the state space was $2K|\Lambda|$ and the cardinality of the action space was 2 for an arrival and 1 for a departure, the decision epoch being taken after the occurrence of an event rather than before like in our case. Clearly, in the state descriptor used here, an event occurs upon a call departure from the link or upon a call arrival in the system. A decision epoch occurs every time the state of the system changes, that is, upon departure of a call from the link or upon acceptance of a call in the link. Hence the decision to accept or reject a call will already be made before the arrival of a call. If the call is accepted then the state of the link changes and a decision epoch occurs to make sure that the system will be ready for the next event. It has been proved in [7] that the state descriptor used here results in faster optimal access policy computation.

3.3 Formulation of an Optimal Access Policy as a Semi-Markov Decision Process and as a Linear Programming Problem

Now that the communication network model has been established we can proceed to find the optimal access policy itself. In order to obtain an optimal access policy the communication network needs to be expressed as a Semi-Markov Decision Process (SMDP). As it may be remembered from Chapter 1, the main characteristic of the optimal access policy is to maximize the overall utilization of a link over all states of the state space. Consider for instance a communication network with a link of 10 channels ($m_n = 10$), 3 classes of calls ($K = 3$), and the following parameters for circuit-switched classes: $(b_1, b_2, b_3) = (1,4,8)$, $(\lambda_1, \lambda_2, \lambda_3) = (1,1,1)$, and $(1/\mu_1, 1/\mu_2, 1/\mu_3) = (1,1,1)$, as shown in Table 3.1.

Table 3.1 Traffic Parameters.

Network Parameters	Value		
	Class k=1	Class k=2	Class k=3
λ_k	1	1	1
$1/\mu_k$	1	1	1
b_k	1	4	8

In this example the optimal access policy assigns the same action in every state as would do the complete sharing access policy with one exception: in state $(x_1, x_2, x_3) = (2,0,0)$ the optimal policy blocks class-1 call because even though there is enough bandwidth in the link, the optimal policy prefers to wait for a "wider" class-2 or class-3 call. This way the overall utilization of the link will be increased [14]. The remaining of this section shows

how the Semi-Markov Decision Process can be expressed as a Linear Programming problem. Our goal will then be achieved since a Linear Programming problem can be solved using a neural network implementation as it was shown in Chapter 2.

Let us first tackle the problem of *unconstrained* access policies. An unconstrained optimal access policy maximizes the utilization of a link without considering any of the user requirements. This policy will thus provide the optimal utilization of a link at the price of a possible degradation in user services. Let us also assume for now that the time interval between two consecutive decision epoches (arrival or departure of a call) is always the same. Using this assumption we can represent the optimal access policy as a discrete Markov decision process. The overall utilization of a link in a given state x can be expressed as follows:

$$r(x) = \sum_{k=1}^{C_n} x_k b_k + \sum_{k=C_n+1}^{C_n+P_n} x_k \eta_k / \omega. \tag{3.12}$$

Also of importance is the transition probability from, say, state x to, say, state y at a decision epoch given that action a is chosen. This probability can be stated as follows:

$$\tilde{P}_{xay} = \begin{cases} \lambda_k a_k & \text{if } y = x + e_k, \ k = 1, \ldots, K \\ \mu_k x_k & \text{if } y = x - e_k, \ k = 1, \ldots, K \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

where

e_k = a vector of zeroes except for the k^{th}component.

The Markov Decision Process may now be expressed as a Linear Programming problem to find a solution to the optimal access problem. First the objective function must be chosen to provide the best action to take in each state of the state space so as to maximize the long-run overall utilization of the link. If one lets z_{xa} to be the long-run fraction of

decision epoches at which the system is in state x and action a is chosen then the Linear Programming problem corresponding to the Discrete Markov Decision Process in z_{xa} can be expressed as follows [15]:

Maximize $$\sum_{x \in \Lambda} \sum_{a \in B_x} r(x) z_{xa} \tag{3.14}$$

subject to $$\sum_{a \in B_y} z_{ya} \sum_{x \in \Lambda} \tilde{P}_{yax} = \sum_{x \in \Lambda} \sum_{a \in B_x} \tilde{P}_{xay} z_{xa}, \ \forall\, y \in \Lambda, \tag{3.15}$$

$$\sum_{x \in \Lambda} \sum_{a \in B_x} z_{xa} = 1, \tag{3.16}$$

$$z_{xa} \geq 0, \ x \in \Lambda \text{ and } a \in B_x. \tag{3.17}$$

The first set of restrictions (equation 3.15) represents the balance equations requiring that for every state $y \in \Lambda$ the long-run average number of transitions from state y per time unit must be equal to the long-run average number of transitions into state y per time unit. Now since these equations must be respected for every state, the number of equality restrictions required will be $|\Lambda|$, that is, the number of possible states for the communication system. The next equality restriction (equation 3.16) emphasizes the fact that the sum of the fractions z_{xa} must be equal to 1. The last set of inequality restrictions (equation 3.17) makes sure that the values of the decision variables z_{xa} are always positives [16].

However in real life, the time spent by the system in each state will not be the same. We hence cannot let the time between consecutive decision epoches be constant. So instead of having a discrete Markov decision process the optimal access policy will be represented by a semi-Markov decision process. In order to obtain the semi-Markov decision process

from a discrete Markov decision process, we need to know the expected time until a new state is entered from a given state $x \in \Lambda$ when action $a \in B_x$ is chosen. This expected time may be given as follows [15]:

$$\tau(x,a) = \left(\sum_{k=1}^{K} \mu_k x_k + \sum_{k=1}^{K} \lambda_k a_k \right)^{-1}. \tag{3.18}$$

Notice here that in order for equation 3.18 to hold the call arrivals are assumed to follow a Poisson distribution and the service time of the calls is assumed to be exponentially distributed. The next step is to express the semi-Markov decision process as a Linear Programming problem. This is done by multiplying each term of the Linear Programming problem corresponding to the discrete Markov decision process by the expected time. The resulting Linear Programming problem in z_{xa} which corresponds to the semi-Markov decision process is given as follows [15]:

Maximize
$$\sum_{x \in \Lambda} \sum_{a \in B_x} r(x)\tau(x,a)z_{xa} \tag{3.19}$$

subject to
$$\sum_{a \in B_y} z_{ya} \sum_{x \in \Lambda} \tilde{P}_{yax}\tau(y,a) = \sum_{x \in \Lambda} \sum_{a \in B_x} \tilde{P}_{xay}\tau(x,a)z_{xa}, \quad \forall\, y \in \Lambda, \tag{3.20}$$

$$\sum_{x \in \Lambda} \sum_{a \in B_x} \tau(x,a)z_{xa} = 1, \tag{3.21}$$

$$z_{xa} \geq 0, \; x \in \Lambda \text{ and } a \in B_x. \tag{3.22}$$

This representation can be simplified since the transition probabilities from a state y with a given action a should sum up to one, that is,

$$\sum_{x \in \Lambda} \tilde{P}_{yax} \tau(y,a) = 1. \tag{3.23}$$

And if we further let

$$P_{xay} = \tilde{P}_{xay} \tau(x,a), \tag{3.24}$$

that is,

$$P_{xay} = \begin{cases} \lambda_k a_k \tau(x,a) & \text{if } y = x + e_k, \ k = 1,\ldots,K \\ \mu_k x_k \tau(x,a) & \text{if } y = x - e_k, \ k = 1,\ldots,K \\ 0 & \text{otherwise.} \end{cases} \tag{3.25}$$

the resulting Linear Programming formulation for the semi-Markov decision process becomes [15]

Maximize $$\sum_{x \in \Lambda} \sum_{a \in B_x} r(x) \tau(x,a) z_{xa} \tag{3.26}$$

subject to $$\sum_{a \in B_y} z_{ya} = \sum_{x \in \Lambda} \sum_{a \in B_x} P_{xay} z_{xa}, \quad \forall\, y \in \Lambda, \tag{3.27}$$

$$\sum_{x \in \Lambda} \sum_{a \in B_x} \tau(x,a) z_{xa} = 1, \tag{3.28}$$

$$z_{xa} \geq 0, \ x \in \Lambda \text{ and } a \in B_x. \tag{3.29}$$

Notice here that the *steady state* probability of being in state x and choosing action a is given by $\tau(x,a)z^*_{xa}$ for the semi-Markov decision process. For each state $x \in \Lambda$, there will be at most one action $a \in B_x$ in the unconstrained case such that $z^*_{xa} > 0$; call this action $\pi_a(x)$. If $z^*_{xa} = 0$ for all $a \in B_x$, then $\pi_a(x)$ should be set to an arbitrary element of B_x. The optimal policy will choose action $\pi_a(x)$ whenever in state x, maximizing over all policies the long-run overall utilization of the link [15].

3.4 Neural Network Implementation of the Unconstrained Optimal Access Policy

As it was demonstrated in Chapter 2, a Linear Programming problem may be solved using a neural network. Next the implementation of the Linear Programming problem corresponding to the unconstrained optimal access policy will be described. The objective function (equation 3.26) and the Linear Programming restrictions (equations 3.27-3.29) being found, we may now proceed with the following associations. The components of vector $\vec{A}$ (see equation 2.2 and Figure 2.5) correspond to the coefficients of the decision variables in the objective function (equation 3.26), that is, the $r(x)\pi(x,a)$ terms. The outputs of the variable amplifiers (vector $\vec{V}$ in equation 2.2) correspond to the decision variables themselves, that is, z_{xa}. Notice here that only equality restriction amplifiers are required in the neural network to find an unconstrained optimal access policy (the inequality restrictions represented by $z_{xa} \geq 0$ being taken care of with the modified variable amplifiers as discussed in Chapter 2, equation 2.18). Let us assume that the first equality restriction amplifier takes care of the restriction given in equation 3.28. One of the inputs to this restriction amplifier should hence be set to the boundary condition value, that is, $B_1=1$ (see Figure 2.5). The other inputs should be set to the output value of the variable amplifiers multiplied by the

corresponding weights in $\tau(x,a)$. The remaining of the equality restriction amplifiers serves to ensure that every equality restriction given in equation 3.27, which may also be written as

$$\sum_{a \in B_y} z_{ya} - \sum_{x \in \Lambda} \sum_{a \in B_x} P_{xay} z_{xa} = 0, \quad \forall\, y \in \Lambda, \tag{3.30}$$

are satisfied. The number of elements required in the neural network in order to solve an unconstrained optimal access problem may be stated as follows:

$$|\text{Var. Ampl.}| = \sum_{x \in \Lambda} |B_x| \;\le\; 2^K |\Lambda|. \tag{3.31}$$

$$|\text{Rest. Ampl.}| = |\Lambda| + 1. \tag{3.32}$$

$$\begin{aligned} |\text{Weights}| &= 2\,(|\text{Var. Ampl.}|)\,(|\text{Rest. Ampl.}|) \\ &= 2\left(\sum_{x \in \Lambda} |B_x|\,(|\Lambda|+1)\right) \;\le\; 2^{K+1}|\Lambda|^2 + 2^{K+1}|\Lambda|. \end{aligned} \tag{3.33}$$

$$\begin{aligned} |\text{OpAmps}| &= 3\,(|\text{Var. Ampl.}|) + 2\,(|\text{Equ. Rest. Ampl.}|) + 3\,(|\text{Ineq. Rest. Ampl.}|) \\ &= 3 \sum_{x \in \Lambda} |B_x| + 2\,(|\Lambda|+1) \;\le\; 3 \cdot 2^K |\Lambda| + 2|\Lambda| + 2. \end{aligned} \tag{3.34}$$

Next, the neural network implementation of the unconstrained optimal access policy will be illustrated by a simple example. A link with three channels ($m_n = 3$) is to be shared between two circuit-switched classes of calls. Let the arrival rate and the service time of the calls to be equal to unity, that is, $(\lambda_1, \lambda_2) = (1,1)$, and $(1/\mu_1, 1/\mu_2) = (1,1)$; and let us assume that the two classes of calls require one and two channels respectively, that is, $(b_1, b_2)=(1, 2)$ (see Table 3.2 for a more concise form). Figure 3.2 shows the state diagram of such a communication network.

Table 3.2 Traffic Parameters.

Network Parameters	Value	
	Class k=1	Class k=2
λ_k	1	1
$1/\mu_k$	1	1
b_k	1	2

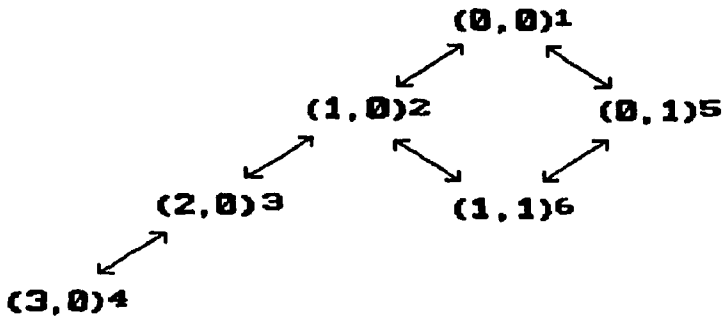

Figure 3.2 State Space Diagram.

The state space can also be enumerated as

$$\Lambda = \{ (0,0)\ (1,0)\ (2,0)\ (3,0)\ (0,1)\ (1,1) \}. \tag{3.36}$$

(Note here that the states have been sequentially numbered in Figure 3.2 for later use to get a more concise notation). Since there are only two classes of calls, the action space corresponding to this example can be enumerated as follows:

$$B = \{ (0,0)\ (0,1)\ (1,0)\ (1,1) \}. \tag{3.37}$$

A specific state dependent action space corresponds to every state of the state space. Table 3.3 enumerates all actions which are allowed in a given state.

Table 3.3 State Dependent Action Space.

State Number	State x	Action Space B_x			
		(a_1,a_2)=(0,0)	(a_1,a_2)=(0,1)	(a_1,a_2)=(1,0)	(a_1,a_2)=(1,1)
1	(0,0)		X	X	X
2	(1,0)	X*	X	X	X
3	(2,0)	X		X	
4	(3,0)	X			
5	(0,1)	X		X	
6	(1,1)	X			

* Symbol X means that the action is allowed in the corresponding state.

With this information in mind we can proceed with the computation of the main parameters for the semi-Markov decision process, that is, the overall utilization of the link in every state of the state space, $r(x)$, the expected time until next state, $\tau(x,a)$, and the transition probabilities, P_{xay}. Let us first obtain the overall utilization of the link in every state. Since there are only two classes of circuit-switched calls, the utilization can be expressed from equation 3.7 as:

$$r(x) = \sum_{k=1}^{2} x_k b_k = x_1 + 2x_2. \tag{3.38}$$

Table 3.4 gives the overall utilization of the link for every state of the state space for this example.

Table 3.4 Overall Utilization of the Link.

State Number	State x	Utilization $r(x)$
1	(0,0)	0*
2	(1,0)	1
3	(2,0)	2
4	(3,0)	3
5	(0,1)	2
6	(1,1)	3

* $r(x)$ computed from equation 3.38.

The expected time until next state and the transition probabilities are given in Tables 3.5 and 3.6 respectively. The equation of the expected time can be obtained from equation 3.18 as:

$$\tau(x,a) = \left(\sum_{k=1}^{2} \mu_k x_k + \sum_{k=1}^{2} \lambda_k a_k\right)^{-1} = \frac{1}{x_1 + x_2 + a_1 + a_2}, \tag{3.39}$$

whereas the transition probability formula can be derived from equation 3.25 as:

$$P_{xay} = \begin{cases} a_k\tau(x,a) & \text{if } y = x + e_k,\ k = 1,\ldots,K \\ x_k\tau(x,a) & \text{if } y = x - e_k,\ k = 1,\ldots,K \\ 0 & \text{otherwise.} \end{cases} \tag{3.40}$$

Table 3.5 Expected Time Until Next State.

State Number	State x	Expected Time $\tau(x,a)$			
		$(a_1,a_2)=(0,0)$	$(a_1,a_2)=(0,1)$	$(a_1,a_2)=(1,0)$	$(a_1,a_2)=(1,1)$
1	(0,0)	-	1	1	1/2
2	(1,0)	1*	1/2	1/2	1/3
3	(2,0)	1/2	-	1/3	-
4	(3,0)	1/3	-	-	-
5	(0,1)	1	-	1/2	-
6	(1,1)	1/2	-	-	-

* $\tau(x,a)$ computed from equation 3.39.

Table 3.6 Transition Probabilities.

State #	Transition Probability P_{xay}																							
	(a_1,a_2)=(0,0)						(a_1,a_2)=(0,1)						(a_1,a_2)=(1,0)						(a_1,a_2)=(1,1)					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	0*	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	$\frac{1}{2}$	0	0	$\frac{1}{2}$	0
2	1	0	0	0	0	0	$\frac{1}{2}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$
3	0	1	0	0	0	0	0	0	0	0	0	0	0	$\frac{2}{3}$	0	$\frac{1}{3}$	0	0	0	0	0	0	0	0
4	0	0	$\frac{1}{1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$	0	0	0	0	$\frac{1}{2}$	0	0	0	0	0	0
6	0	$\frac{1}{2}$	0	0	$\frac{1}{2}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* P_{xay} computed from equation 3.40.

We are now ready to give the parameters for the resulting Linear Programming problem (from equation 3.26 to 3.29). Let us also sequentially number actions in the action space, that is, $B = \{(0,0)^1, (0,1)^2, (1,0)^3, (1,1)^4\}$, so that the decision variable z_{xa} corresponds to state x and action a. The objective function (equation 3.26) for our example may thus be stated as:

$$\Pi = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \\ A_9 \\ A_{10} \\ A_{11} \\ A_{12} \\ A_{13} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ V_{11} \\ V_{12} \\ V_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1/2 \\ 1/2 \\ 1/3 \\ 1 \\ 2/3 \\ 1 \\ 2 \\ 1 \\ 3/2 \end{bmatrix} \cdot \begin{bmatrix} z_{12} \\ z_{13} \\ z_{14} \\ z_{21} \\ z_{22} \\ z_{23} \\ z_{24} \\ z_{31} \\ z_{33} \\ z_{41} \\ z_{51} \\ z_{53} \\ z_{61} \end{bmatrix} \tag{3.41}$$

where "•" represents the dot product, and the equality restrictions given in equation 3.27 and 3.28 may be represented as:

$$\begin{bmatrix} 1 & 1 & 1 & -1 & \frac{-1}{2} & \frac{-1}{2} & \frac{-1}{3} & 0 & 0 & 0 & -1 & \frac{-1}{2} & 0 \\ 0 & -1 & \frac{-1}{2} & 1 & 1 & 1 & 1 & -1 & \frac{-2}{3} & 0 & 0 & 0 & \frac{-1}{2} \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{2} & \frac{-1}{3} & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{3} & 1 & 0 & 0 & 0 \\ -1 & 0 & \frac{-1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \frac{-1}{2} \\ 0 & 0 & 0 & 0 & \frac{-1}{2} & 0 & \frac{-1}{3} & 0 & 0 & 0 & 0 & \frac{-1}{2} & 1 \\ 1 & 1 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} & 1 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} z_{12} \\ z_{13} \\ z_{14} \\ z_{21} \\ z_{22} \\ z_{23} \\ z_{24} \\ z_{31} \\ z_{33} \\ z_{41} \\ z_{51} \\ z_{53} \\ z_{61} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix} \tag{3.42}$$

With these results in mind we may now proceed with the neural network implementation corresponding to the communication network as shown in Figure 3.3. In this figure, 13 variable amplifiers (**f**) and 7 equality restriction amplifiers (**h**) are found which results in (3*13) + (2*7) = 53 OpAmps and 2(13)(7) = 182 weights (resistors)-(see equations 3.31 to 3.34). The number of amplifiers found here is the strict minimum allowable to solve our simple example, if the neural network is implemented as shown in Chapter 2. The point to keep in mind here is that the Linear Programming problem will be distributed among all the amplifiers to find a solution.

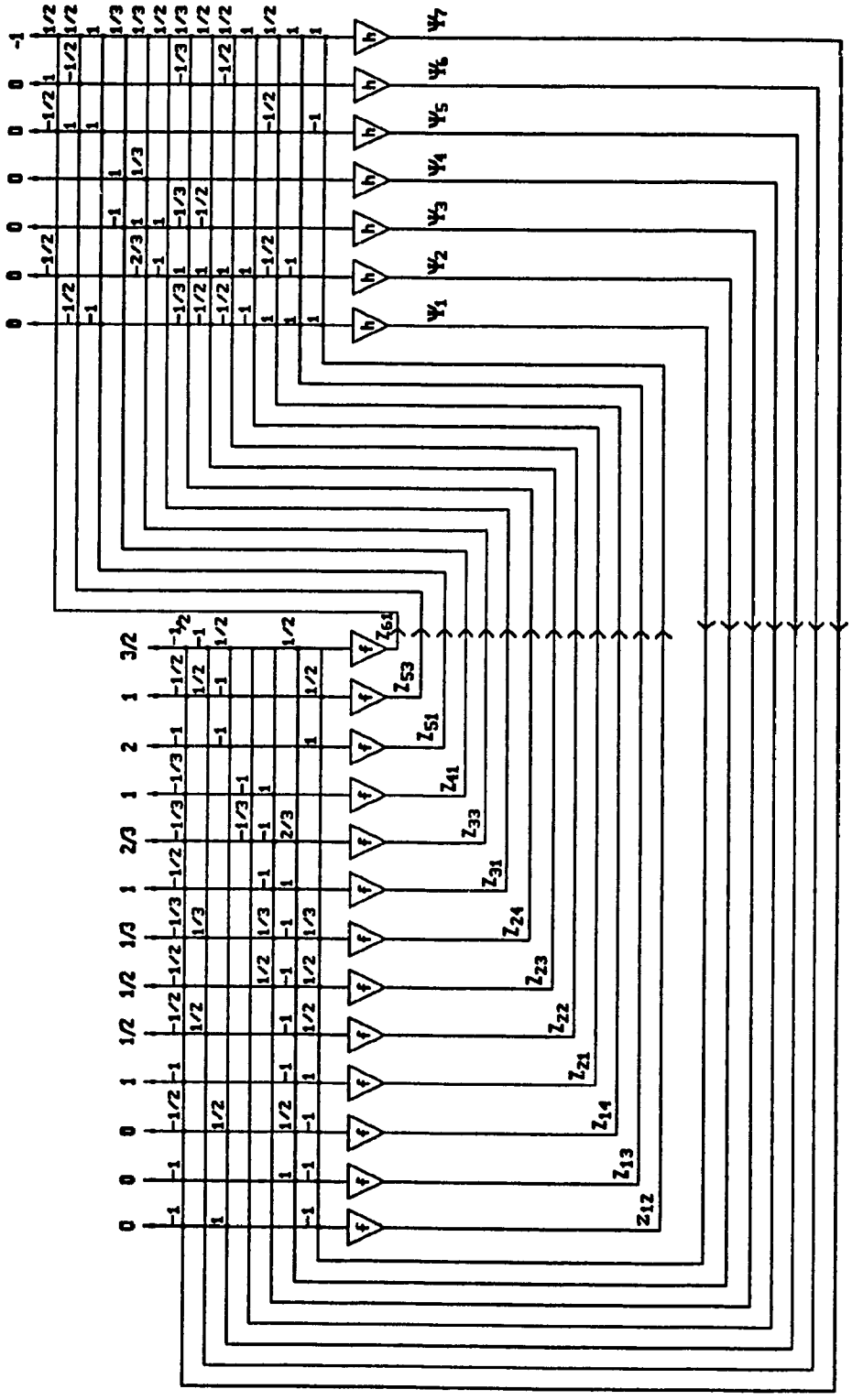

Figure 3.3 Neural Network Corresponding to the Unconstrained Example Given in this Section.

3.5 Neural Network Implementation of the Constrained Optimal Access Policy

Let us now turn our attention to the case of optimal access policies when the communication system is subject to user requirements. Guaranteeing to meet the user constraints results in a decreased utilization of the network resources compared to that of unconstrained optimization for the benefit of the user satisfaction as to be demonstrated with the simulation results in Chapter 4. These constraints will enable circuit-switched users to specify the maximum blocking probability and packet-switched users to specify the maximum delay probability which can occur during their packet transmission. First the theoretical basis which will allow a user to add these restrictions will be developed and then a communication network example will be given.

The optimal access policy functions as a dynamic movable boundary system. Thus as the circuit-switched traffic increases, the capacity allocated to the packet-switched traffic will decrease, and therefore the delay seen by these users will increase. During the temporary overload conditions, this degradation of the performance may not be acceptable to the packet-switched traffic. The constrained optimal access policy prevents this from happening by allowing packet-switched users to specify a maximum delay probability that their packet may experience.

So far in the unconstrained optimization we have only considered stationary policies under which the actions are chosen deterministically. It is also possible to impose constraints on certain state frequencies in such a way to satisfy the user needs. A policy π is called a stationary *randomized* policy when it can be described by a probabilistic distribution $\{\pi_a(x) \geq 0, a \in B_x\}$ for each state $x \in \Lambda$ such that action a is to be chosen with probability $\pi_a(x)$ whenever the process is in state x. In the case where $\pi_a(x)$ is 0 or 1 for every x and a, the stationary randomized policy π reduces to the familiar stationary process policy choosing the actions in a deterministic way. For any policy π, let the state-action frequencies $f_{xa}(\pi)$ be defined by [17]

$f_{xa}(\pi)$ = the long-run fraction of decision epoches at which the process is in state x and action a is chosen when policy π is used.

Suppose now that the goal of a semi-Markov decision process is to maximize the long-run overall utilization of a link subject to the following *linear* constraints associated with given classes of calls on the state-action frequencies,

$$\sum_{x \in \Lambda} \sum_{a \in B_x} \alpha_{xa}^{(k)} f_{xa}(\pi) \leq \beta^{(k)} \quad \text{for } k = 1, .., K \tag{3.44}$$

where $\alpha_{xa}^{(k)}$ and $\beta^{(k)}$ are given *constants*. Notice here that $\alpha_{xa}^{(k)}$ depends only on the state and the specific action chosen. It can be shown that an optimal access policy may be obtained when the following Linear Program is solved [18]:

Maximize $$\sum_{x \in \Lambda} \sum_{a \in B_x} r(x,a)\tau(x,a)z_{xa} \tag{3.45}$$

Subject to $$\sum_{a \in B_y} z_{ya} = \sum_{x \in \Lambda} \sum_{a \in B_x} P_{xay} z_{xa}, \quad \forall\, y \in \Lambda \tag{3.46}$$

$$\sum_{x \in \Lambda} \sum_{a \in B_x} \alpha_{xa}^{(k)} \tau(x,a) z_{xa} \le \beta^{(k)}, \quad k = 1,..,K \tag{3.47}$$

$$\sum_{x \in \Lambda} \sum_{a \in B_x} \tau(x,a) z_{xa} = 1 \tag{3.48}$$

$$z_{xa} \ge 0, \; x \in \Lambda \text{ and } a \in B_x. \tag{3.49}$$

In order to understand how the constraints affect the optimal policy it is important to see that the steady state probability of being in state x and choosing action a is given by $\tau(x,a)z_{xa}^*$ for the semi-Markov decision model, where z_{xa}^* is the value of the decision variable taken at the end of the Linear Programming process. Notice here the similarity between (3.47) and (3.48). In (3.47) the probability of being in state x and choosing action a is subject to a user "penalty" of probability (α_{xa}) which is small if the user "likes" taking action a in state x and large otherwise. The optimization process thus tries to avoid the less "appreciated" decisions in every state while making sure that the user constraint (β) is respected. Making the user constraints tighter by decreasing the value of β will thus reduce the optimum value reachable, that is, the utilization of the link as it will be shown with the simulation results in Chapter 4. To understand this reduction notice that the value of the expected time $\tau(x,a)$ in equation 3.47 remains the same as β^k is decreased. Hence z_{xa} will have to decrease to make sure that the tighter user constraint is followed. It may also

be noticed that a decrease in z_{xa} will reduce the value of the objective function (equation 3.45) which implies a reduction in the utilization of the link. Denoting by z^*_{xa} an optimal basic solution to this linear program and letting the set $J_0 = \left\{ x \mid \sum_{a \in B_x} z^*_{xa} > 0 \right\}$, an optimal stationary randomized policy can be given as [19],

$$\pi^*_a(x) = \begin{cases} \dfrac{\tau(x,a) z^*_{xa}}{\sum_{a \in B_x} \tau(x,a) z^*_{xa}}, & a \in B_x \text{ and } x \in J_0, \\ \text{arbitrary}, & \text{otherwise.} \end{cases} \tag{3.50}$$

Here it is pointed out that the ***unichain assumption*** is essential for guaranteeing the existence of an optimal stationary policy [19]. The unichain assumption states that for each stationary policy R, a state r (which may depend on R) exists that can be reached from any other state under policy R [20]. For example in the state diagram of the communication network depicted in Figure 3.2, every state can be reached from any other state and the unichain assumption is thus satisfied. Notice also that when an additional constraint is imposed the optimal policy will in general be randomized [21].

A nice feature of the Linear Programming algorithm for solving semi-Markov decision processes is hence to permit optimization over additional constraints. Let us now apply this optimization technique in order to guarantee a minimum performance for the circuit-switched and packet-switched traffic types. For example suppose that the blocking probability of a class-k circuit-switched call should be kept bellow β^k. This constraint can be incorporated in the Linear Programming formulation as follows:

$$\sum_{x \in \Lambda} \sum_{a \in B_x} Pr\{\text{Class-}k \text{ call blocked in } (x,a)\} \tau(x,a) z_{xa} \leq \beta^k \tag{3.51}$$

where

$$Pr\{\text{Class-}k\text{ call blocked in }(x,a)\} = \begin{Bmatrix} 0, & \text{if } a_k = 1 \\ 1, & \text{if } a_k = 0 \end{Bmatrix} = 1 - a_k. \qquad (3.52)$$

Hence the blocking probability of a call is either 0 (when calls are allowed in the link) or 1 (when calls are not allowed). As before, the decision to accept or reject a call is taken from the state-dependent action space for every class of calls. Another set of constraints will be used to limit the packet delay experienced by a class-k packet-switched call to a maximum as follows:

$$\sum_{x \in \Lambda} \sum_{a \in B_x} Pr\{\text{Packet queueing delay in } (x,a) > t\}\tau(x,a)z_{xa} \le \beta^k \qquad (3.53)$$

where

$$Pr\{\text{Packet queueing delay in } (x,a) > t\} = \begin{Bmatrix} \rho_{xp} e^{-\omega m_{xp}(1-\rho_{xp})t} & m_{xp} > 0 \\ 1 & m_{xp} = 0 \end{Bmatrix} \qquad (3.54)$$

and

$$\rho_{xp} = \sum_{x=C_n+1}^{C_n+P_n} \frac{x_k \eta_k}{m_{xp}\omega}. \qquad (3.55)$$

Notice the major assumption made here for packet-switched traffic requirements: the system must stay in a given state long enough to ensure that the steady state equilibrium delay formula (such as equation 3.54) can apply. The formula used here to obtain the queueing delay for packet-switched traffic (equation 3.54) is the one corresponding to the waiting time for the M/M/1 queueing model where packet arrivals follow a Poisson distribution and the packets service time are exponentially distributed [22]. However it is assumed that the service rate varies with the number of channels available to the packet-switched traffic (m_{xp}). As it will be demonstrated later on in the next section the packet delay formula proposed here (equation 3.54) is precise enough for our purposes.

The Linear Programming algorithm will thus find the best action to be taken in every state in order to satisfy the users requirements based on the time spent between decision epoches and on the probability constraints corresponding to the state and action. Let us illustrate the implementation of a constrained optimal access policy using our previous communication network example developed in Section 3.4. The class-2 circuit-switched calls could require for instance that their blocking probability must be less then 0.2. In order to influence the choice of action to be taken in each state, a new inequality constraint will be added into our previous example, that is, (see equation 3.51)

$$\sum_{x \in \Lambda} \sum_{a \in B_x} (1 - a_2)\, \tau(x,a)\, z_{xa} \;\leq\; 0.2 \tag{3.56}$$

From the semi-Markov decision process parameters computed earlier for the unconstrained problem, this inequality restriction may be written as follows:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1/2 \\ 0 \\ 1/2 \\ 1/3 \\ 1/3 \\ 1 \\ 1/2 \\ 1/2 \end{bmatrix} \cdot \begin{bmatrix} z_{12} \\ z_{13} \\ z_{14} \\ z_{21} \\ z_{22} \\ z_{23} \\ z_{24} \\ z_{31} \\ z_{33} \\ z_{41} \\ z_{51} \\ z_{53} \\ z_{61} \end{bmatrix} \;\leq\; 0.2 \tag{3.57}$$

where "$\cdot$" is the dot product. The objective function and the equality restrictions being the same as in the unconstrained optimization example (see equations 3.41 and 3.42 respec-

tively), the resulting neural network implementation may be given as shown in Figure 3.4. Notice here that the only difference between this neural network and the one given in Figure 3.3 for the unconstrained example is the addition of an inequality restriction amplifier (**g**). The cost to guarantee a certain grade of service to a given user is thus minimal.

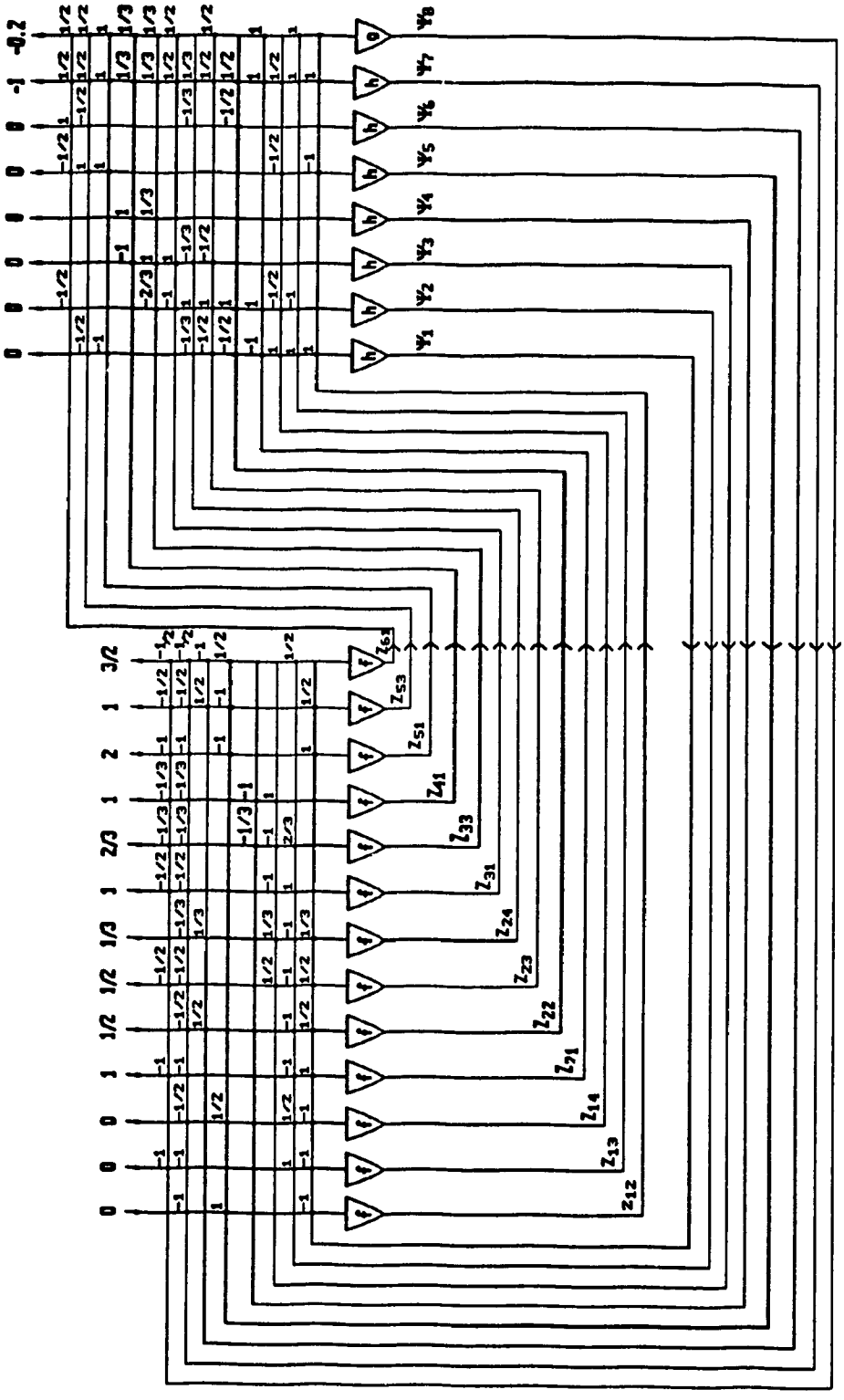

Figure 3.4 Neural Network Corresponding to the Constrained Example Given in this Section.

3.6 Packet Delay Computation Using Queueing Models

An important parameter for the packet-switched users of the resource is the packet queueing delay, as opposed to the blocking probability for circuit-switched users. The packet queueing delay will increase in a higher system load, that is, when new packet-switched calls are accepted in the link or when the number of channels available for packet-switched traffic decreases in the link. In the queueing models developed here, it is assumed that the packet-switched calls are lumped together at the input of the queue and that the packet arrivals follow a Poisson distribution. The queueing models proposed here are tools which can be used to compute the packet queueing delay when the communication system is in a given state x with m_{xp} channels available in the link. Next, three queueing models are presented with increasing complexity.

i) M/M/1.

It will also be assumed that the arriving packets are served on a first come, first served basis; that the transmission line is error free (that is, the same packet is never retransmitted); and that the waiting room is infinite, so that the arrival rate of packets is independent of the number of calls already in the system. In the M/M/1 queue model the further assumptions made are that the service time of packets is exponentially distributed, and that the service rate of the single server varies according to the number of channels available for packet-switched traffic in the link. Clearly, so long as the number of packets in the system is larger than the number of channels available this will be exact. Otherwise it will be an approximation providing a lower bound for the delay. In this case the probability distribution of the time spent by the packets waiting in queue is given as follows [22]:

$$Pr\{\text{Packet queueing delay in } (x,a) > t\} = \rho_{xp} e^{-\omega m_{xp}(1-\rho_{xp})t}, \quad t \geq 0, \tag{3.58}$$

where

$$\rho_{xp} = \sum_{k=C_n+1}^{C_n+P_n} \frac{x_k \eta_k}{m_{xp}\omega}, \tag{3.59}$$

that is, the utilization of any channel occupied by packet-switched traffic. An essential condition for the stability of this queueing model is hence that $\rho_{xp} < 1$. Otherwise the server could not keep up with the arriving packets which would have to wait for an infinite amount of time (or be lost due to buffer overflow in a more realistic system).

ii) M/M/S.

A better approximation is provided by the M/M/S queueing model where more than one server is available to the arriving packets. In the M/M/S queueing model an arriving packet waits in queue only if all servers are occupied. The probability distribution of the queueing time can be stated as follows: [23]

$$Pr\{\text{Packet queueing delay in } (x,a) > t\} = P_0 \left(\frac{(m_{xp}\rho_{xp})^{m_{xp}}}{m_{xp}!(1-\rho_{xp})} - \frac{(m_{xp}\rho_{xp})^{m_{xp}}(1-e^{-\omega m_{xp}(1-\rho_{xp})t})}{m_{xp}!(1-\rho_{xp})} \right) \tag{3.60}$$

where P_0 is the probability of no packet, that is,

$$P_0 = \left(\sum_{j=0}^{m_{xp}-1} \frac{(m_{xp}\rho_{xp})^j}{j!} + \frac{(m_{xp}\rho_{xp})^{m_{xp}}}{m_{xp}!(1-\rho_{xp})} \right)^{-1}. \tag{3.61}$$

iii) M/D/S.

Now if the ATM transport method is used in the communication network then the length of each packet (or cell) should be the same. The service time of a packet is hence deterministic instead of exponentially distributed as it was the case in the two previous queueing models seen (M/M/1 and M/M/S). Unfortunately the equation required to be solved in order to find the delay using M/D/S is quite involved except may be for the case where there is a single server. For the M/D/1 queueing model, the delay formula can be expressed as follows [24].

$$Pr\{\text{Packet queueing delay in } (x,a) > t\} = (1-\rho_{xp}) \sum_{k=C_n+1}^{C_n+P_n} \sum_{i=0}^{\nu} e^{x_k\eta_k(t-ib)} \frac{(-x_k\eta_k(t-ib))^i}{i!} \tag{3.62}$$

if

$$t = \nu b + w, \quad \nu \geq 0, \quad 0 \leq w < b. \tag{3.63}$$

The following figures (Figures 3.5 and 3.6) compare the probability of delay under each of the above queueing models. The probability of delay for both the M/M/1 and M/M/S queueing models have been computed from their respective formulas (equations 3.58 and 3.60). The waiting time probabilities for the M/D/S queueing model was obtained from tables in a queueing delay probability handbook [25]. The first figure (Figures 3.5) gives the waiting time probability results under light loading condition ($\rho_{xp} = 0.6$) for the case where $m_{xp} = 5$ servers. The next figure (Figures 3.6) presents the results under heavy loading ($\rho_{xp} = 0.9$) condition.

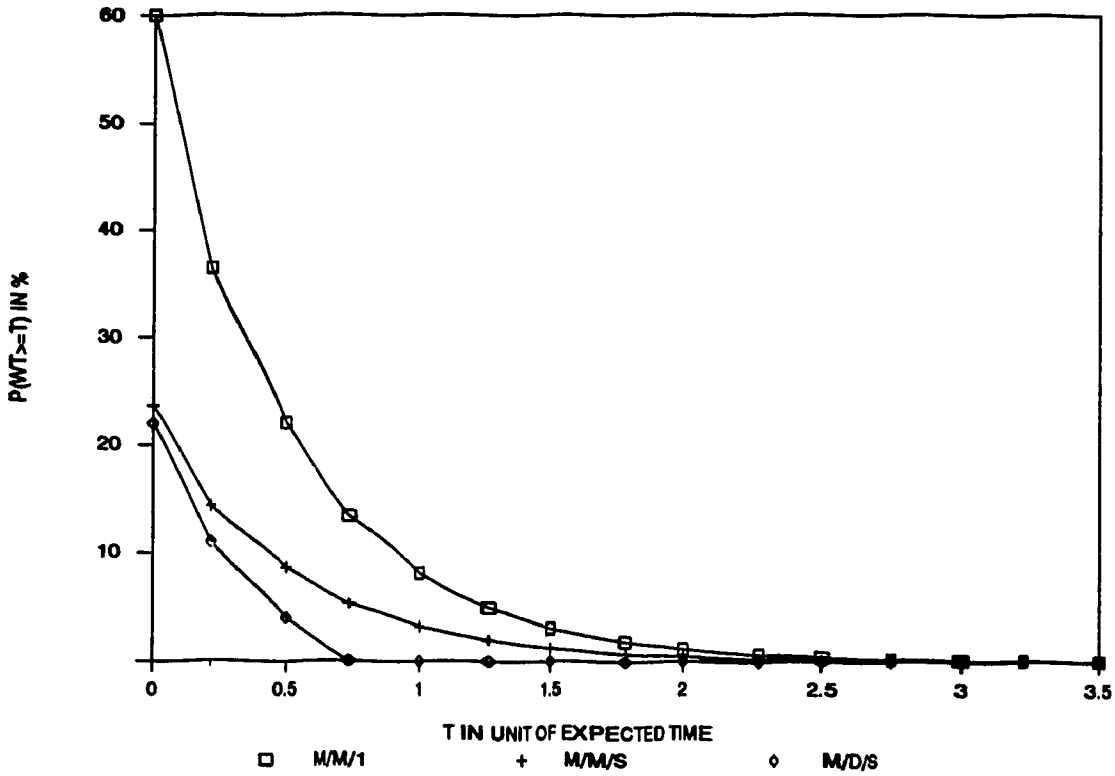

Figure 3.5 Five Servers with Traffic Intensity of 0.6.

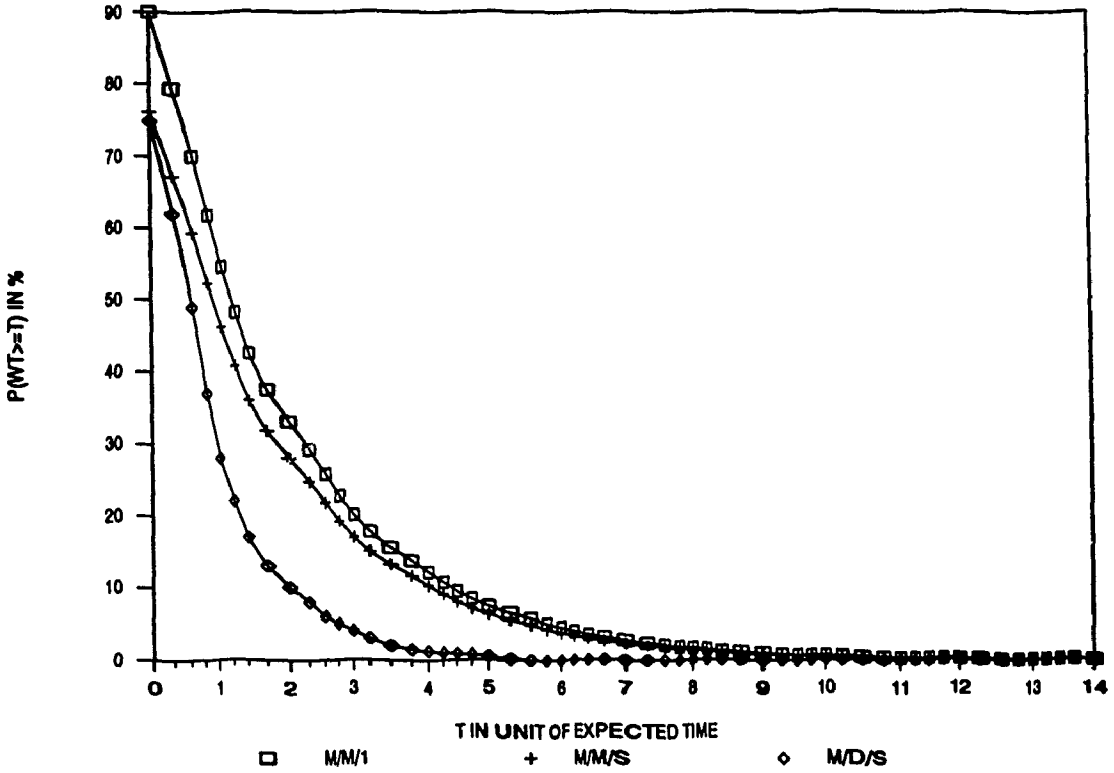

Figure 3.6 Five Servers with Traffic Intensity of 0.9.

As may be seen under heavy loading all the curves approach to each other. Further, the M/M/1 provides a lower bound, the system cannot be better than this result. Finally due to its simplicity, M/M/1 results was used in the neural network implementation. Clearly, the application of the M/M/1, M/M/S, or M/D/S delay formulas assume that the queueing system has reached to the steady-state. However in the present problem, the service rate varies with the number of channels available to the packet-switched traffic, and the packet arrival rate varies with the number of packet-switched users in the system. Thus each time

either of these two quantities change, the queueing system will go through a transient state before it reaches to the steady-state. Let us explain this in more detail, by studying the mean arrival and service rates. Let

$x(t)$ = Number of calls connected at time t

$\eta(t)$ = Arrival rate of packets per call per second at time t

$\omega(t)$ = Service rate of packets per channel per second at time t

$S(t)$ = Number of channels available for packet-switched traffic at time t

Assuming $\omega(t) = \omega$, a constant value; $x(t)$ and $S(t)$ are step functions with constant value during a given state; and $\eta(t)$ which is the mean arrival rate of packets per call following a Poisson distribution. An example of this system is shown in Figure 3.7.

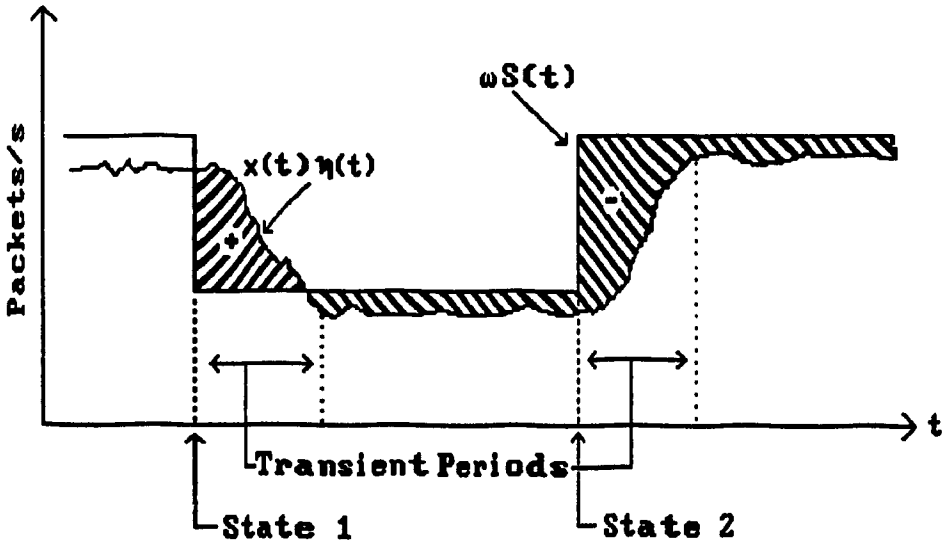

Figure 3.7 Fluid Approximation.

A transient period is shown at the beginning of each state. During the transient period at the beginning of State 1 in Figure 3.7 a backlog is created because the arrival rate of packets

is larger than the service rate (the cross-hatched area labeled "+" corresponds in fact to the "deficit" between service rate and arrival rate and it therefore represents the total amount of packets backlogging in the system). On the other hand, once the arrival rate drops below the departure rate the deficit can be made up with the excess capacity shown as the cross-hatched area labeled "-". Only when the total negative area equals the total positive area will the backlog drop to zero. If the non-rush hour value for $x(t)\eta(t)$ is only slightly less than the departure rate then we can imagine that it will take quite a while to make up for the deficit. Conversely if the rate of accumulation of negative area is large compared to that for the positive area then the backlog will fall off rather quickly. This situation is depicted in the transient period at the beginning of State 2 in Figure 3.7 [26].

In the following it will be assumed that the queueing system will remain in a state long enough to reach the steady-state, so that the stationary delay distributions apply. This is known as the quasi-static approximation [27].

CHAPTER 4

NEURAL NETWORK SIMULATION RESULTS

4.1 Introduction

In this chapter we would like to show how the optimal access policy performs under different communication systems. The main characteristics of the implementation of an optimal access controller using a Linear Programming neural network are first described. The simulation process required to determine the behavior of a Linear Programming neural network is then presented. Finally the simulation results obtained for both the unconstrained and the constrained optimal access policies are given.

4.2 Optimal Access Controller Implementation

The main characteristics of an optimal access controller using a Linear Programming neural network are depicted in Figure 4.1. The access controller receives the call requests and issues the accept/reject decision. The optimization task required to find the best action corresponding to the present state of the link is processed in the Linear Programming neural network which is used much like a coprocessor. When a call request arrives the main processor in the access controller computes the weights required by the Linear Programming neural network from the communication link parameters. These parameters include the arrival rate for every class of calls, their call service time, and their bandwidth requirement in the case of circuit-switched calls or their packet arrival rate in the case of packet-switched calls. The computed weights are then "programmed" into the Linear Programming neural network corresponding to resistances. At this point the optimization process can start. Once

the neural network outputs are stable the resulting values are transmitted back to the main processor which will compute the best action to be taken in the present state of the link upon arrival of a call. The access controller is now ready to issue the accept/reject decision corresponding to the present state of the communication link for every class of calls.

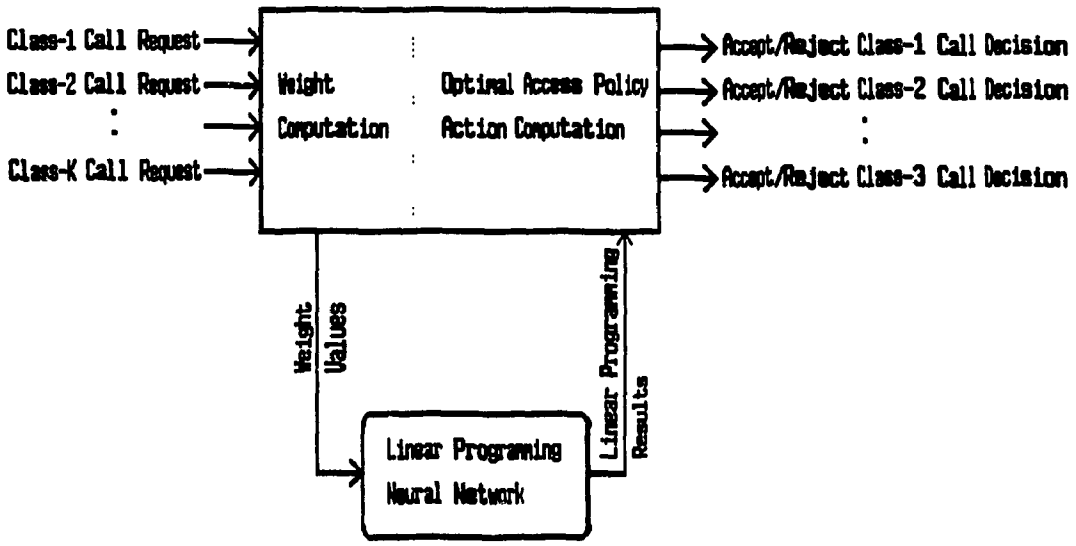

Figure 4.1 Schematic Diagram of the Optimal Access Controller.

The accept/reject decision remains valid as long as the state of the link does not change. The state of the link (and the call arrival rate) can be computed from the call requests themselves. Clearly at the time a call makes its request, an event occurs and the accept/reject decision is already available for that class of calls since this action corresponds to the present state of the link. If the call request is accepted then a decision epoch occurs and the state changes with one more call in the link for that class of calls. Similarly every time a call transmission ends an event occurs and the state space should change with one less call in the link for that class of calls. The main processor can compute the expected departure time of a call from the average service duration usually taken by that class of calls. This com-

putation will unfortunately set the decision epoch at a time which may not be the same as the call departure event as it should be since a call departure will obviously change the physical state of the system. Some feedback information from the link such as the exact service time of a call would hence be very helpful to further improve the performance of the optimal access controller.

From the analysis in Chapter 3, the computed decision variables will be valid so long as the call parameters do not change. When λ_k or μ_k change than the results are not valid anymore and we have to run the system again. Thus the assumption will be that the neural network will be activated whenever these parameters changes. This in a way invalidates some of the reasons why neural network was being used in the first place when these parameters are slowly changing. On the other hand note that some studies like the Japanese work [11] does assume that these parameters would change all the time in an integrated communication network.

The steps required to compute the value of the Linear Programming neural network weights are given in the following flowchart (see Figure 4.2). These steps correspond to the examples given in Chapter 3 (see Sections 3.4 and 3.5). The computation of the Linear Programming neural network weights requires a fair amount of memory. Table 4.1 presents an overview of the amount of memory required in the computation of the different parameters. In order to compute the transition probabilities the following observation was made. To store every component of the transition probabilities P_{xay} a three-dimensional matrix of $|\Lambda| \times |\Lambda| \times 2^K$ should be required. Fortunately the transition from one state is

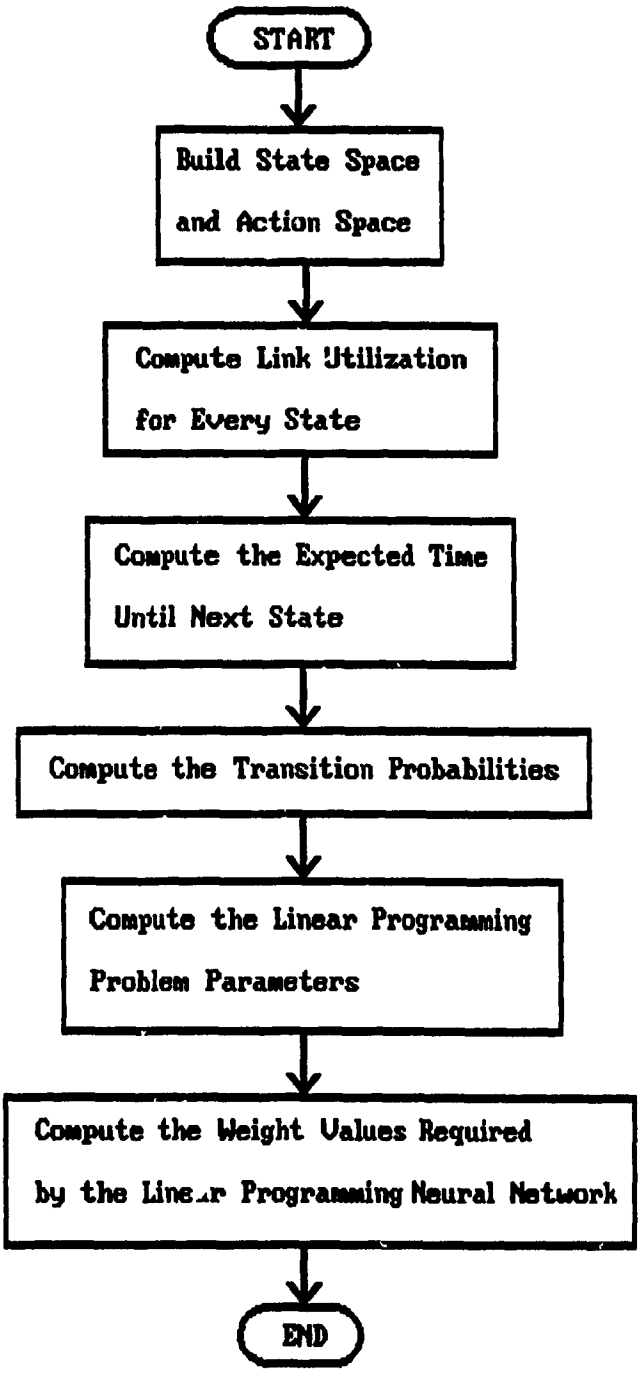

Figure 4.2 Flowchart of the Weight Computation Steps.

possible only to the neighboring states, that is, only if the Hamming distance between two state descriptors is equal to one, since simultaneous call arrivals (or departures) are not allowed in the system (see the state diagram in Figure 3.2). For instance if the present state is (1,1,1) then at most $2(K) = 6$ states are reachable: (0,1,1), (1,0,1), (1,1,0), (2,1,1), (1,2,1), (1,1,2). The memory space required to hold the transition probability coefficients may hence be reduced to a maximum of $|\Lambda| \times 2K \times 2^K$ significant values. From Table 4.1 the total amount of memory required can be roughly stated as

$$\text{Total Memory Requirements} \leq |\Lambda| \, (1 + 2^K + K2^{K+1}). \tag{4.1}$$

Table 4.1 Memory Requirements for the Weights Computation.

Computation Parameters	Memory Requirements
State Space Λ	$\|\Lambda\|^*$
Overall utilization of the link $r(x)$	$\|\Lambda\|$
State-dependent actions B_x	$2^K \cdot \|\Lambda\|$
Expected time until next state $\tau(x,a)$	$2^K \cdot \|\Lambda\|$
Transition probabilities P_{xay}	$2K \cdot 2^K \cdot \|\Lambda\|$

* $|\Lambda|$ refers to the number of elements in Λ.

4.3 Linear Programming Simulation

At present time the hardware implementation of a Linear Programming neural network of the size required to solve an optimal access control problem is not practical, and for the time being a simulation program will be used instead. The speed at which one can find the optimal access policy depends on the speed at which the Linear Programming problem

corresponding to the communication system can be solved. In the physical implementation of the Linear Programming neural network, the value of all variable amplifiers outputs as well as the value of all restriction amplifiers outputs are computed simultaneously (in parallel). It is hard to know how long it would take for the neural network outputs to reach a stable state when the optimization process is computed. With todays high speed VLSI technology a solution should however be provided easily on the order of a few milliseconds. This is the kind of speed which is required by the access controller to work efficiently in a multimedia environment. In a uniprocessor mainframe computer system every instruction is executed in sequence so that the value of all amplifiers output has to be computed one after the other. For this reason the execution time of the simulation process required to find the Linear Programming solution for an optimal access policy depends heavily on the size of the problem itself. This problem size difficulty would not appear if in the physical implementation of the Linear Programming, neural network was used since the computation is made in parallel in that case.

In order to simulate the action of a Linear Programming neural network, one may start from its circuit equation (equation 2.10). If we let $C=1$ for convenience then the circuit equation to solve the maximization Linear Programming problem may be expressed as follows:

$$\frac{dV_k}{dt} = A_k - \sum_{i=1}^{I} D_{i,k} g(\vec{D_i} \cdot \vec{V} - B_i) - \sum_{e=I+1}^{E+I} D_{e,k} h(\vec{D_e} \cdot \vec{V} - B_e), \quad k = 1,2,..,N, \tag{4.3}$$

so that we can compute the value of the variable amplifiers iteratively using the first order difference as:

$$V_k^{t+1} = \max\left(V_k^t + \left(\frac{dV_k^t}{dt}\right)\Delta t, 0\right). \tag{4.4}$$

The flowchart in Figure 4.3 shows the steps required to simulate the action of the Linear Programming neural network. Notice the first step where the output values of the variable amplifiers are initialized to zero (initial state of the Linear Programming neural network). Once the values of the variable amplifier outputs are stable the simulation process stops since the neural network is in its final state. Clearly the simulation precess will compute the value of the momentum function, that is,

$$M = -(\vec{A} \cdot \vec{V}) + \sum_{i=1}^{I} G(\vec{D_i} \cdot \vec{V} - B_i) + \sum_{e=I+1}^{I+E} H(\vec{D_e} \cdot \vec{V} - B_e), \tag{4.5}$$

after each iteration to see if its value changed from the last iteration. When this value does not change for a given number of iterations the optimization process stops. At this point the output values of the variable amplifiers are assumed to be stable and the maximum possible overall utilization of the link may be obtained from the objective function itself, that is,

$$\sum_{x \in \Lambda} \sum_{a \in B_x} r(x,a)\pi(x,a)z_{xa} \tag{4.6}$$

The plot in Figure 4.4 shows the value of the momentum function as the time goes on. At the beginning the value of the momentum function is very high since the equality restrictions are all violated. When the energy stabilizes to a minimum value (towards infinity in time) the neural network solution will ultimately satisfy every restriction. At this point the value of the momentum function should be equal to the negative of the overall utilization of the link (the value of the objective function) since all the equality restrictions as well as

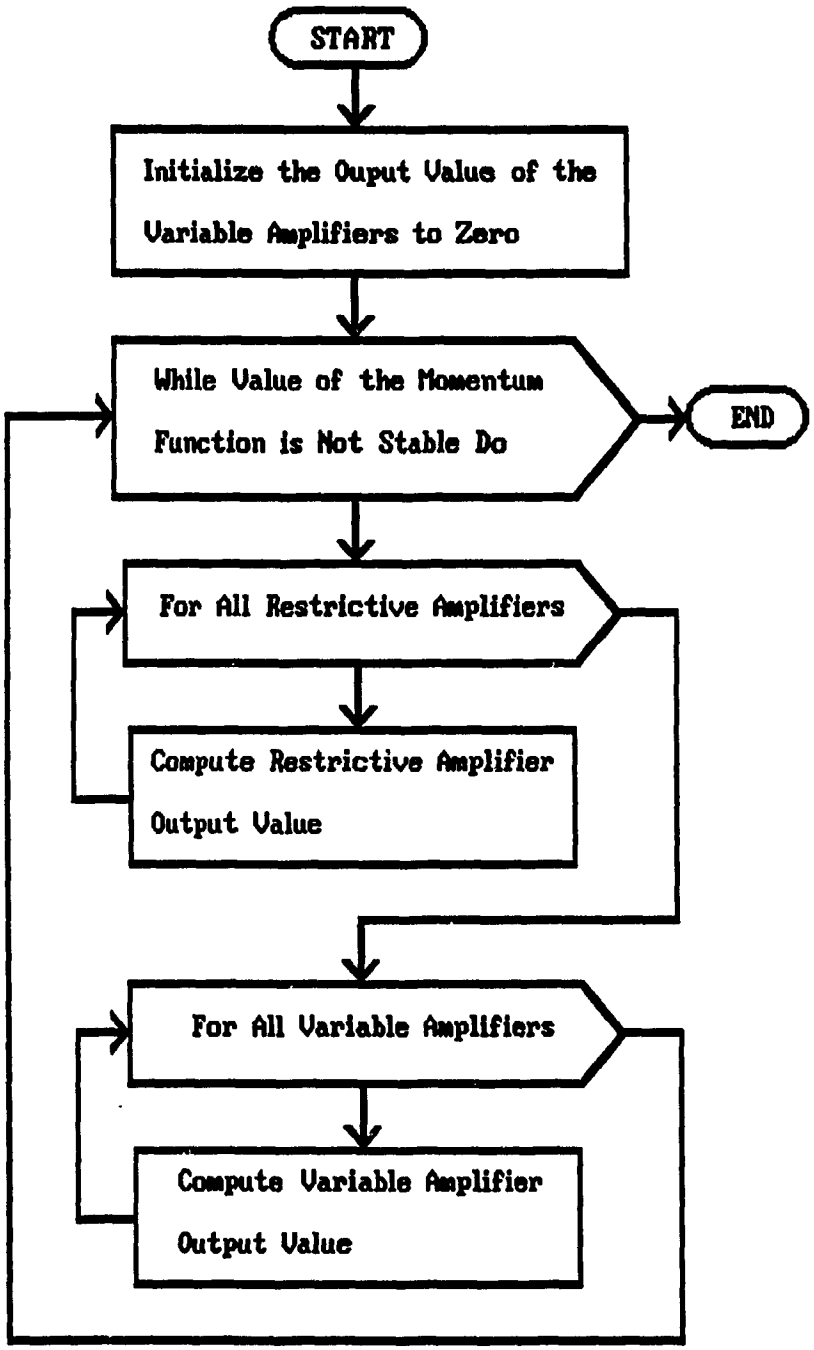

Figure 4.3 Flowchart of the Neural Network Simulation Program.

all the inequality restrictions are satisfied. To see it one may compare the momentum function (equation 4.5) with the objective function which may be written as

$$\Pi = \vec{A} \cdot \vec{V}, \tag{4.7}$$

that is,

$$M = -\Pi \tag{4.8}$$

if

$$\sum_{i=1}^{I} G(\vec{D_i} \cdot \vec{V} - B_i) + \sum_{e=I+1}^{I+E} H(\vec{D_e} \cdot \vec{V} - B_e) = 0. \tag{4.9}$$

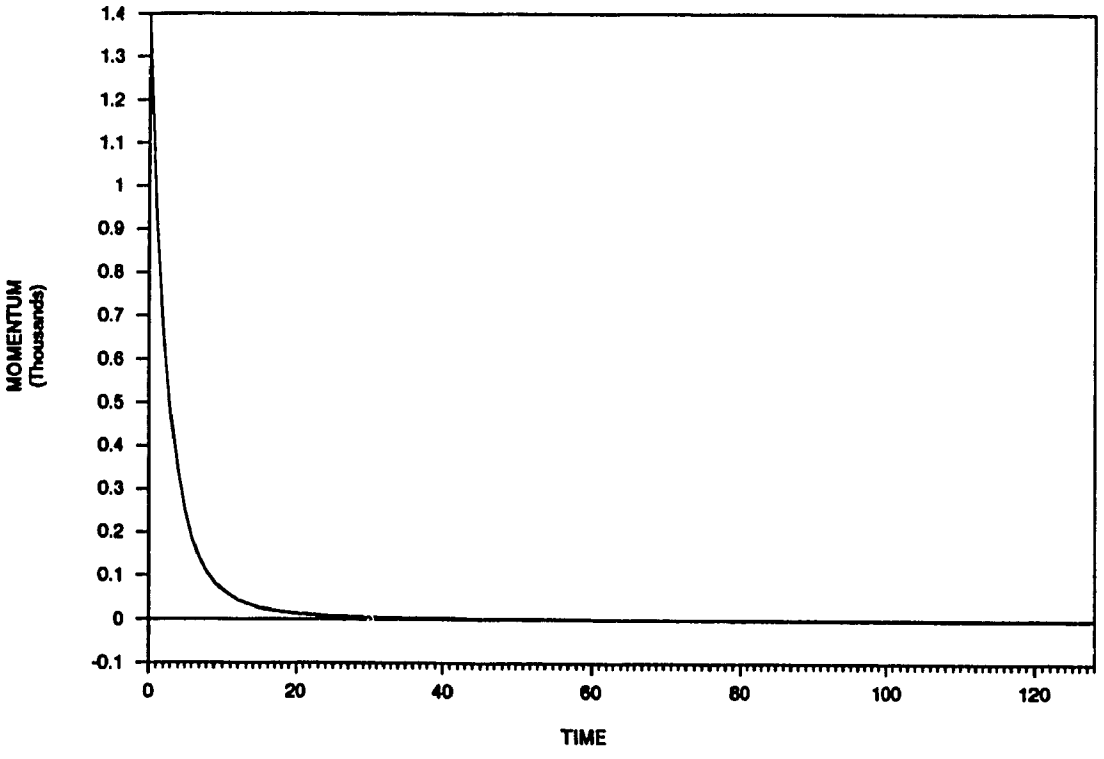

Figure 4.4 Momentum of the Neural Network.

4.4 Simulation Results

For the first simulation example let us assume that the optimal access controller is used to determine the access for two classes of circuit-switched traffic in a link of four channels ($m_n = 4$). Class-1 calls take up one channel and class-2 calls, two channels (b_1, b_2) = (1,2). The service time of these calls is assume to be one second for both classes ($1/\mu_1$, $1/\mu_2$) = (1,1). Figure 4.5 shows the state space diagram for such a communication system.

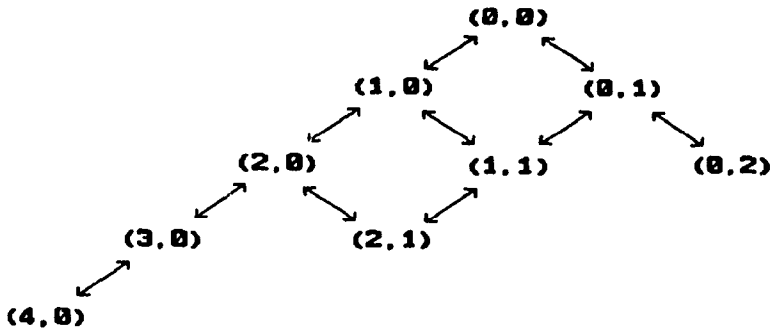

Figure 4.5 State Space Diagram for the 4-Channels Example.

We are now ready to compute the optimal utilization of the system from the Linear Programming neural network simulation results. In order to be consistent with the literature though [15], we will make sure that the arrival rate ratio λ_1/λ_2 is varied so that

$$\frac{b_1\lambda_1}{m_n\mu_1} + \frac{b_2\lambda_2}{m_n\mu_2} = 1.5. \tag{4.10}$$

The curve for $\lambda_3 = 0$ in Figure 4.6 shows the utilization of the four channels link for these two circuit-switched classes of calls when the optimal access policy is used. As it can be seen from Figure 4.6 when one class of calls is predominant the utilization of the link

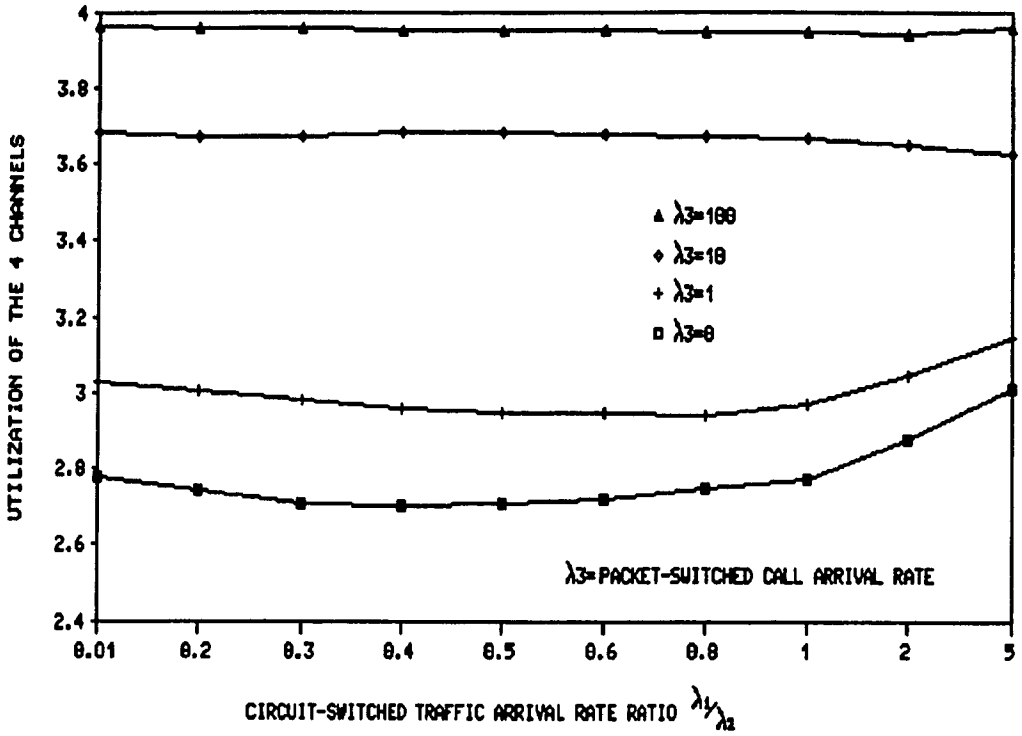

Figure 4.6 Utilization in a 4-Channels Link.

(Call parameters are given in Table 4.2)

Table 4.2 Utilization in a 4-Channels Link.

Network Parameters	Value		
	Class k=1	Class k=2	Class k=3
b_k	1	2	-
$1/\mu_k$	1	1	1
ω	-	-	125
η_k	-	-	250

increases. Clearly if the arrival rate of one class of calls is much higher than the other class then the access controller following an unconstrained optimal access policy will have a tendency to accept only that class of calls.

We would like now to introduce packet-switched traffic into the communication system in order to increase the utilization of the link. A very important ratio for packet-switched traffic is the arrival rate of packets per call over the service rate of packets per channel ratio. The lower this ratio is the higher the number of states in the state space. As it was shown in Chapter 3 the state space may be derived from the following formula:

$$\Lambda = \left\{ x : x \geq 0; \sum_{k=1}^{C_n} x_k b_k + \sum_{k=C_n+1}^{C_n+P_n} x_k \eta_k/\omega \leq m_n \right\}. \qquad (4.11)$$

It can be seen from this formula (equation 4.11) that the number of packets-switched calls allowed simultaneously on the link (as well as the number of elements in the state space) increases as the η_k/ω ratio decrease assuming everything else remains the same. Up to two packet-switched calls could hence be allowed simultaneously in a four channels link in order to respect the bandwidth limitation of the link. As it can be seen from Figure 4.6 the utilization of the link increases when the packet-switched traffic is allowed. The utilization of the link increases to a maximum as the packet-switched calls arrival rate is raised towards infinity (see curve $\lambda_3 = 100$ for instance). (Note here that a table showing the communication system parameters follows every simulation result given.)

Let us now assume a more complex communication system where the number of channels in the link is 24 (which corresponds to the T1 transmission standard). Two circuit-switched classes of calls are allowed in the link with a service time of 1 second each $(1/\mu_1, 1/\mu_2) = (1,1)$. The class-1 calls require 1 channel only while 6 channels are needed

for class-2 calls transmission. Figure 4.7 shows the utilization of the link when the unconstrained optimal access policy is used in this case (curve $\lambda_3 = 0$). Also on Figure 4.7 the utilization of the link under the *complete sharing* access policy is demonstrated [15] (see curve Complete Sharing). As shown in Figure 4.7, for small and large values of the arrival rate ratio there is little difference in performance between the optimal policy and complete sharing; for moderate values of λ_1/λ_2 however the improvement can be close to 10 percent in this example. Figure 4.7 also shows the influence of the packet-switched traffic on the utilization of the link. In order to reduce the cardinality of the state space (and hence the complexity of the system) the arrival rate of packets per call over service rate of packets per channel ratio 12 was chosen. In such a system an ATM packet-switched call would bring an arrival rate of 1500 cells per second. Here again the utilization of the link is increased as the arrival rate of the packet-switched calls is increased.

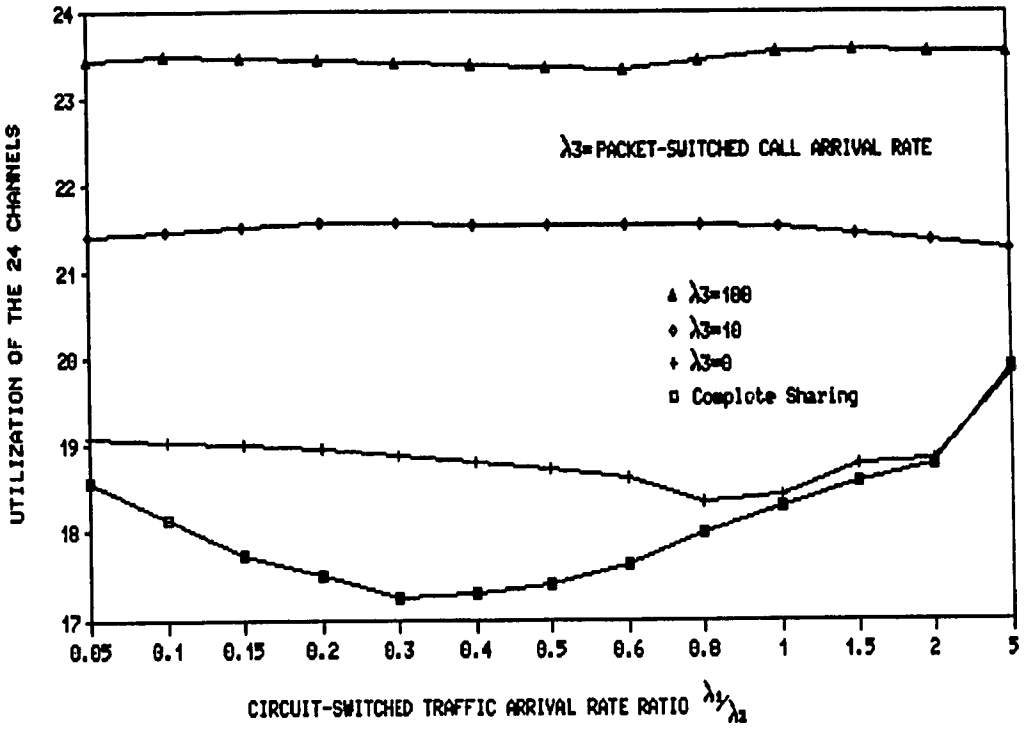

Figure 4.7 Utilization in a 24-Channels link.

(Call parameters are given in Table 4.3)

Table 4.3 Utilization in a 24-Channels Link.

Network Parameters	Value		
	Class k=1	Class k=2	Class k=3
b_k	1	6	-
$1/\mu_k$	1	1	1
ω	-	-	125
η_k	-	-	1500

In the third situation let us assume a link of four channels where there are only two classes of calls allowed; one circuit-switched class and one packet-switched class. In this

case however the circuit-switched traffic has a service time of 10 seconds while the packet-switched traffic has a service time of 1 second (a more realistic situation), that is, $(1/\mu_1, 1/\mu_2) = (10,1)$. The bandwidth taken by a circuit-switched call is 1 channel and the ratio of the arrival rate of packets to the service rate of packets per channel is 2. Figure 4.8 shows the utilization of the link under the unconstrained optimal access policies as the arrival rate of circuit-switched calls is varied. As it can be seen from Figure 4.8 the utilization of the link increases very fast as the circuit-switched traffic is increased because the service time of these calls is 10 seconds.

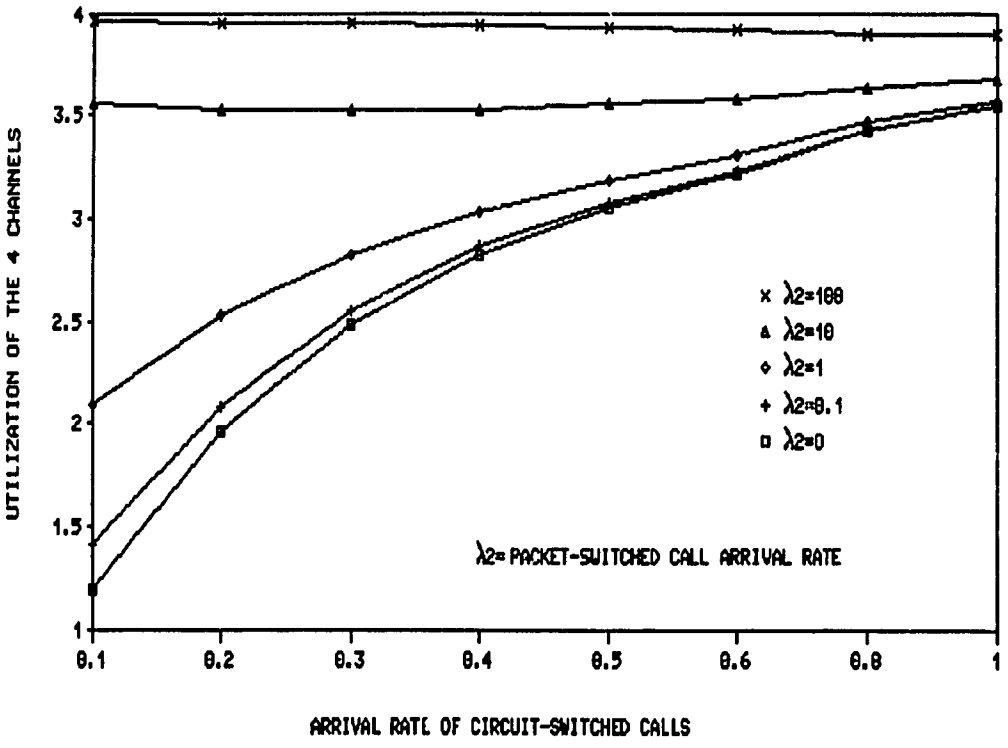

Figure 4.8 Utilization in a Realistic Situation.

(Call parameters are given in Table 4.4)

Table 4.4 Utilization in a Realistic Situation.

Network Parameters	Value	
	Class k=1	Class k=2
b_k	1	-
$1/\mu_k$	10	1
ω	-	125
η_k	-	250

In the simulation results seen so far the utilization of the link was optimized without taking the user specifications into account. In Figure 4.6 for example it can be seen that by allowing more packet-switched calls in the system an overall increase in the performance of the communication network may be obtained. But what happens when user demands are taken into account? Clearly the utilization may be maximized by flooding the link with a given class of calls. In order to be able to guarantee a certain level of service quality to other classes of calls however additional constraints must be added to the system. Let us assume for now that only two circuit-switched classes of calls are allowed to the link. Each class of calls will be able to specify its blocking probability, that is, the probability that a call will be blocked should be less than what is specified by the given class. Figure 4.9 shows the utilization of a link following the addition of these constraints. The situation given in Figure 4.9 is the case where the link has 4 channels and where the bandwidth required from each class of call is $(b_1, b_2) = (1,2)$. The results are plotted as a function of maximum blocking probability for class-1 or class-2 calls (B). Clearly, because of fixed arrival rates of this example, this probability can be specified only for one class. As it can be seen from Figure 4.9, when the maximum blocking probability specified is very loose

(towards $\beta = 1$) the corresponding utilization is very high and tends to the value of the unconstrained utilization of the link as shown in Figure 4.6 when the arrival rate ratio $\lambda_1/\lambda_2 = 1$. As the user specifications get tighter the utilization decreases since more channel capacity is needed to guarantee the user requests.

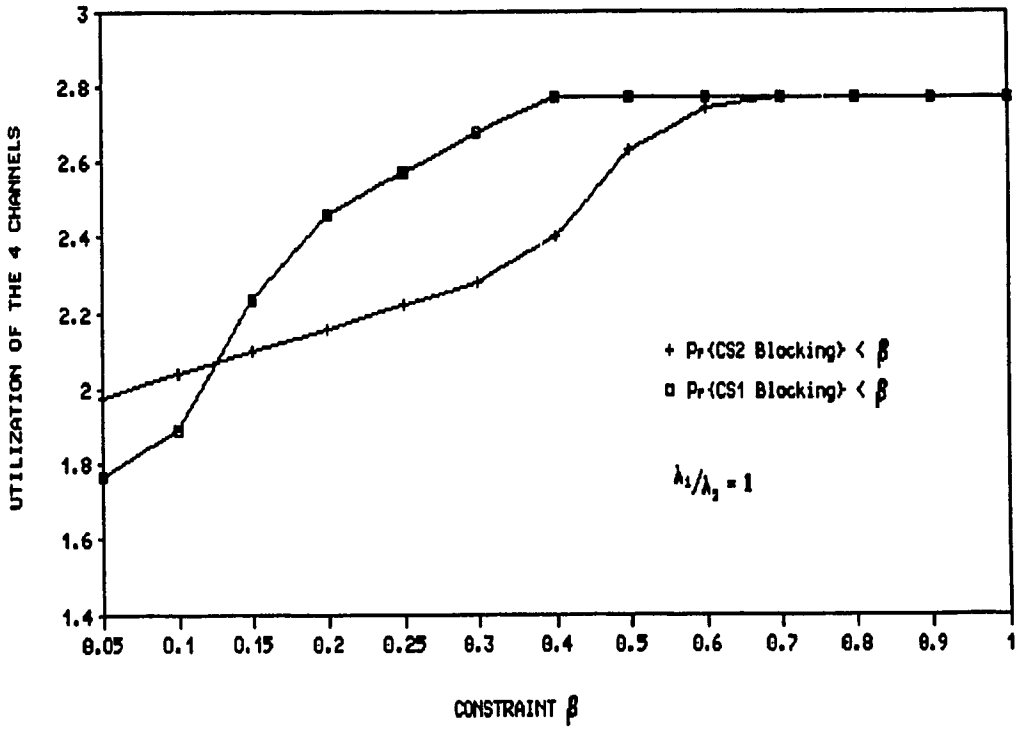

Figure 4.9 Utilization under Constraints with CS Traffic Only.

(Call parameters are given in Table 4.5)

Table 4.5 Utilization under Constraints with CS Traffic.

Network Parameters	Value	
	Class k=1	Class k=2
b_k	1	2
$1/\mu_k$	1	1
λ_k	2	2

Figure 4.10 presents the case where a packet-switched class of calls is also allowed in the link. The packet-switched type of call users will request a guarantee on the probability of their maximum packet queueing delay. In the example shown in Figure 4.10 a packet-switched user will make a request to ensure that the probability of having a packet delay greater than 0.01 second must be less than, say, β (see curve "Pr{PS Delay > .01} < β"). Here again it may be noticed that the utilization of the network is the same as it was for the unconstrained situation when the constraint is loose (see Figure 4.6 in the case when the arrival rate ratio $\lambda_1/\lambda_2 = 1$ and when the arrival rate of packet-switched calls is $\lambda_3 = 100$) while the utilization decreases as the specification set by the user gets tighter. Notice here that the probability of delay was computed from the modified M/M/1 queueing model, that is,

$$Pr\{\text{Packet queueing time in } (x,a) > t\} = \begin{cases} \rho_{xp} e^{-\omega m_{xp}(1-\rho_{xp})t} & \text{for } t \geq 0, m_{xp} > 0 \\ 1 & \text{for } t \geq 0, m_{xp} = 0 \end{cases} \quad (4.13)$$

Figure 4.10 also demonstrates that even though the link is flooded with packet-switched traffic calls, the utilization of the link will decrease to guarantee a certain grade of service to a circuit-switched user.

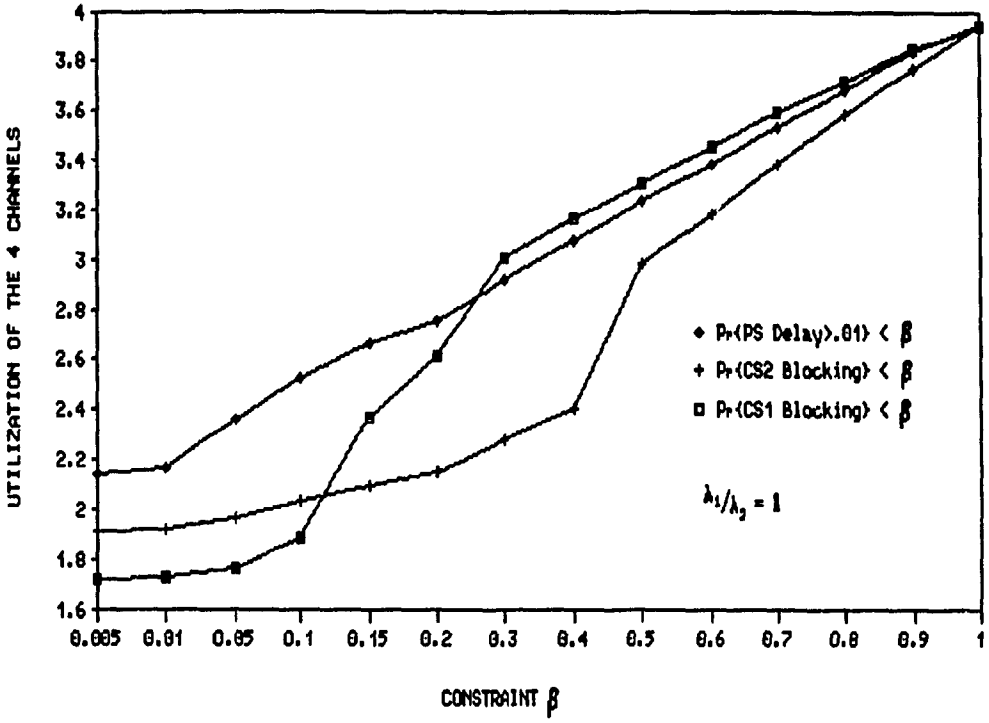

Figure 4.10 Utilization under Constraints with PS Traffic.

(Call parameters are given in Table 4.6)

Table 4.6 Utilization under Constraints with PS Traffic.

Network Parameters	Value		
	Class k=1	Class k=2	Class k=3
b_k	1	2	-
$1/\mu_k$	1	1	1
λ_k	2	2	100
ω	-	-	125
η_k	-	-	250

CHAPTER 5

CONCLUSION

The main objective of this thesis was to provide an efficient mean to control the call access in a multimedia communication system. An efficient controller should optimize the utilization of the communication link bandwidth while guaranteeing the user requirements in the network. The technique determining the optimal access policy corresponding to a communication system where both circuit-switched traffic and packet-switched traffic types are integrated was developed in Chapter 3. This technique is very flexible in the sense that every class of user is allowed to specify its requirements. The optimal policy was modelled as a semi-Markov decision process which was then mapped to a Linear Programming problem.

The computational task required to find an optimal access policy expressed as a Linear Programming problem may be quite intensive on a conventional uniprocessor computer system. And due to the short time delay allowed to make the decision to accept or reject a call, another approach had to be found to solve this processing time problem. Chapter 2 describes the Linear Programming neural network which can effectively find a solution to an optimization problem. A neural network can be described as a highly interconnected network of simple analog processors (the neurons) which can collectively compute difficult optimization problems. The processing time problem is thus resolved since the computation in a neural network is distributed among all the neurons working in parallel. Moreover the processing time will be independent of the problem size to be solved when a neural network is used.

As demonstrated from the simulation results found in Chapter 4, the optimal access policy provides superior link utilization compared to less complex access schemes such as Complete Sharing. The optimal access policy efficiently integrates both packet-switched and circuit-switched traffic types with the possibility of allowing a certain grade of service for every class of calls. For circuit-switched traffic this involves specification of maximum blocking probability, while for packet-switched traffic, maximum probability that the packet delay exceeds a limit.

Suggestions for further research in this area could include the study of a physical optimal access controller implementation. A simpler system would make use of a probabilistic model while a more evolved model could benefit from the use of certain feedback information such as the exact duration of a call. Such information would be very helpful to provide a more efficient utilization of the link.

To conclude, the main objective that we wished to demonstrate in this thesis work was that close control of the access mechanism in a multimedia communication network to achieve an optimal performance out of a link while guaranteeing all users requirements is possible using techniques which can effectively be implemented using todays technology.

REFERENCES

[1] B. Kraimeche and M. Schwartz, "Analysis of Traffic Access Control Strategies in Integrated Service Networks," IEEE Transactions on Communications, vol. COM-33, October 1985, p. 1086.

[2] B. Kraimeche and M. Schwartz, "Bandwidth Allocation Strategies in Wide-Band Integrated Networks," IEEE Journal on Selected Areas in Communications, vol. SAC-33, September 1986, p. 870.

[3] J.F. Hayes and G. Stamatelos, "An Integrated System for Video, Voice and Data Communications," Proceedings of the 15th Biennial Symposium on Communications, Kingston(Ontario), 1990, p. 21.

[4] G.M. Woodruff and R. Hositpaiboon, "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance," IEEE Journal on Selected Areas in Communication, vol. 8, April 1990, p. 437.

[5] I.S. Gopal and T.E. Stern, "Optimal Call Blocking Policies in an Integrated Services Environment," Conference on Information Science System, The Johns Hopkins University, 1983, pp. 383-388.

[6] K.W. Ross and D.H.K. Tsang, "Optimal Circuit Access Policies in an ISDN Environment: A Markov Decision Approach," IEEE Transactions on Communications, vol. 37, September 1989.

[7] K.W. Ross and D.H.K. Tsang, "Optimal Circuit Access Policies in an ISDN Environment: A Markov Decision Approach," IEEE Transactions on Communications, vol. 37, September 1989, p. 938.

[8] D.W. Tank and J.J. Hopfield, "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," IEEE Transactions on Circuits and Systems, vol. CAS-37, May 1986, p. 533.

[9] H.T. Nguyen and M.M. Ali, "A Neural Network Controller for a High-Speed Packet Switch," Proceedings of the 15th Biennial Symposium on Communications, Kingston(Ontario), 1990, p. 140.

[10] F. Kamoun, *A Neural Network Approach to Routing and Flow Allocation Problems in Communications Networks*, MASC Thesis, Concordia University, November 1990.

[11] A. Hiramatsu, "ATM Communications Network Control by Neural Network," IEEE Transaction on Automotive Control, September 1989.

[12] I.B. Pyne, "Linear Programming on an Electronic Analogue Computer," Transactions AIEE, Part 1(Communications and Electronics), vol. 75, 1956, pp. 139-140.

[13] D.W. Tank and J.J. Hopfield, "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," IEEE Transactions on Circuits and Systems, vol. CAS-37, May 1986, p. 539.

[14] K.W. Ross and D.H.K. Tsang, "Optimal Circuit Access Policies in an ISDN Environment: A Markov Decision Approach," IEEE Transactions on Communications, vol. 37, September 1989, p. 935.

[15] K.W. Ross and D.H.K. Tsang, "Optimal Circuit Access Policies in an ISDN Environment: A Markov Decision Approach," IEEE Transactions on Communications, vol. 37, September 1989, p. 936.

[16] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach.* New York: Wiley, 1986, pp. 181-182.

[17] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach.* New York: Wiley, 1986, p. 185.

[18] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach.* New York: Wiley, 1986, pp. 181, 211.

[19] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach.* New York: Wiley, 1986, p. 186.

[20] H.C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach.* New York: Wiley, 1986, p. 165.

[21] K.W. Ross and D.H.K. Tsang, "Optimal Circuit Access Policies in an ISDN Environment: A Markov Decision Approach," IEEE Transactions on Communications, vol. 37, September 1989, p. 937.

[22] J.F. Hayes, *Modeling and Analysis of Computer Communications Networks.* New York: Plenum, 1984, p.72.

[23] J.F. Hayes, *Modeling and Analysis of Computer Communications Networks.* New York: Plenum, 1984, p. 76.

[24] N.U. Prabhu, *Queues and Inventories: A Study of Their Basic Stochastic Processes*. New York: Wiley, [QA 273 P787 C.2], 1965, p. 115.

[25] F.S. Hiller, O.S. Yu, *Queueing Tables and Graphs*, New York: North Holland, 1981, [T 57.9 H54 C.1], pp. 96-133.

[26] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*. New York, Wiley, 1976, p. 61.

[27] B. Kraimeche and M. Schwartz, "Analysis of Traffic Access Control Strategies in Integrated Service Networks," IEEE Transactions on Communications, vol. COM-33, October 1985.