Abstract

FAST PACKET SWITCHING FOR BISDN WITH NEURAL NET CONTROL

Ajit K. Chaudhuri, MSEE, New Jersey Institute of Technonlgy
Thesis Advisor: Prof. Irving Wang

The concept of integrated network where both the voice packet and data packet are delt with, started taking shape since 1970. Later, Users' damand for communication of data, voice, file, facsimile, image, videotext, videophone, videomovie has also been felt. All these services need bit rate of transmision in the order of 10, even 100 Mbits. An economical way to transmit this higher bit rate, with flexibility of bandwidth, is to treat voice/video in the form of packets and transmit those packets in real time. Studies of this kind of communication is carried out in Broadband ISDN (BISDN). At this high rate of transmission switching speed should be in the order of a few microseconds. When the number of input and output ports is very high, usual logic circuit and polling system can not deal with this high rate. Controlling time of the switch must be less than the transmission time of a packet. A Neural Net consists of densely connected simple computation elements–called neurons. To measure the performance of a fast packet switching switch, we used a Hopfield net with 'one-winner-take-all' algorithm to control switching of a crossbar switch. We measured the throughput of the switch and delay of both the data and voice packets at different load with different packet length and with different voice/data ratio. We compared our result with the polling system and found our result was better.

Fast Packet Switching For B-ISDN With Neural Net Control

by

Ajit K. Chaudhuri

Thesis submitted to the Faculty of the Graduate School of the New Jersey Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

1991

APPROVAL SHEET

Title of Thesis: **Fast Packet Switching For B-ISDN With Neural Net Control**

Name of Candidate: Ajit K Chaudhuri
Master of Science in Electrical Engineering, 1991

Thesis & Abstract Approved
by the Examining Committee:

Dr. Irving Wang, Advisor — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. John Carpinelli — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. Edwin Hou — Date
Assistant Professor
Department of Electrical and Computer Engineering

New Jersey Institute of Technology, Newark, New Jersey.

VITA

Name: Ajit K Chaudhuri

Permanent address:

Degree and date to be conferred: M.S.E.E., 1991.

Date of birth:

Place of birth:

Secondary education: Park Institution, Calcutta, India

Collegiate institutions attended	Dates	Degree	Date of Degree
Calcutta University, India	1970-73	B.S.	August, 1973
Calcutta University, India	1973-76	B.Tech	August, 1976
New Jersey Institute of Technology	1988-90	M.S.E.E.	January, 1991

Major: Electrical Engineering

Contents

Chapter 1

Chapter 2

List of Figures

Chapter 1

Introduction

1.1 Voice and Data communication

Thousands of Data Communication Networks are in services throughout the world. The systems range from small networks within a building to large ones— spanning city, state, country and world. No matter how the networks are operated, privately or publicly, the main objective of any data communication networks is to enhance the ability of users to communicate with one another. Till recent years generally two kinds of networks are in use for two different communication modes— **Voice** (real time) and **non-Voice** (usually called **Data**).

Some networks use **packet-switched** technology. In this case messages are split or segmented into multiple blocks of data of varying length, called **packets,** to improve the performance of a network. These packets are transmitted over a network from source to destination following some routing path prescribed as part of the network design. The source and destination could be user terminals, computers, printers or any other types of data communicating or data handling devices. Packets from multiple users share the same distribution and transmission facilities.

The other kind of networks use **circuit-switched** technology where, in general, voice or data are transmitted over a dedicated transmission path set up between any pair or group of users willing to communicate. The path is held up as long as the transmission is required.

Even today the real time voice transmission is still the main mode of communication and the bulk of voice type networks are still of analog signals. With the technological advancement people want to shrink distances by communicating data, file, facsimile, images, videotext, videophone, videomovie etc. All these services demand bit rate of transmission in magnitude of Mbps. However, most of the telephone networks are still designed for analog signals and by using a modem at most 9600 bps rate of transmission of data can be achieved.

1.2 ISDN

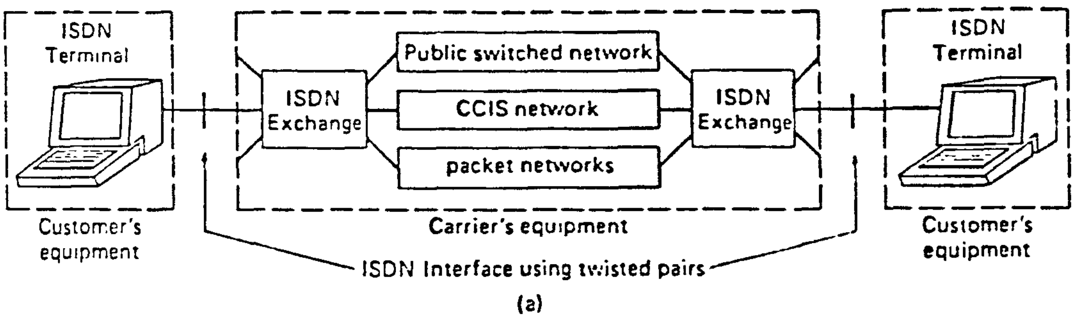
Though the bulk of the voice type (telephone) networks are still analog, digital transmission and switching are being introduced increasingly into these kind of networks (AT&T pioneered the first commercial introduction of digital carrier system). At present, telephone networks are partly analog and partly digital. The services that need higher bit rate can not be transmitted through this type of networks. Therefore, search for a new type of transmission system that can accommodate users with various type of traffic services comes up.

The most common type of data transfer is interactive, generally a few character to 1000 character (ie, 8 Kbits), between terminals and/ or computers. The other kind is file transfer between computers or between mass storage systems. Users' demand for these services have led to the continuing growth in digitization of the telecommunication network. Once telephone networks become **all digital**, any kind of data, whether of interactive type or inter-computer type, digital voice, or digital images, presumably can traverse the same network under different mode of interfaces. Each type of data might be handled differently or some could be combined.

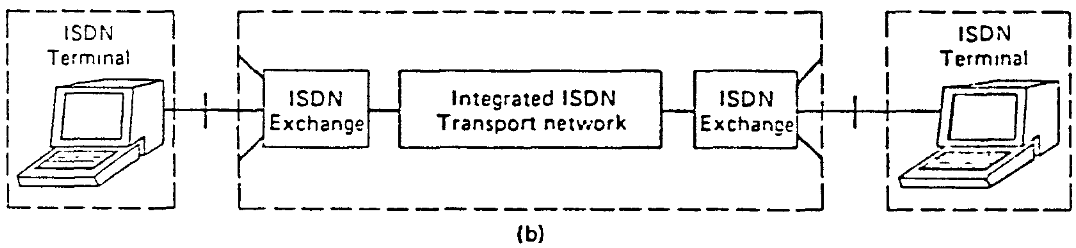
It is of great interest of today's technology to have a network that can implement aspects of both the circuit-switched and packet-switched networks. **ISDN** (*Integrated Services Digital Networks*) comes up to replace data networks and voice networks giving users an integrated service in the near future. The concept of integrated network started taking shape since 1970.

1.2.1 Evolution of ISDN

A variety of voice and non-voice services can be supported over a single homogeneous infrastructure. This is the primary aim of ISDN. In its early stage ISDN was designed considering the limitations of the existing public circuit switching networks, Common Channel Signaling networks (CCS, developed by AT&T, using Time Division Multiplexed frame) and packet switched networks. So the first step to implement ISDN is to define and standardize user to ISDN network interface. Next step is to replace existing network by an integrated network as shown in fig 1.1. The key ISDN recomendations were approved in 1984 and redefined in 1988.



(a) Initial stage of ISDN evolution.



(b) Later stage.

Figure 1.1: ISDN evolution

Till recent days, for different type of services, users are using different kinds of interfaces and sometime different mediums. ISDN supports single transport system. Users need not buy multiple services for multiple needs, users can get multiple services through a single access link (which is becoming possible as the fibre optic cables have transmission capacity in order of Gbits). As the price of fibre optic is becoming cheaper day by day, overall ISDN services will be very much cost effective.

Moreover, although we know that voice communication is universally of circuit switched type, there is great interest in making it possible to transmit voice in the form of packets in real time. Studies of this kind of voice, data communication are now being carried out namely in **B-ISDN** (ie, *Broadband ISDN*) and **ATM**(*Asynchronous Transfer Mode*).

1.3 B-ISDN

Broadband is no longer just a pipedream. Its introduction does not depend on implausible technological leaps, nor on enormous demand for hitherto unknown services. A clear market-led pathway could be foreseen.

Broadband switching and transmission technologies are maturing very rapidly. The key point is that an optical fibre infrastructure has already started taking shape. The fiber optic offers virtually almost unlimited bandwidth and the cost of single mode fibre is now merely a couple of cents. A 64 input–64 output digital video switch can now be placed on a single card measuring 4 sq. inches. Millions of packets can be switched with input-output delay of a few milliseconds. These

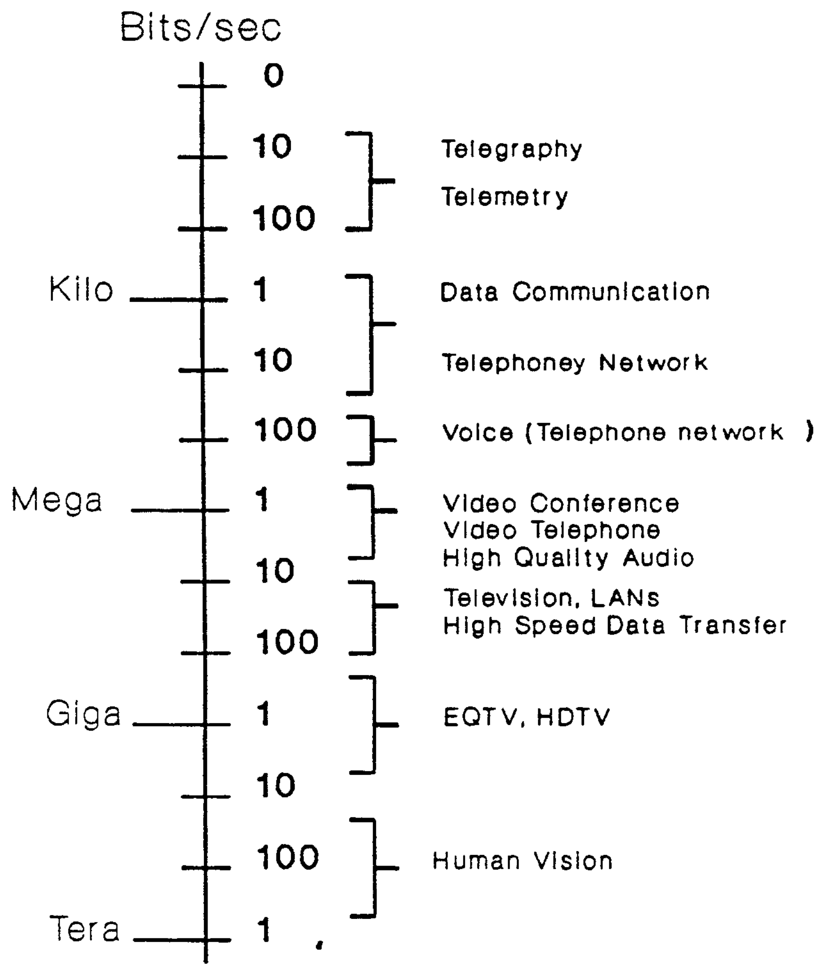


Figure 1.2: B-ISDN services and required bandwidth

and other advantages ofer the possibility of a practical and economical **Broadband ISDN (B-ISDN)** in the not-too-far future.

The diversity of possible B–ISDN services, shown in figure 1.2, provides much impetus for the implementation of B-ISDN, but complicates the task of specifying network switching requirements for different traffic characteristics(shown in fig.1.3).

Traffic Characteristics

	Voice	Data	Sub-Vedio	Vedio
General nature	on-off type 60-70 % idle time	Bulk or interactive	on-off type usually	Continuous
Bandwidth	64 Kbps	1-5 Mps	5-10 Mbps	Higher
Err.control	None	Essential	Medium	Medium
Pkt. rate	None during silence	Stochastic	No statistics	Continuous Vedio switching
Delay requirment	constant delay with upper bound	stochastic but low delay for interactive mode	a reasonable can be tolerated	Constant with upper bound fast call set up high calling volume
Blocking	1-2 % tolerated before setup	None	Some none some 2% tolerated	1-2 % tolerated before set up

Figure 1.3: Services and characteristics

Networks at present often focus on specific services, like speech communication, and needs of that services, such as bandwidth requirement, signal to noise ratio, call setup time and blocking probability. B-ISDN on the other hand, will be able to support many widely varying services each having different network requirements.

1.4 Broadband Services

1.4.1 Full motion video services

1. **Switched access television services:—** This means delivery of entertainment video to residential customers over switched services. It is a good representative of circuit-switched video. Using compression techniques like, Differential Pulse Code Modulation (DPCM), Diagonal filtering and decimation, variable length coding, a channel of 150 Mbps will be able to transmit NTSC or EQTV(Extended Quality video) signals. With advanced coding technology, in the future HDTV may potentially be transported on a single 150 Mbps channel. This service demands **Fast call setup** mode.
   Space division circuit-switched switch using CMOS technology is a promising switch with capability of 150 Mbps through a 64 x 64 switching subsystem. This can be fabricated by using 4x4 matrix of 16x16 CMOS cross-point chips.

2. **Video-on-demand and Videophone services:–** These two services require full motion video transmission channel, too. However videophone requires two-way channel. Both the services have less stringent setup response time than switched access television.

   The compressed video signal can be converted to 150 Mbps and circuit-switched. Alternatively, it will be more economical to packet-switch the compressed video signal at intermediate switch offices.

1.4.2 Subvideo- rate services.

The information transfer rate is less than 150 Mbps. All services except call setup services fall under this category.

1. **Audio Communication:—** Speech communications will be as important in B-ISDN as in the current telephone networks with existing encoding law (8000 samples/sec and 8 bit a sample) and pulse code modulation (PCM) transfer information at the rate of 64 Kbps.

   Different speech coding algorithms, those are now being used result in transfer rates of 9.6, 32, 64, 192 and 384 Kbps. So B-ISDN is to handle these different transmission rates for audio communication.

2. **Still picture communication:–** This has applications in advertising industry, library and conference. From these places videotext, pictures, drawings etc. can be shown. Depending on the resolution and color of the picture/ text demanded in the service, the range of the channel capacity could be from 50 Kbps to 48 Mbps.

3. **Remote disk services:–** The use of B-ISDN for remote disk access removes the distance limitation of local area networks, allowing centralized file servers to be accessed from home or some other location. Here B-ISDN needs fast call setup and high transfer rate capability and it should be comparable to latency time and transfer rates of disk drivers.

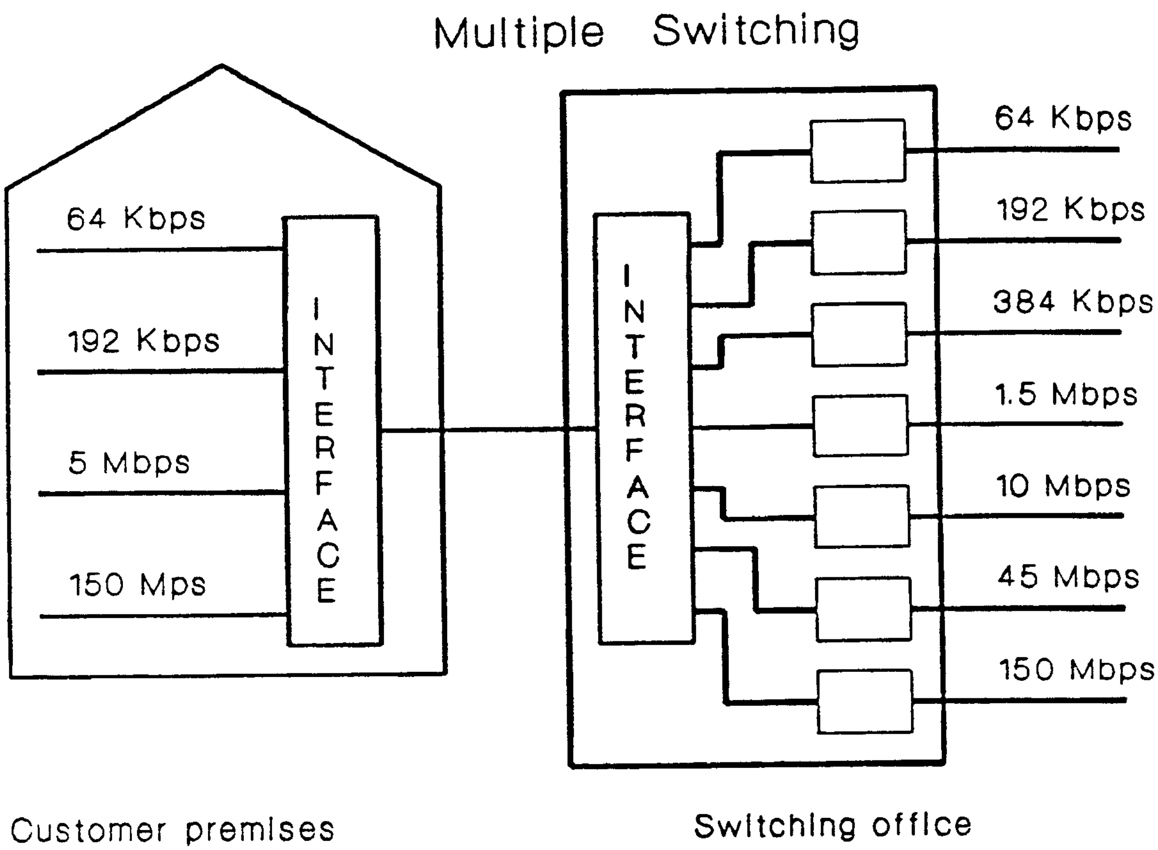


Figure 1.4: Multiple switching

1.4.3 Switching for sub–video services

High speed packet switching is a technique to handle sub-video rate services. CCITT Study Group XVIII is now studying a broadband packet switching technique called New Transfer Mode or Asynchronous Transfer Mode (NTM or ATM). This may offer a potential for effective resource utilization and more flexibility to accommodate a wide variety of service requirements.

Among several similar techniques for switching one technique is *Asynchronus time division.* This technique merges the simplicity of circuit switching with the flexibility of packet switching by using fixed length of packets with virtual circuits. Figure 1.4 shows the nature of the switch.

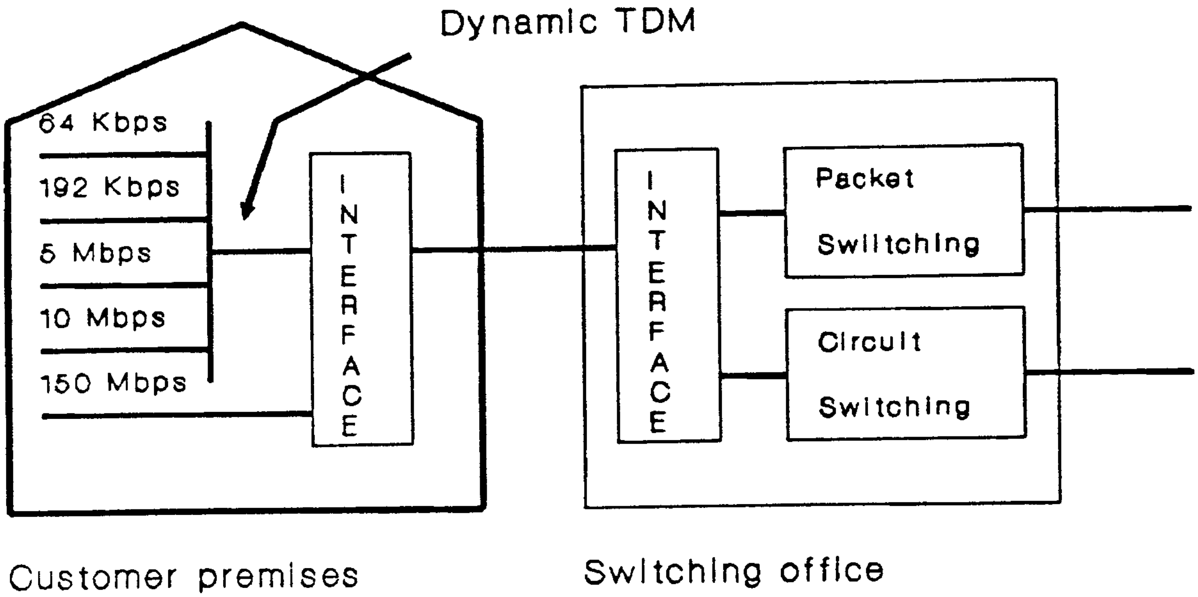


Figure 1.5: Dynamic Time Division Multiplexing

Another technique is dynamic time division multiplexing **DTDM** . This subdivides a single high speed physical access channel into multiple virtual channels. The multiple channels are dynamically allocated and sized as needed on per call basis. Both techniques use a single physical access channel (as shown in fig.1.4) connected to a *fast packet switching fabric* that routes packets to the proper destination.

Chapter 2

Model Design of the switch

2.1 High performance switches

This chapter introduces a model for **fast packet switching** technique that can be implemented in **B-ISDN** services where we need fast data (we mean voice or video with upper bound of delay and data) transfer access but no call setup procedure. This model does not consider the case of continuous flow of data through a dedicated channel. In our model we are going to study a non-blocking network switching that is highly applicable to *fast packet switching.* As we have mentioned in the earlier chapter that in sub-video transmission range where we do not need call setup, circuit-switched packets can be treated like packet-switched packets. Here all types of packets have a fixed length. This allows the use of a very simple framing mechanism which is more suitable for high transmission rate. In Broadband transmission we consider a wide bandwidth channel where data will be transmited at high rate and in a non-blocking way. The crossbar switch is in the category of a non-blocking switch.

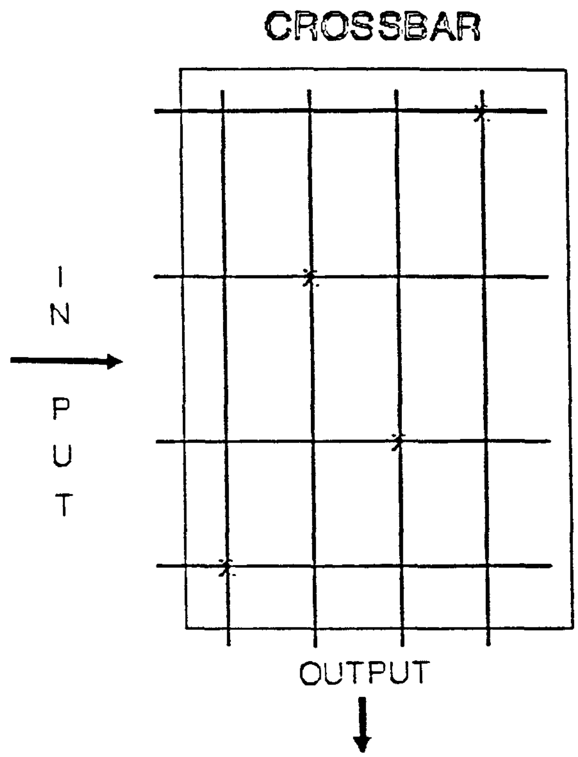


Figure 2.1: Crossbar Switch

2.1.1 Crosbar Switch

From the start, it was recognized that a **crossbar** switch with **n** terminals and n^2 crosspoints, shown in figure 2.1, could achieve the non-blocking characteristics, only at prohibited cost in a large system. By non-blocking we mean that any pair of idle terminals can communicate under arbitrary traffic conditions.

Here we model a high performance switching fabric that can accommodate circuit-switched and packet-switched traffic in a unified manner. The switch is self-routing and uses fixed length **minipackets** within the switching fabric. The motivation for this model arises out of need for future telecommunication that presumably will support a variety of interfaces such as ISDN, high speed LAN, high performance integrated work stations. This can very well be described as pre-**B-ISDN** architecture.

2.1.2 Switching Architecture

There are basically two distinct switching architectures, namely– single path networks such as bus, ring and common memory system; and the other one is multipath networks such as crossbar and multistage interconnection networks. The multipath architectures become the key alternative to meet the higher capacity requirements of the future networks of higher transmission rate of various traffic. The recent approaches in designing high performance telecommunication switch are buffered Banyan and Batcher–Banyan based switch. Buffering at each node, in case of buffered switch, introduces delay through the switch. This is undesirable for time constrained services. But for packet–switched services it gives high throughput. This switch does not provide simple modular growth from the basic building block. The Batcher-Baynan switch, though internally non-blocking, has a problem of congestion and blocking at the output port. Another kind of switch is of knock-out type, but this is purely for packet switching circuits.

What we are going to model here can switch both types of packets and has the concept of modularity growth. This can be built with VLSI technology and aims for a single chip basic building block such that later, if necessary, a large switching range can be configured. If there are 16 to 2000 input-output ports, then the throughput capability of the switch will be 500 Mbps to 60 Gbps when each of the ports is assumed to have the speed of 32 Mbps.

As circuit-switched transmission is time constrained the call packets are given priority over data-type packets such that if the desired output is not busy the packet-switched packet can be switched. If any output desired by the data packet is found to be requested by any call packets then data packets will have to wait in their respective queues.

2.2 Basic Switch Structure

The switch system consists of two elements :

- **Switch Fabric Adapter** (SFA)
- **Switch Fabric Element** (SFE).

2.2.1 SFA

At the SFA, the incoming packets are converted into uniform fixed length **mini-packets** with a header containing the routing information, the input port number and a priority bit (1 for call, 0 for data) of the minipackets. Therefore, within the switch fabric all information traverse from one SFA at the input side to the other at the output side in the form of minipackets. The size of the minipackets are assumed to be 32 bytes including the header. The link speed is assumed to be 32 Mbps per port. This range of speed can be handled by today's lowcost CMOS technology.

2.2.2 SFE

SFE is a crossbar switch of **n** inputs and **n** outputs, shown in figure 2.1, having n^2 cross elements. In our simulation model we assumed **n** =4. This is totally a non-blocking type of switch. The row represents the output port and the column represents the input port. When one input port wants to communicate to a non-busy output port and the switch connection is allowed, the corresponding crosspoint closes and the first minipacket at the outgoing side of the corresponding input queue is dispatched to the output.

To avoid collision and loss of minipackets, we consider that at each input there are two queues–one for circuit-switched (we will name this **call** type minipackets) and the other for **data** type minipackets. The packets arriving in some input line will pass through SFA, get minipacketized and then be put in the corresponding call or data queue of that input.

2.3 Queue Model

Voice packets, for a long time, have been considered as a source of continuous or stream traffic. Hence, only circuit–switched method was considered for switching. But Brady's experiment indicated that the voice is actually made of alternate talk spurts and silence periods. It was also shown that both of these are exponentially distributed. So in our model we have also assumed Poisson arrival rate and an exponential service rate for circuit-switched packets.

In our model, we generate both types of packets with average packet length of 16, 32, 64 and 128 Kbits and the inter-arrival time of those packets are in Poisson distribution. While generating packets we assign input port number, output port number and priority bit to those packets. Then the packets are minipacketized and put in the queues of the corresponding input ports.

Now the question comes from which input port and what type of packets are to be despatched if there are demands for the same output port by more than one input ports or/and by both types of packets? To resolve this competition we considered two criteria :

1. **Higher priority** of call packet over data packet.

2. **Largest queue size** will win the competition for the same output port.

First, for each output port, the call packet queues at all the input ports are checked whether any packet is waiting there for the output port. If there is only one input port for an output port, it will be ready for switching. If more than one input ports are there for the same output port only one input port will be selected after competition.

Since the call packets have higher priority and only one queue can be selected for each output port, the queues of data packets are selected for the output ports those are not demanded by the queues of call packets.

After each switching switched queues are updated. This means the mini-packets are pushed towards the outgoing-end of their respective queues. Checking and selecting of the queues are repeated again. The switch fabric is shown in figure 2.2.

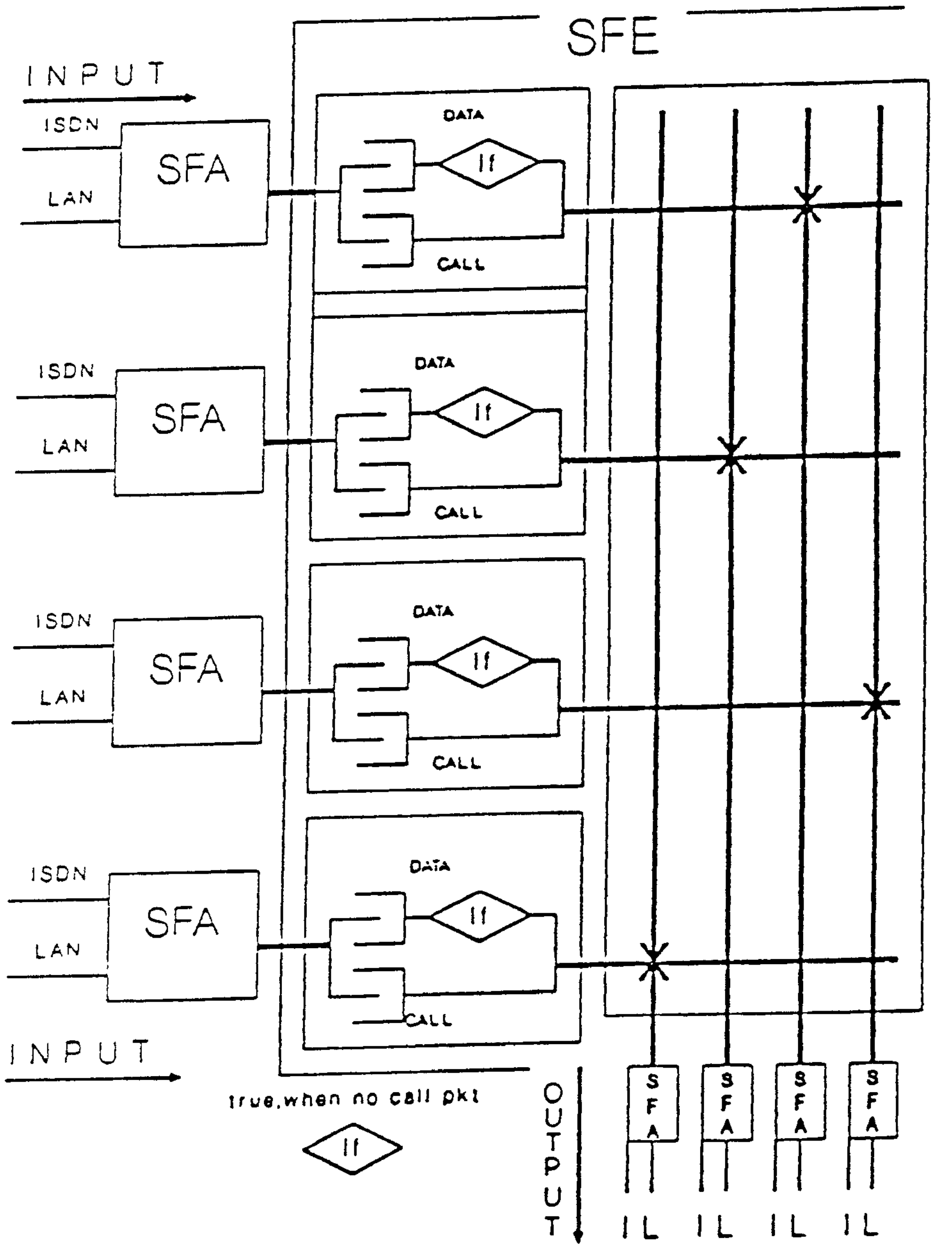


Figure 2.2: Switch Fabric

2.4 Neural Networks

It was mentioned before that to resolve the competition ie, to find the winner input port for an output port we want to use a **Neural Network** model which works on "**winner-take-all**" algorithm.

The Neural Nets attempt to achieve good performance via dense connections of simple computation elements–called **neurons** .

Instead of performing a program of instructions sequentially as in Von Neumann computer, Neural Net model explores many competing hypotheses simultaneously using massively parallel nets composed of many tiny computational elements connected via links with variable weights. In a simplest model, neural nodes sum up N weighted inputs and passes the result through some non-linear function namely hard limiter, threshold logic and sigmoid. A simplest model (shown in figure 2.3) along with a block diagram of Neural Net classifier. The classifier determines which of the M classes is the most representative of an unknown static input pattern containing N input elements. If the character class is provided, then the classifier's output can be fedback to the first stage of the classifier to adapt weights using a learning algorithm.

2.4.1 Hopfield Net

There are a few types of classifier nets. The Hopfield net normally is used with binary inputs. This net is mostly appropriate when exact binary representation is possible.

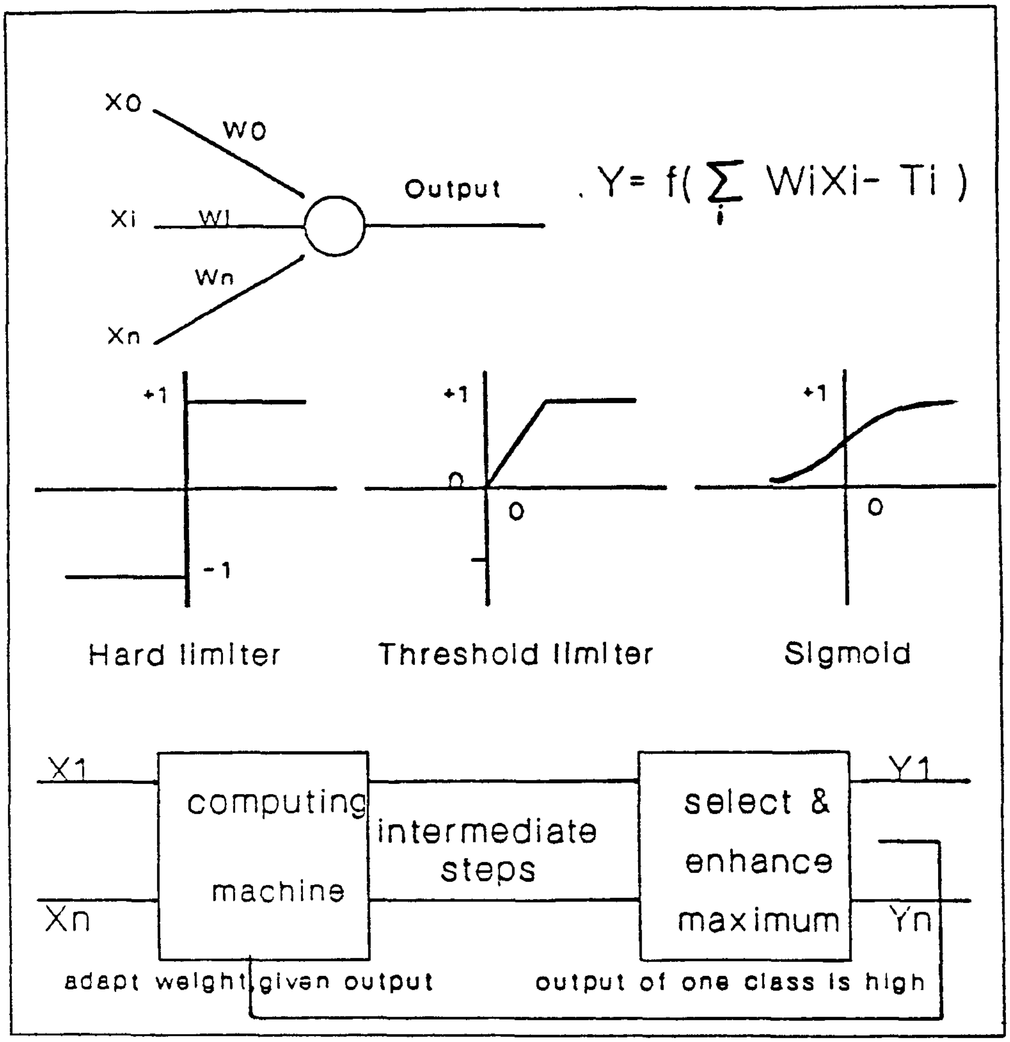


Figure 2.3: Neural Net Principle

This net can be used as an associative memory or to solve optimization problem. This net has nodes containing hard limiting non-linearities and binary inputs. The outputs take values +1 and -1 (sometime +1 and 0). The output of each node is fedback to all other nodes via weights denoted by t_{ij}. First, weights are set using the given recipe from input pattern. After this, the net iterates in time steps using given formula. The net ultimately converges when the outputs no longer changes. The desired output is obtained.

2.4.2 Basic Analog Neural Network.

Figure 2.4 shows a basic Neural Net model in terms of electrical circuit. The connection from neuron j to neuron i is through register R_{ij}. The weight of the connection is denoted by mod $|T_{ij}|$. The strength of the connection is inversely proportional to the register R_{ij}. The sign of the connection is determined by whether it comes from the positive or negative output of neuron j. If T_{ij} is greater than 0, then the connection is said to be excitatory, otherwise it is inhibitory.

One important aspect we assume is that the output of neuron j does not immediately affect the output of neuron i. The neuron i has an internal capacitance C_i that causes a time delay. No direct feedback in the circuit ie, T_{ij}=0 for all i. Each neuron can have an external input denoted by I_i. A threshold t_i is subtracted from the sum of all the inputs and the result is the net input to the neuron. Mathematically, in a system of N neurons the state variable is governed by

$$C_i \frac{du_i}{dt} = -\lambda_i u_i + \sum_{j=1}^{N} T_{ij} . g(u_j) + I_i - t_i$$

where

$$\lambda_i = \sum_{j=1}^{N} |T_{ij}|$$

$$g(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

(-1 could also be set at value 0)

S is the sum of the last three elements. We can see that u_i evolves as a negative exponential with time constant $\frac{C_i}{\lambda_i}$ and decays towards the value $\frac{S}{\lambda_i}$. Since

λ_i is positive and function g is a hard limiter, output of neuron i evolves towards +1 and -1 (or 0) depending on the total input minus the threshold.

2.5 Implemented Model

A Hopfield Neural Net architecture is proposed to control a **crossbar switch** in a real-time in order to effect the switching of packets at very high rate. General learning algorithms have great difficulty in finding a solution for this. Instead, for each problem in the class there is a corresponding network that solves the problem in a more straight forward way.

Our problem of dispatching minipackets through a crossbar switch is explained by referring to figure 2.5. The figure shows how a request for packet transmission through an n x n (we have n=4) crossbar can be mapped onto an n by n matrix **r**. Let us name it request matrix. The rows and the columns of **r** are associated with the output and the input lines of the crossbar switch respectively. The elements of the matrix are determined by the queue managers.

We mentioned it before that, if for any output there is a call packet request, data queues are prevented from switching for that output at that cycle of switching because of higher priority of the call packets.

Each queue manager reads header containing priority bit(1 for call, 0 for data traffic) and address bit (output port number) from the minipacket at outgoing end of its queue. It always keeps track of the size (r_c for call and r_d for data) of its both queues. If a queue manager has $r_c > 0$ it always requests for call packet switching to control manager, else request for data packet switching if $r_d > 0$.

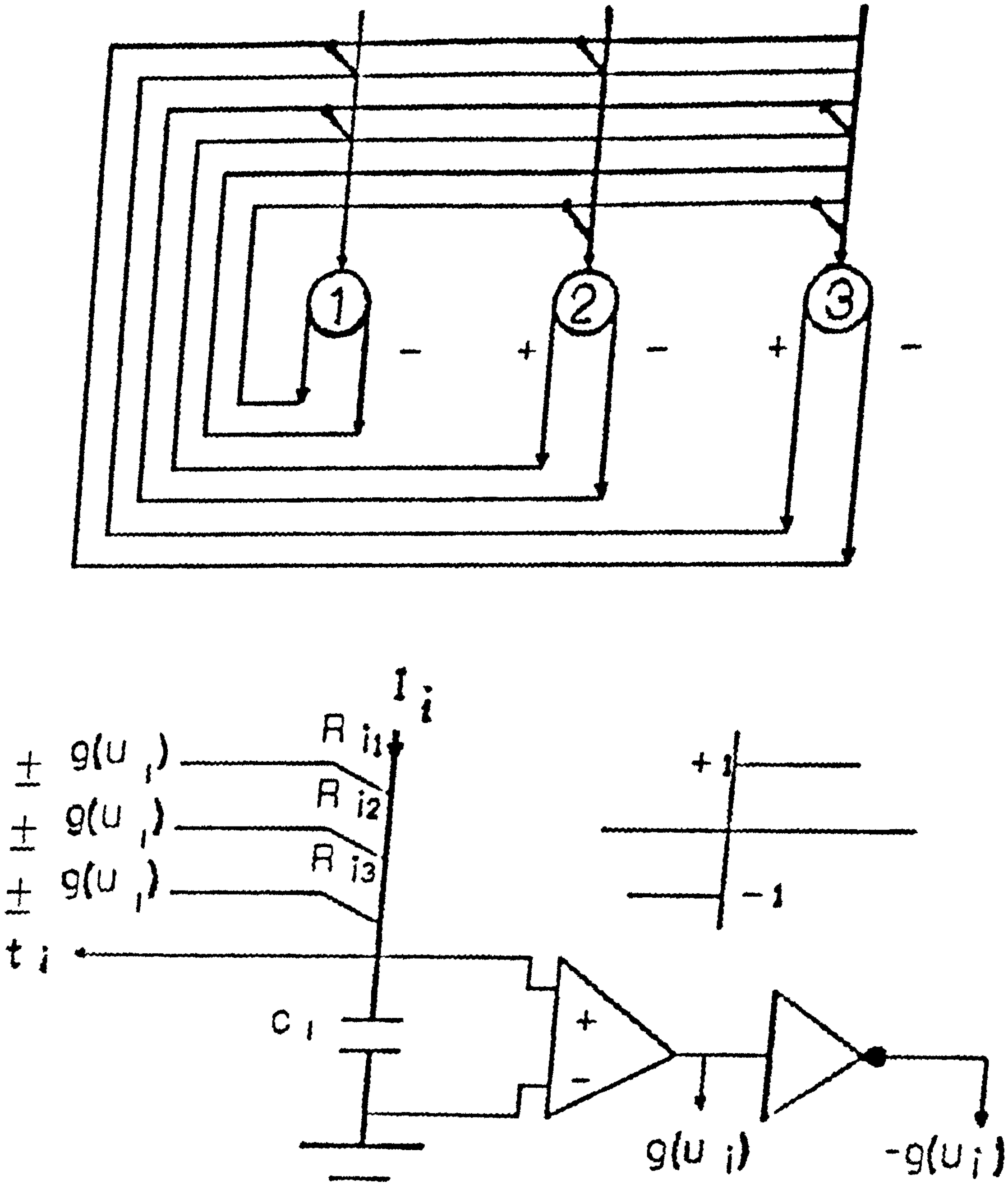


Figure 2.4: Neural Net model

The control manager all the time keeps track of all the queue managers and their requests. The queue size with its input port and destination port is reffered to as request vectorr_{ij}. The control manager always takes the call request vectors, if available, from all queue managers. If control manager finds any of the output port is not booked by any call request vector, it picks up data request vector from the input lines. Finally request matrix is formed and sent to Neural Network.

Request for call packet switching is immediately supplied to **Neural Net** having one neuron for each crosspoint. The Neural Net, using **One-Winner-Take-All** algorithm for each output line, determines which input line and what type of queue will be switched at that time. If minipacket on the input port j is to be transmitted to the output port i of the crossbar then in the **C** matrix $C_{ij} = 1$ otherwise $C_{ij} = 0$. The resulting configuration vector is returned to the queue managers and all the crosspoints, chosen by the configuration matrix, are closed onto the crossbar switch. Each queue manager, if selected for transmission, transmits a single minipacket from their outgoing-end to the appropriate output port. For example we put the following matrices with an arbitrary value :

$$Request matrix \mathbf{r} = \begin{vmatrix} c_{00} & 0 & c_{02} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & d_{21} & 0 & d_{23} \\ d_{03} & 0 & 0 & 0 \end{vmatrix}$$

$$Configuration Matrix \mathbf{C} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$$

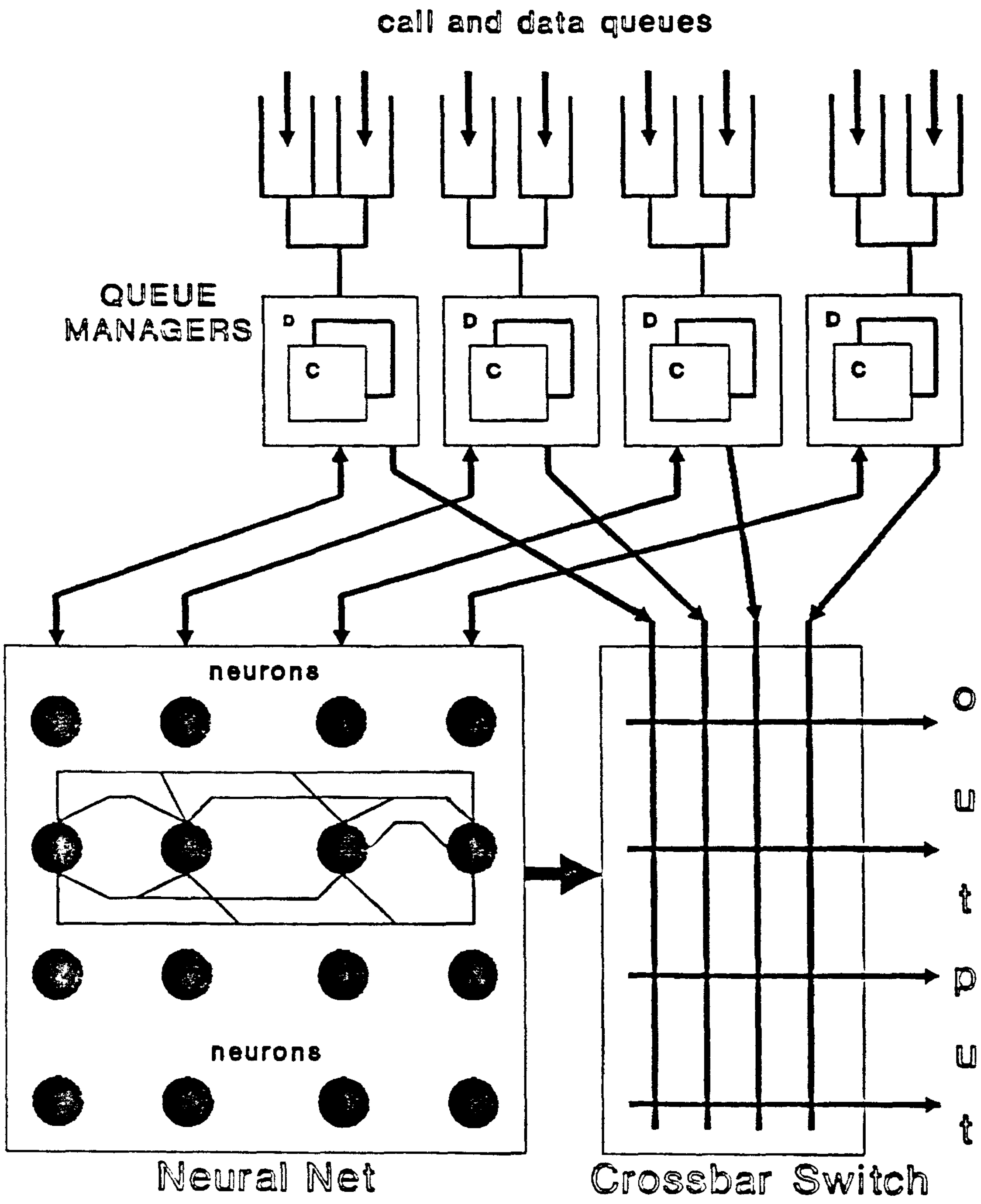


Figure 2.5: Suggested Model

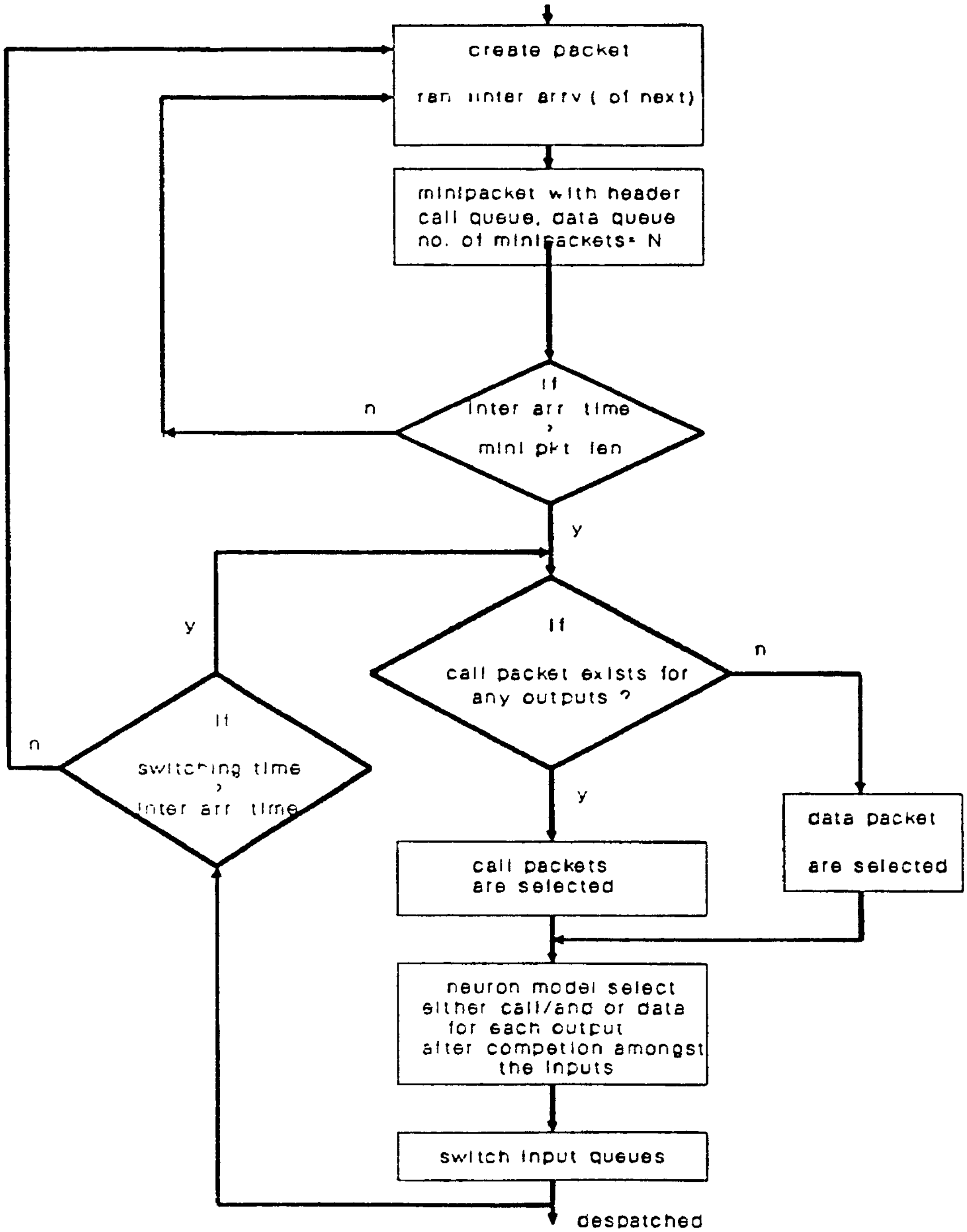


Figure 2.6: Flow Diagram

After the switching, row request vectors are updated by the queue managers again. Before sending request, evreytime queue managers check whether they have received any packets. This process continues. The flow diagram of our simulation model is shown in figure 2.6.

2.6 Winner-Take-All-Circuit

This is a kind of network where, given **n** neurons with the same internal state u_0 and all initially off, only one which has highest input value will turn on. The process of choosing a neuron with largest value is known as **competition** . By using external inputs (in our model we used queue size as input value) the circuit provides a method of selecting a neuron to turn on. It is a simple circuit where each neuron has an inhibitory connection to other neurons and initial threshold value. Analysis by E.Majani 'On K-winner-take-all network' shows that with properly chosen threshold and connection strength this circuit has the desired properties.

2.6.1 Improvement

In Neural Net all neurons should be densely interconnected. In large switch the interconnection is a big problem. In the above mentioned network the circuit can be made of fewer number of connections by making a weighted sum of all inhibitory strength of all neurons only once for the whole circuit and providing a self excitatory connection to negate the resulting feedback. In this method, for n x n crossbar, instead of n^2 connection we can do the same result with 3n connections. This modified circuit is shown in figure 2.7

Winner take all circuit

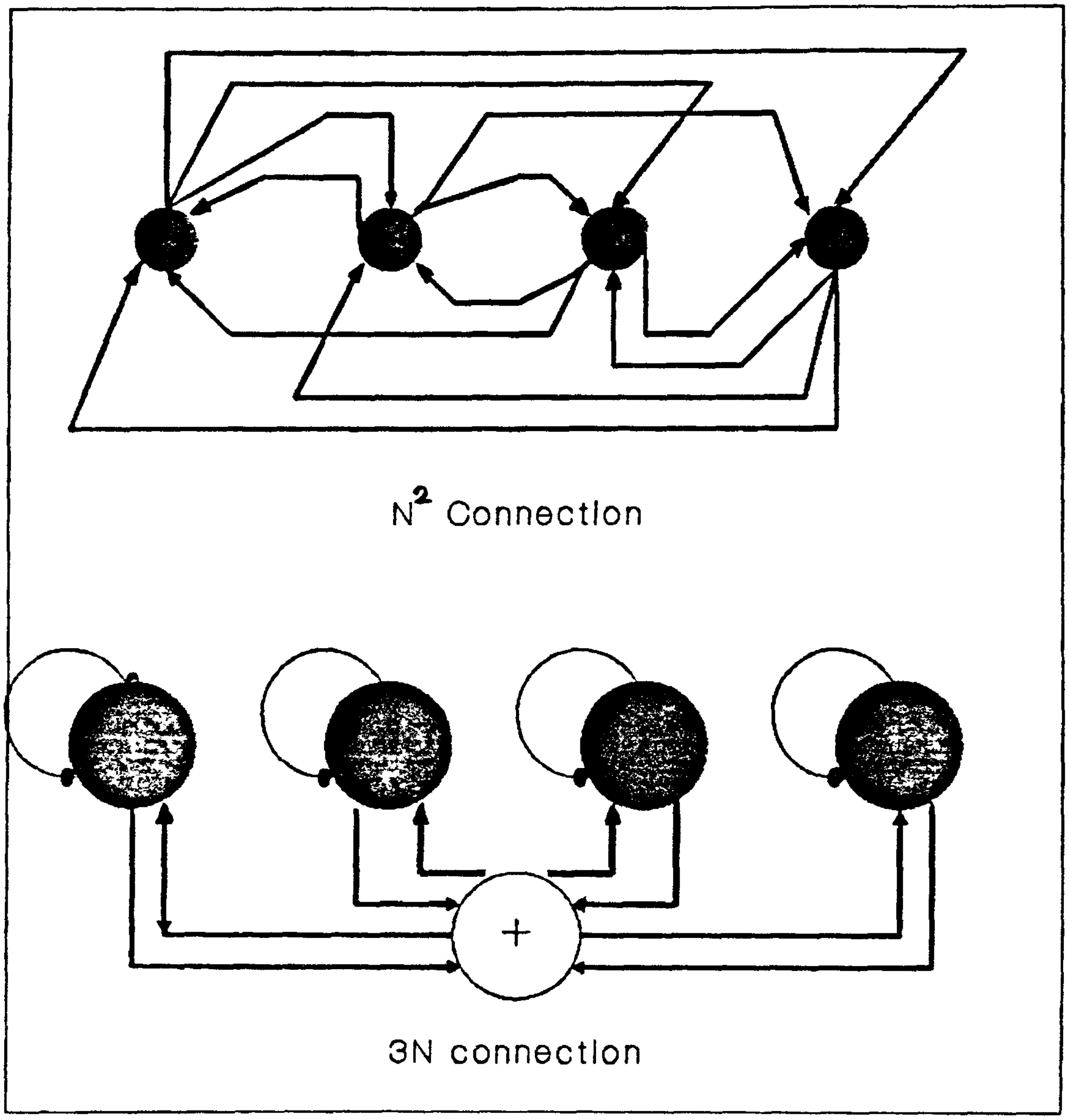


Figure 2.7: Simple model for winner-take-all circuit

Chapter 3

Result

Problem Statement

In our simulation model we wanted to analyze the performance of **Fast Packet Switching** in a **Non-Blocking Crossbar Switch**.

Experiment

- Fixed length of **minipacket** was taken to be 0.000008 sec. ie, 32 bytes long if the capacity of each port is assumed to be 32 Mbps. Four bytes of header length was assumed. The header contains the input-output port address, priority bit, arrival time and minipacket numbers for each parent packet.

- Simulation was done to measure two parameters, Delay and Throughput, to assess network performance with different packet lengths and different voice/data packet ratios at different input loads.

- The same was done when any one of the output port is highly demanded by more than one input port (ie, non-uniform loading condition of the switch).

- To measure performance of our model, simulation was also done with algorithm other than neural net to select input port for switching.

Packet length: Three different average packet lengths- 16 Kbits, 32 Kbits and 64 Kbits were chosen and for each the normalized throughput and delay were measured at different input loads. The results are shown in figures (1, 2, 3, 5&6, 7&8, 9&10).

It was also observed that voice packets are usually longer than the data packets. Therefore, we studied our model with data packet of length 32 kb and voice packet of length 128 kb. The results are shown in figures (13 and 14).

Voice/Data ratio: Statistical studies show that in communication networks usually 30-35 % packets are voice packets. So we did the simulation at 30 % of voice packet. Only at 32 kbits of packet length simulation was done at 40 % of voice packet, too. The results are shown in figures (4,11&12)

Loading : In the original model we selected input and output ports from uniform distribution. But in the real world one output could be demanded by more than one input port at the same time. So we studied our model with non-uniform demand ie, 50 % voice packets are directed to one output port and the rest of the traffic is uniformly distributed among the other three output ports. The results are shown in figures (15 and 16)

Model variation: Instead of enabling the input ports to switch by Neural Net algorithm we also studied a simulation where it polls the input ports in an asceding order of numbers. Throughput and delay were measured at different loads. The results are shown in figures (23 and 24)

To assess the reliability w.r.t delay, we measured the probability that a packet will be delayed beyond a cut-off delay time. Figures (17 - 22) show how the probability varies w.r.t load and cut-off delay time.

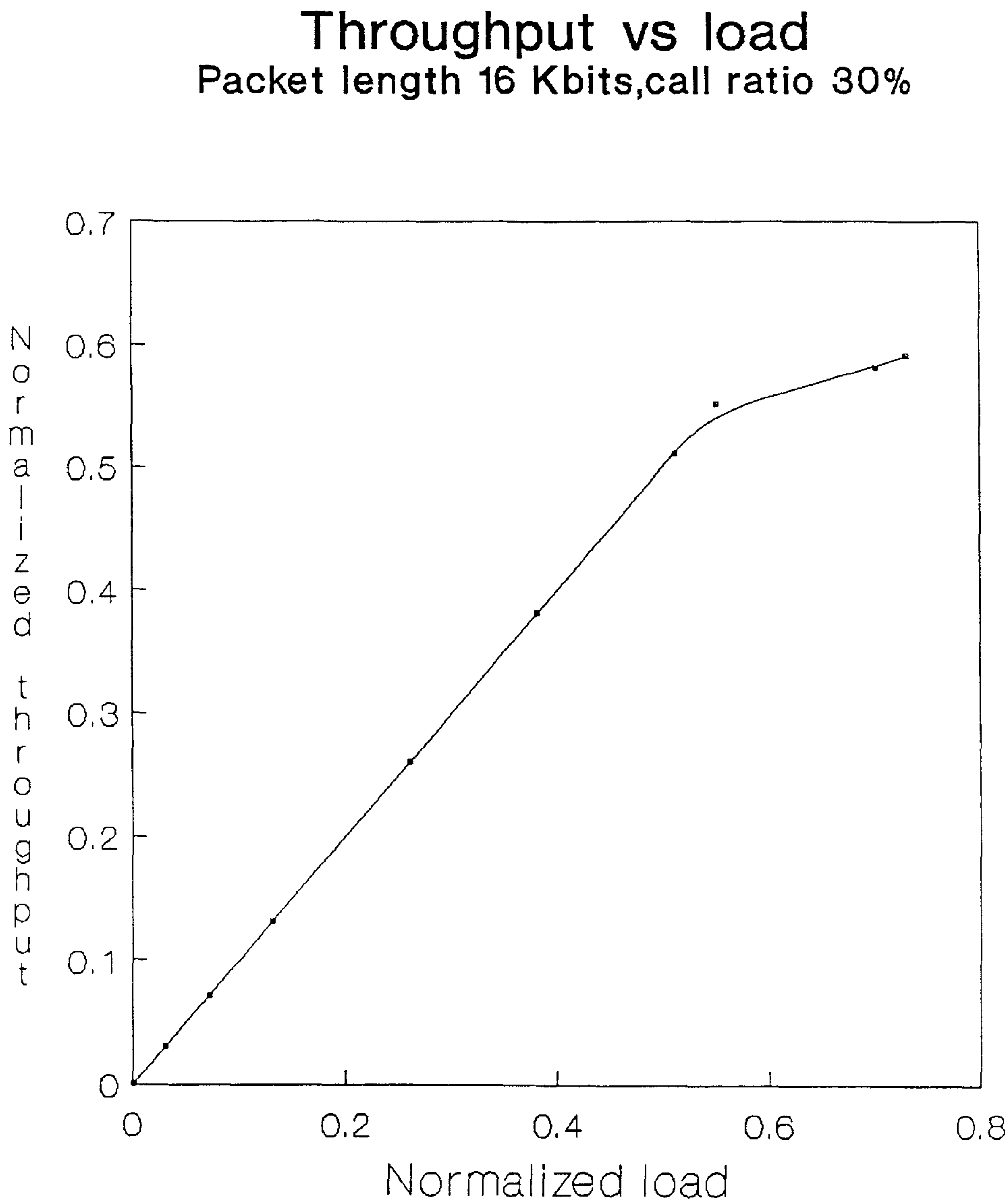
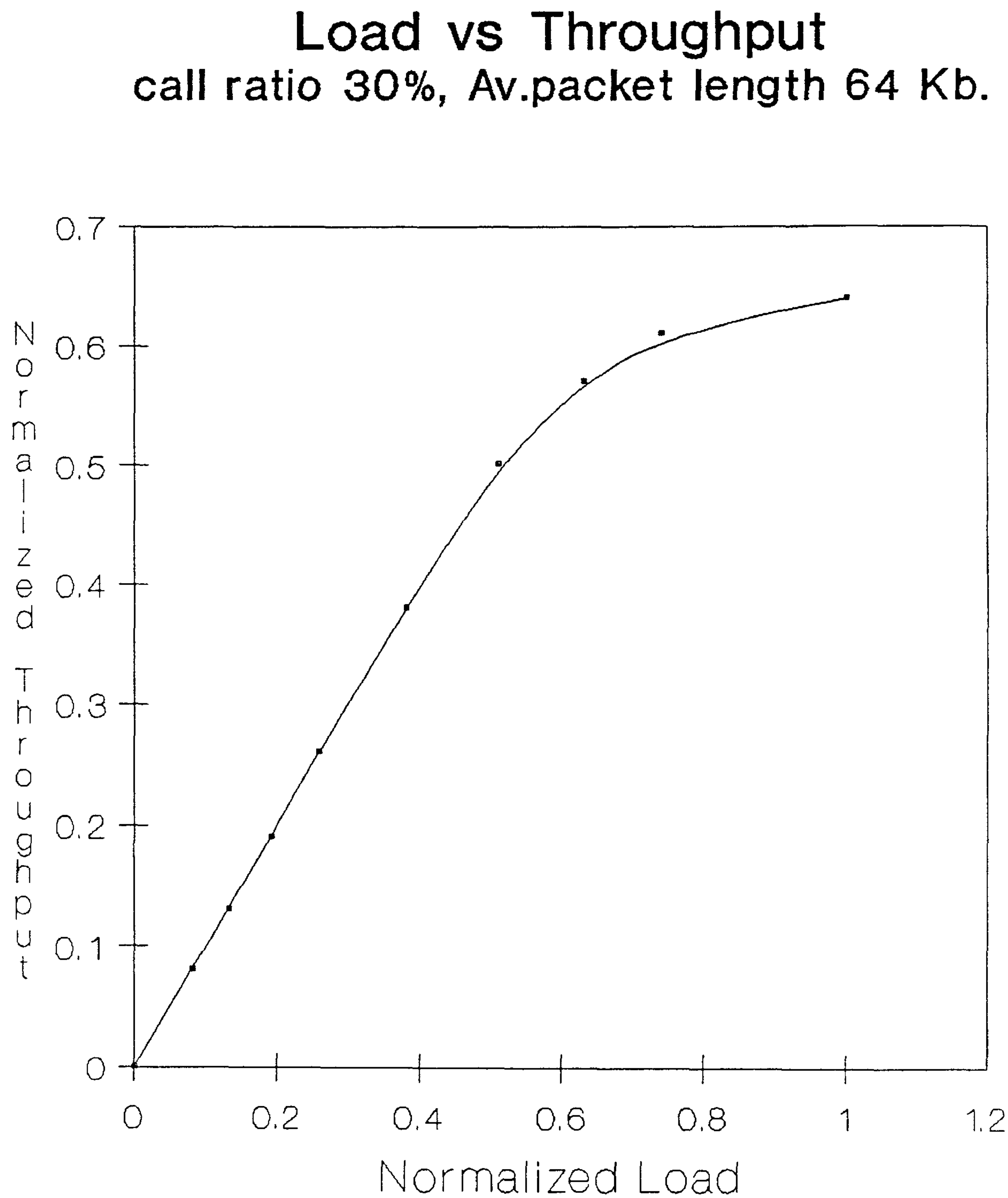

Throughput vs load
Packet length 16 Kbits,call ratio 30%
Normalized throughput
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
Normalized load

Figure 3.1:



Load vs Throughput
call ratio 30%, Av.packet length 64 Kb.
Normalized Throughput
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
Normalized Load

Figure 3.2:

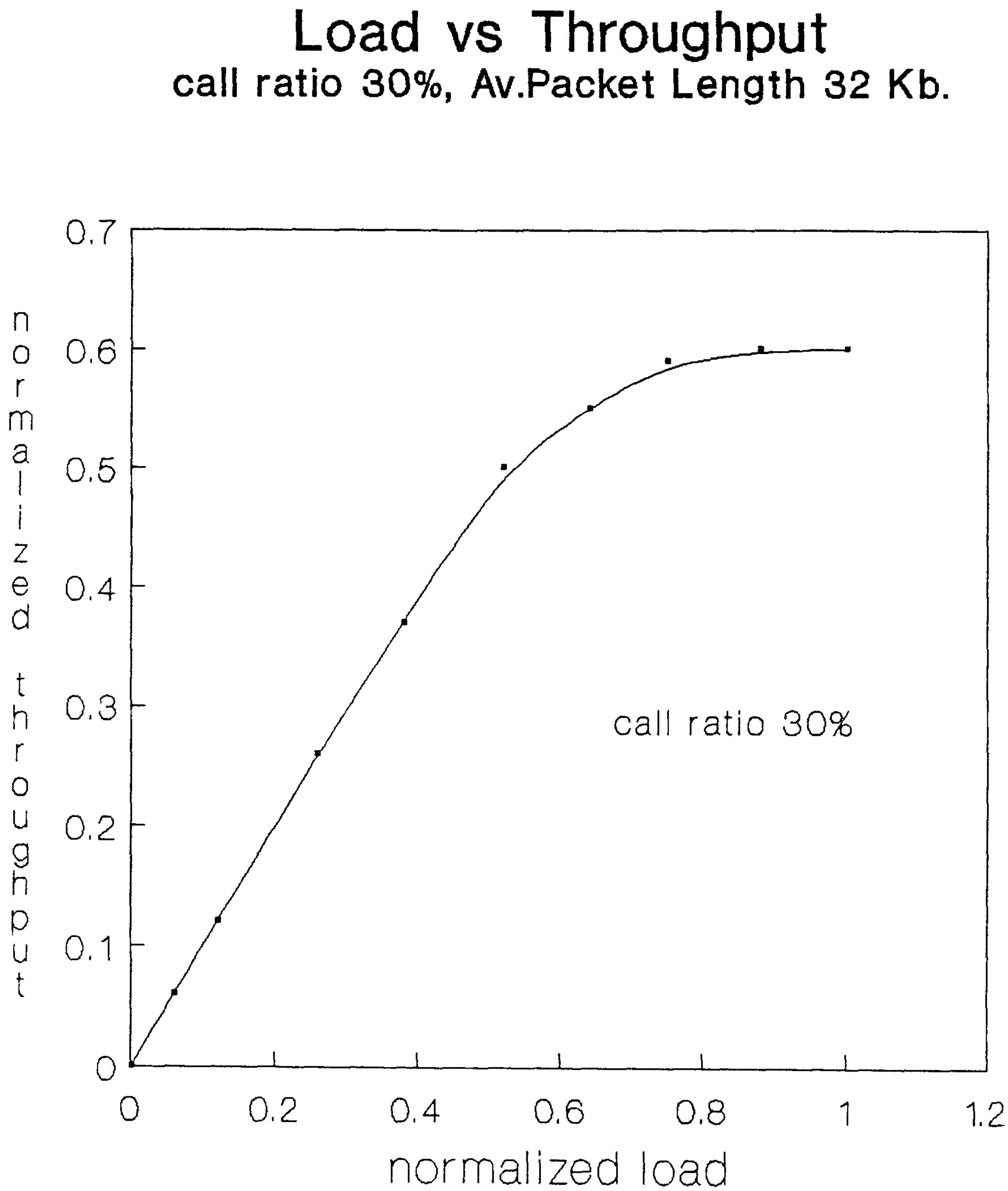

Load vs Throughput
call ratio 30%, Av.Packet Length 32 Kb.
normalized throughput
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
call ratio 30%
0
0.2
0.4
0.6
0.8
1
1.2
normalized load

Figure 3.3:

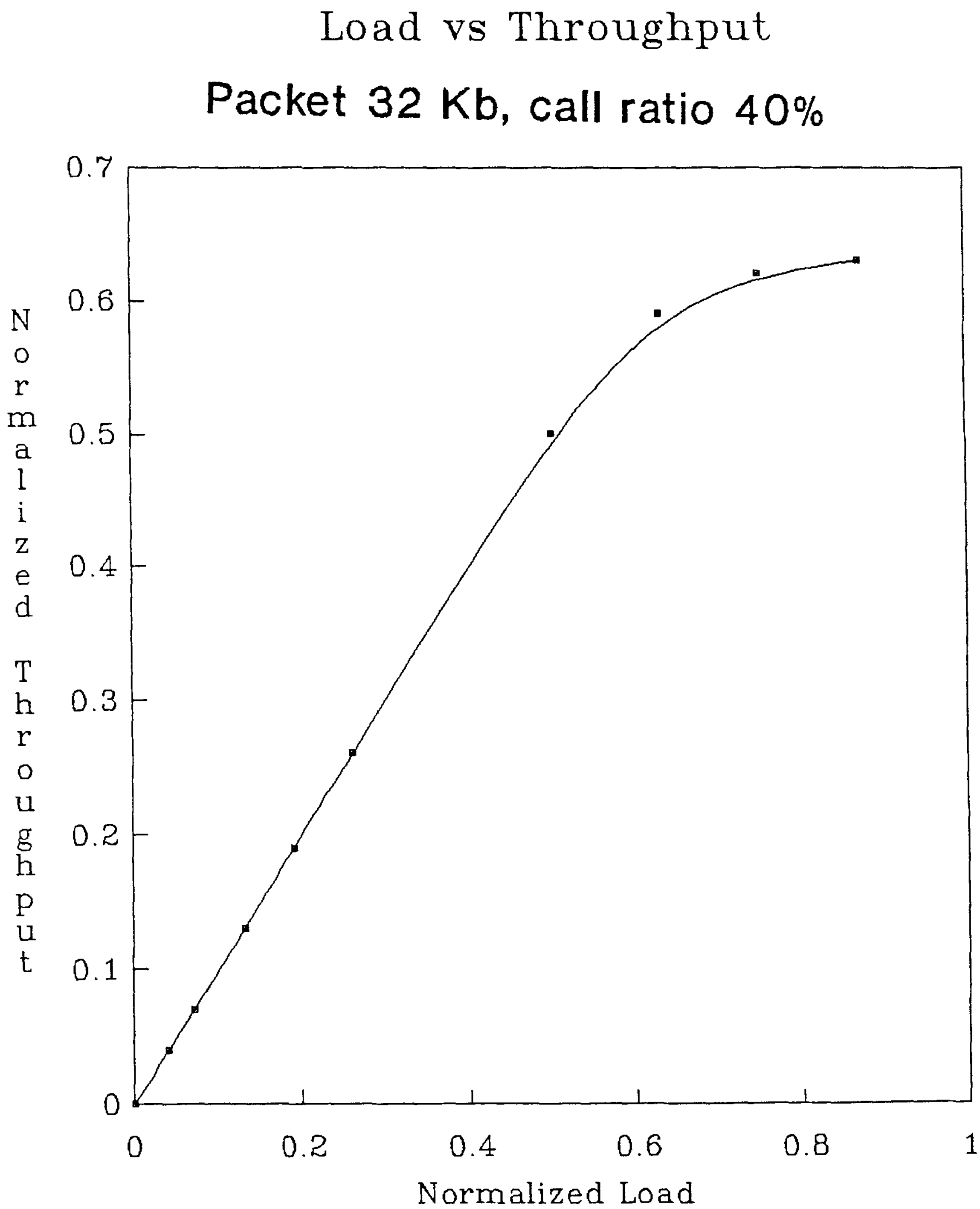

Load vs Throughput
Packet 32 Kb, call ratio 40%
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Normalized Throughput
0
0.2
0.4
0.6
0.8
1
Normalized Load

Figure 3.4:

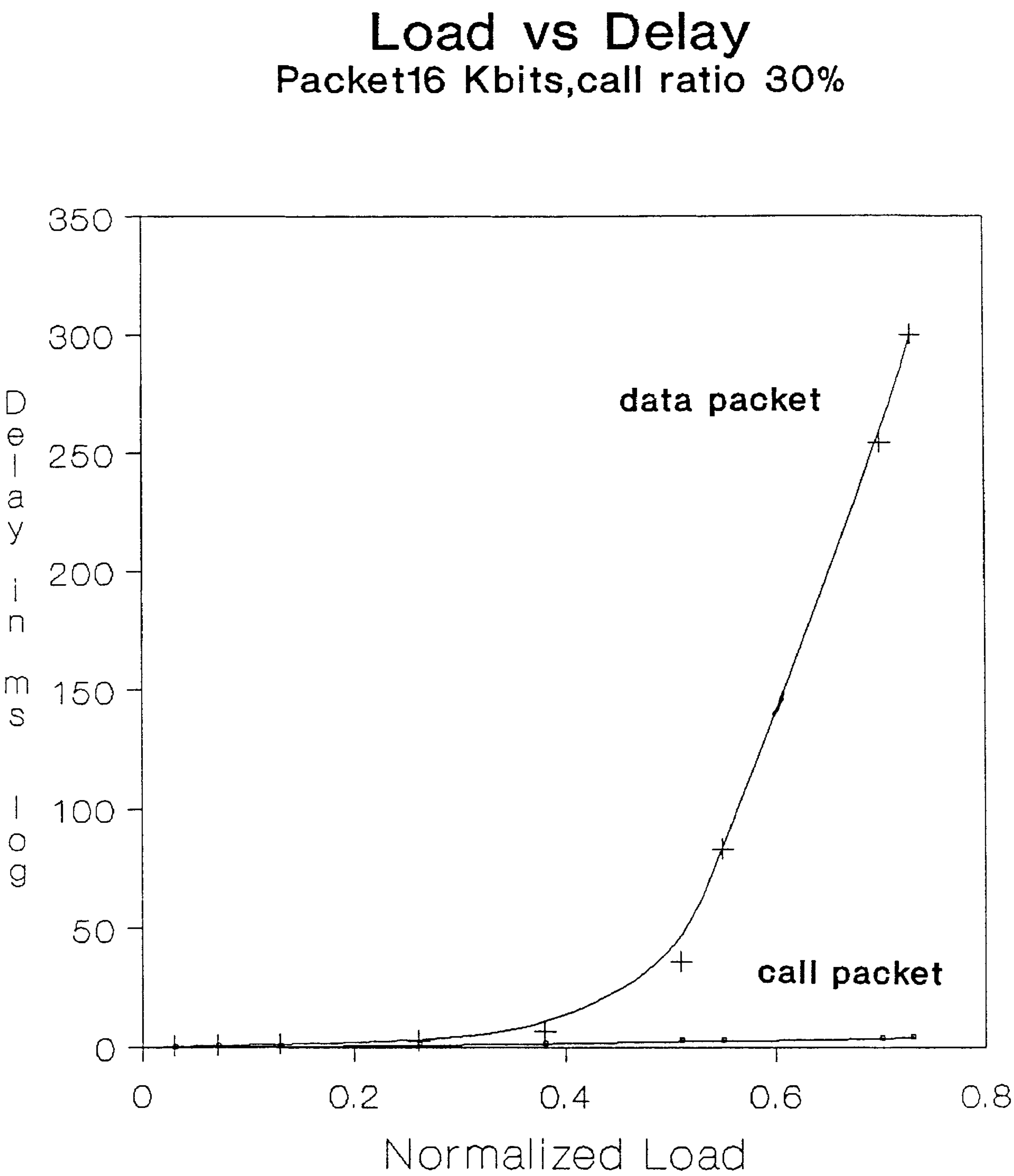
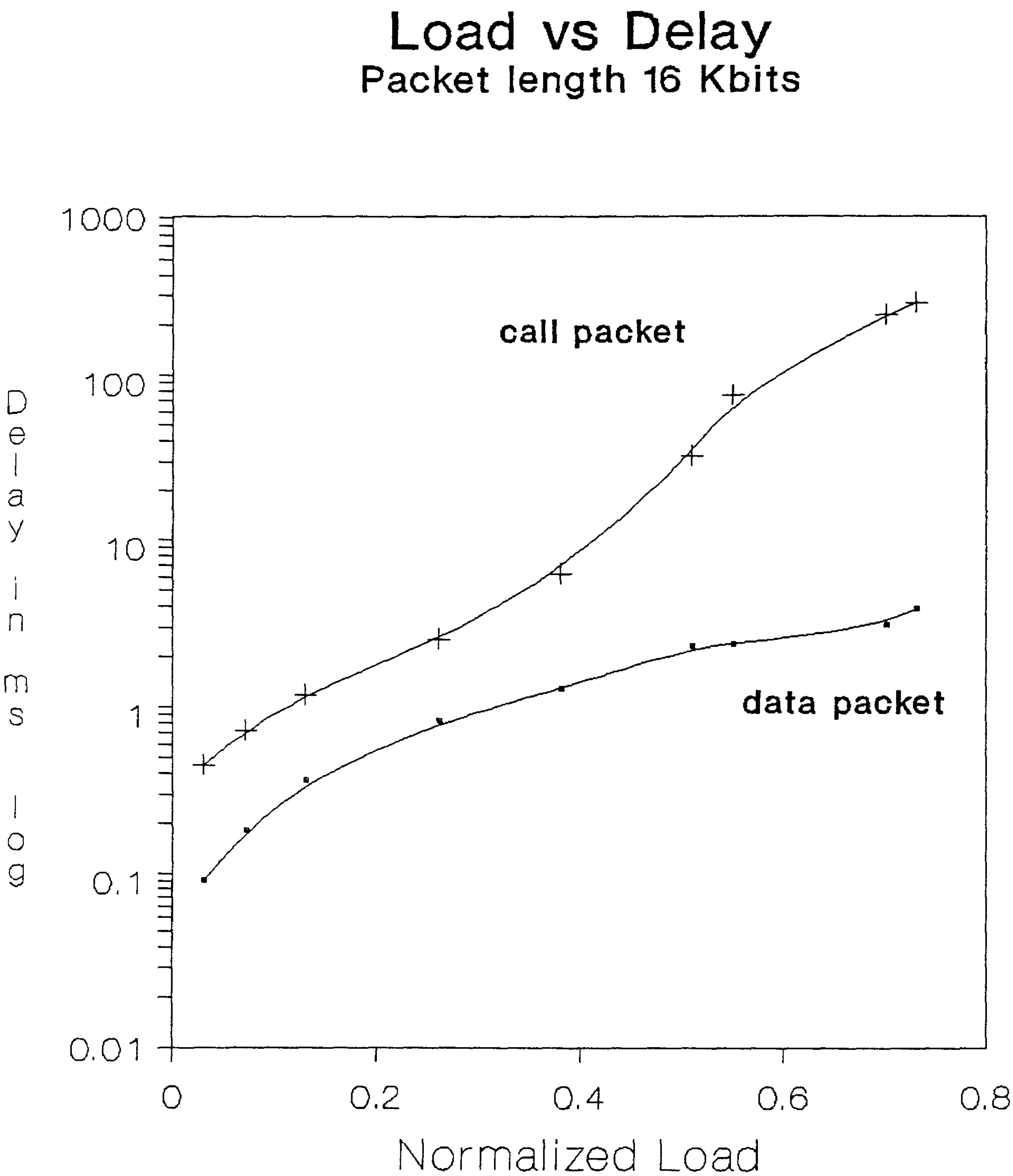

Load vs Delay
Packet16 Kbits,call ratio 30%
350
300
250
200
150
100
50
0
Delay in ms log
data packet
call packet
0
0.2
0.4
0.6
0.8
Normalized Load

Figure 3.5:



Load vs Delay
Packet length 16 Kbits
1000
100
10
1
0.1
0.01
D e l a y i n m s l o g
call packet
data packet
0
0.2
0.4
0.6
0.8
Normalized Load

Figure 3.6:

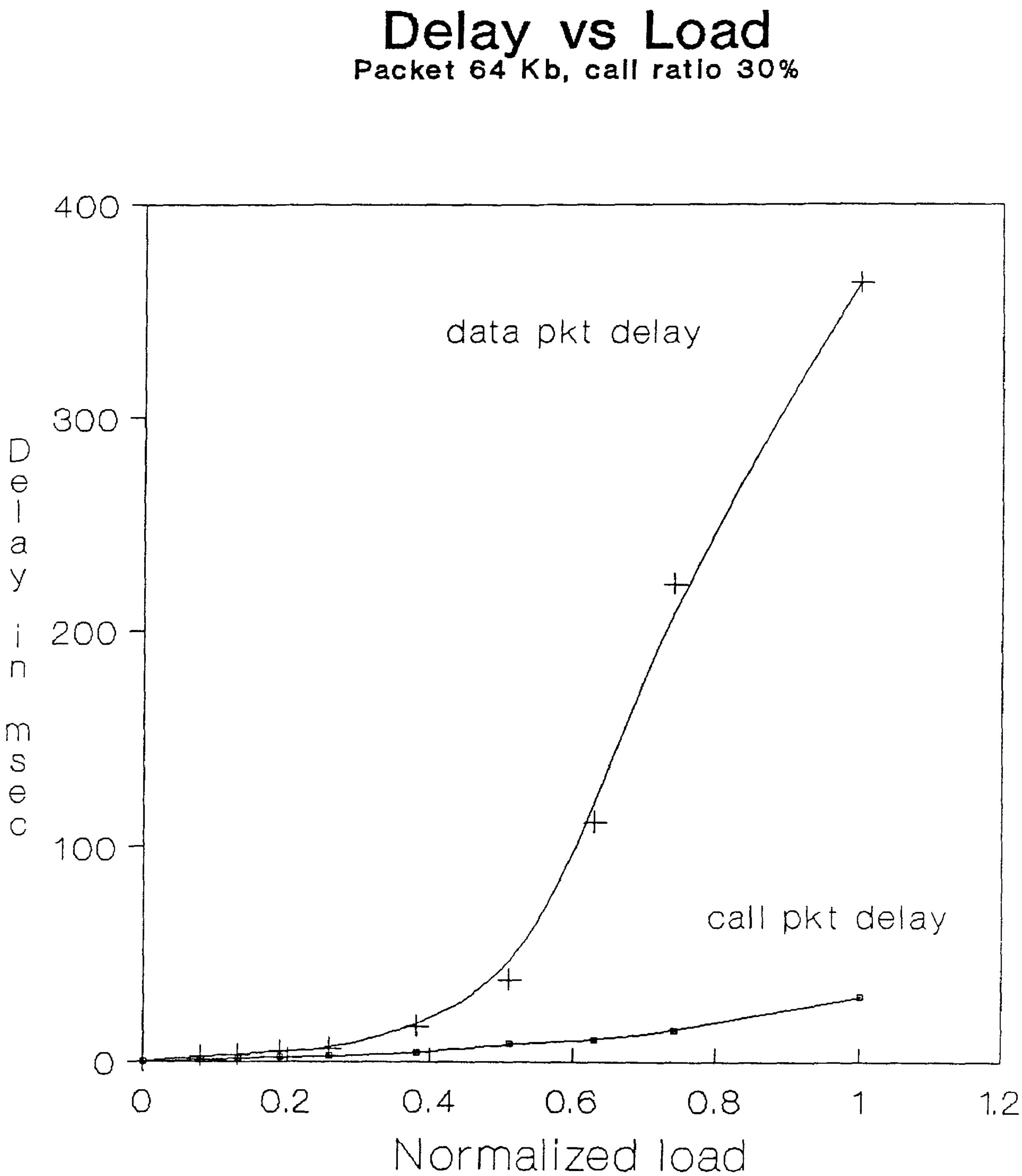

Delay vs Load
Packet 64 Kb, call ratio 30%
400
300
200
100
0
Delay in msec
data pkt delay
call pkt delay
0
0.2
0.4
0.6
0.8
1
1.2
Normalized load

Figure 3.7:

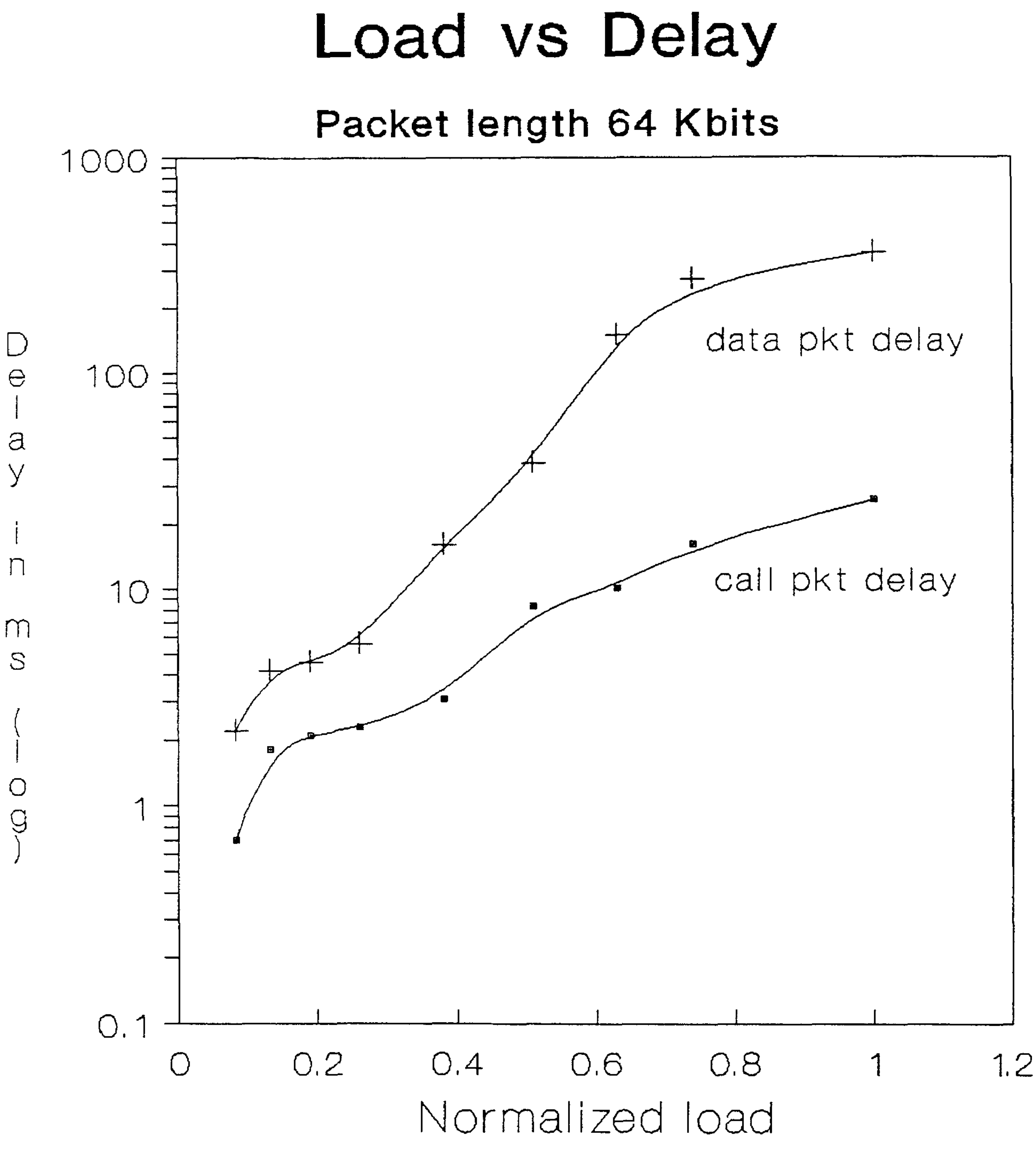

Load vs Delay
Packet length 64 Kbits
1000
100
10
1
0.1
D e l a y i n m s (l o g)
data pkt delay
call pkt delay
0
0.2
0.4
0.6
0.8
1
1.2
Normalized load

Figure 3.8:

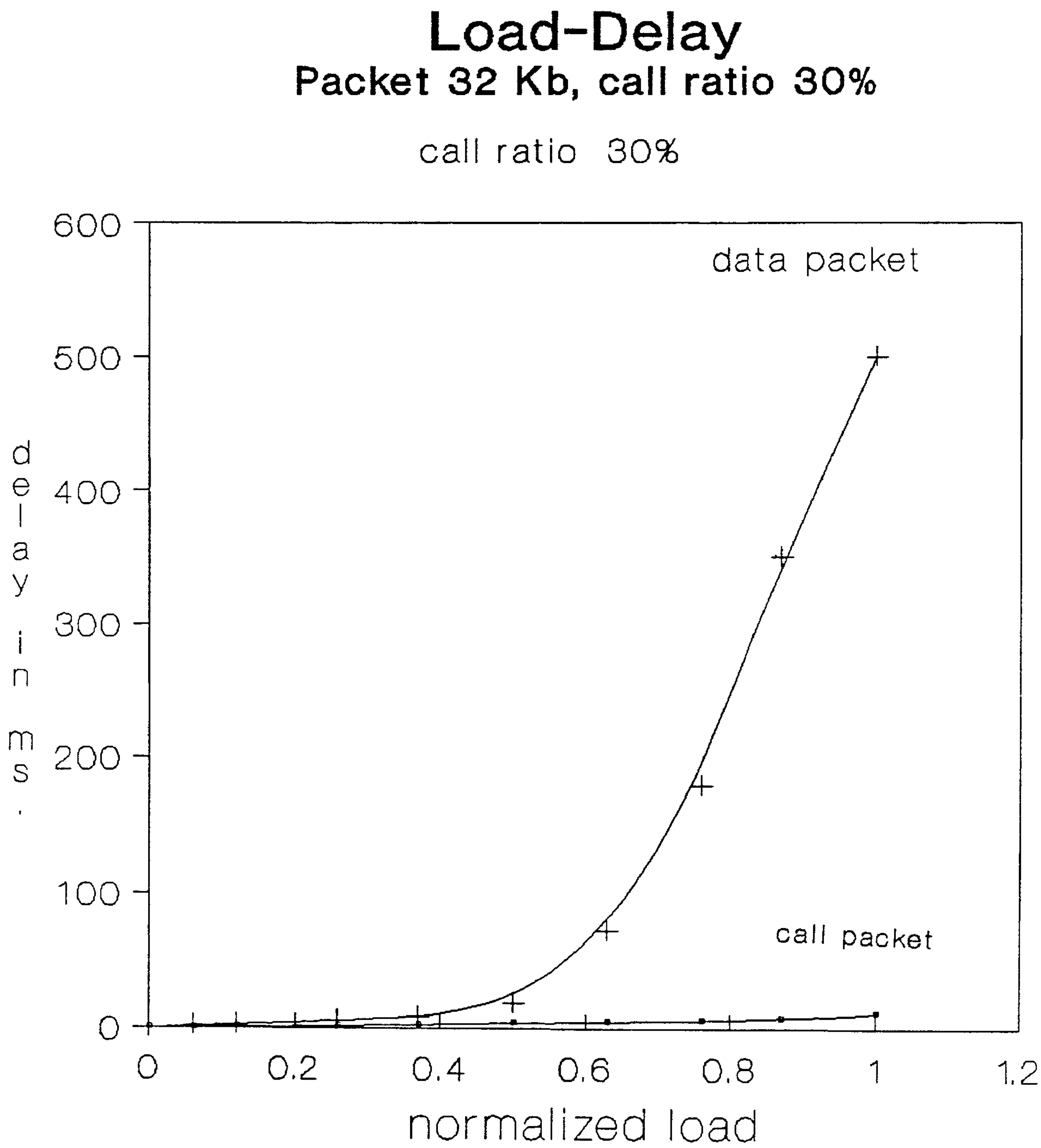

Load-Delay
Packet 32 Kb, call ratio 30%
call ratio 30%
data packet
call packet
delay in ms.
normalized load
600
500
400
300
200
100
0
0
0.2
0.4
0.6
0.8
1
1.2

Figure 3.9:

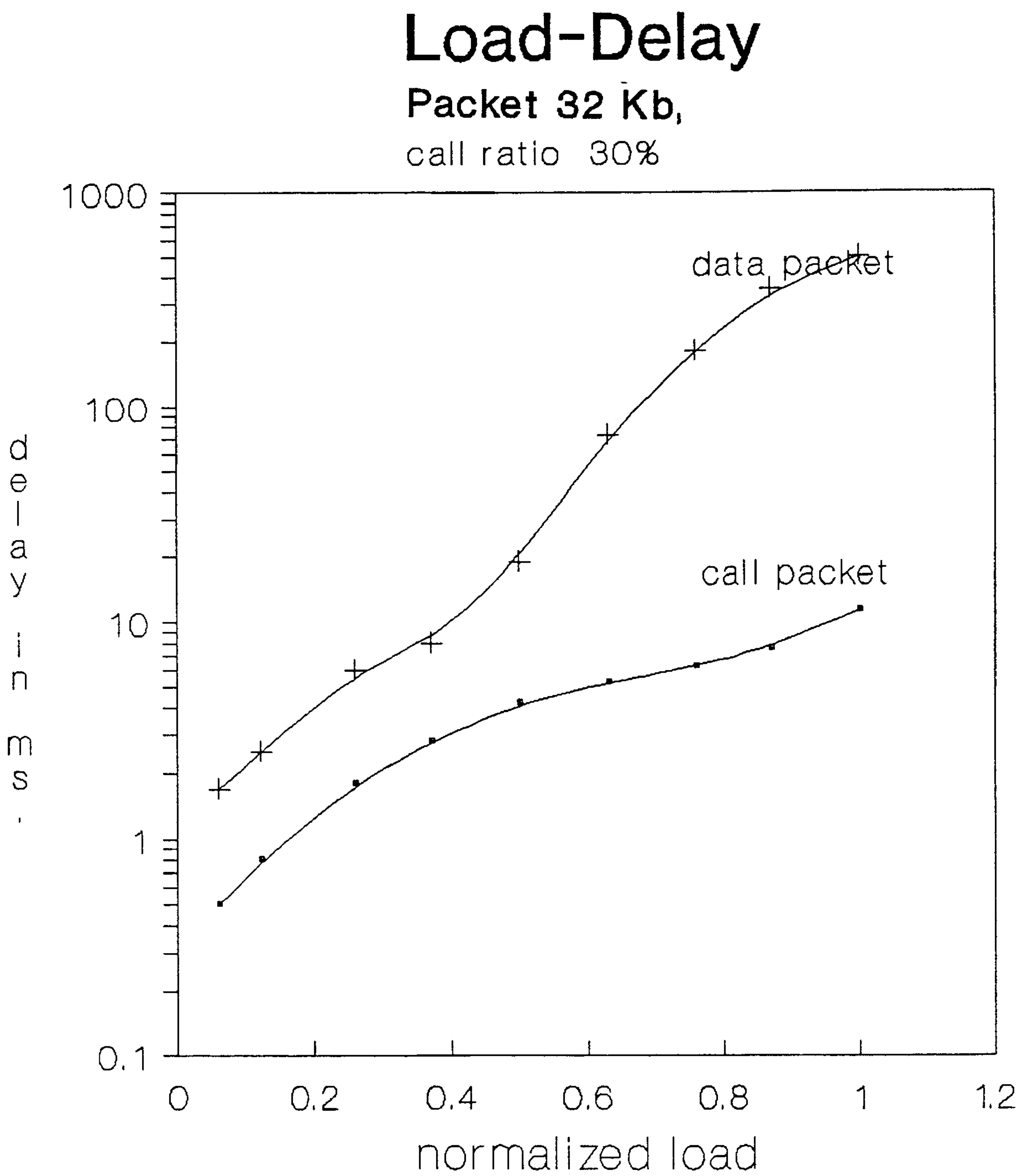

Load-Delay
Packet 32 Kb,
call ratio 30%
1000
100
10
1
0.1
delay in ms.
data packet
call packet
0
0.2
0.4
0.6
0.8
1
1.2
normalized load

Figure 3.10:

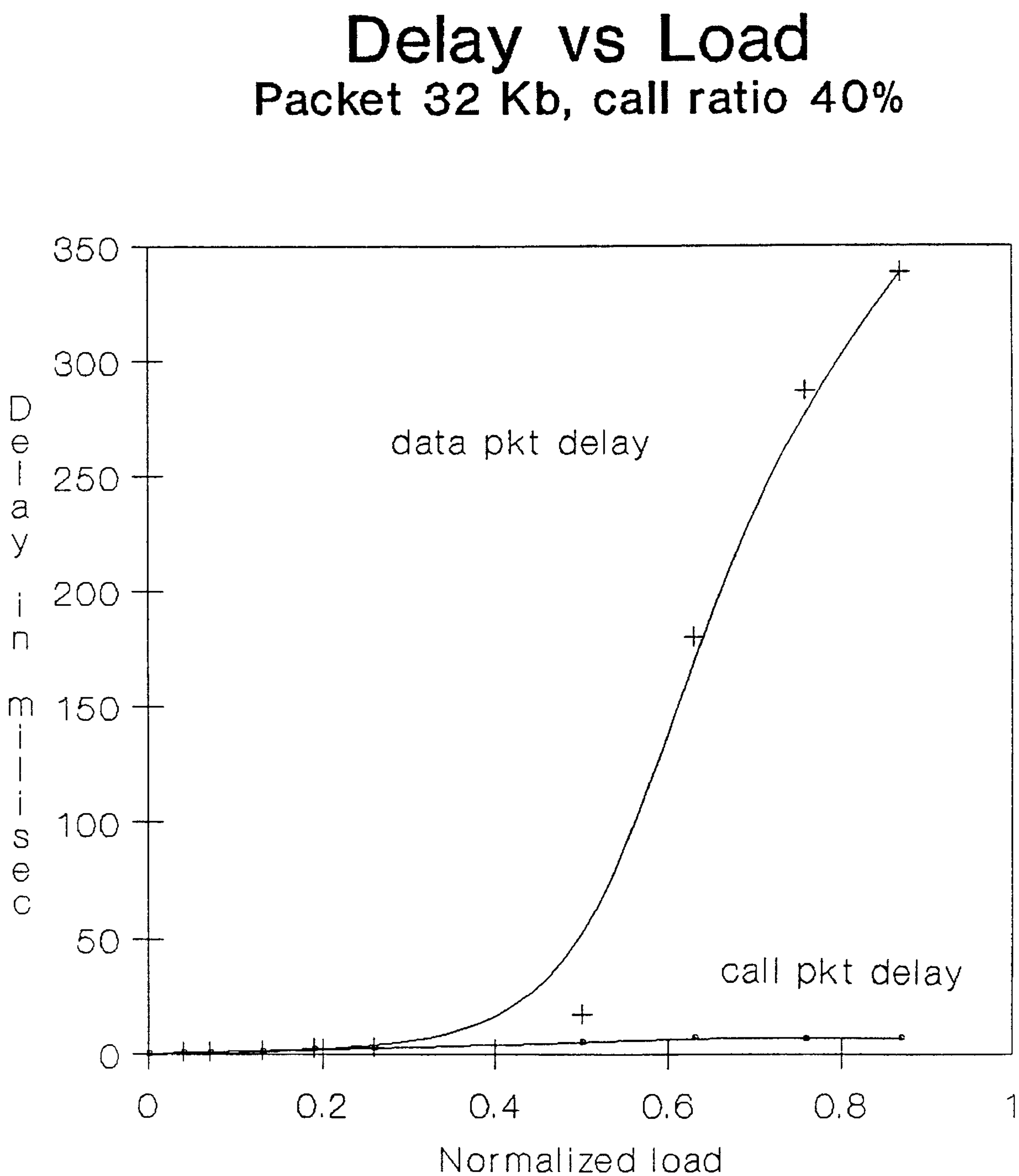

Delay vs Load
Packet 32 Kb, call ratio 40%
Delay in millisec
350
300
250
200
150
100
50
0
data pkt delay
call pkt delay
0
0.2
0.4
0.6
0.8
1
Normalized load

Figure 3.11:

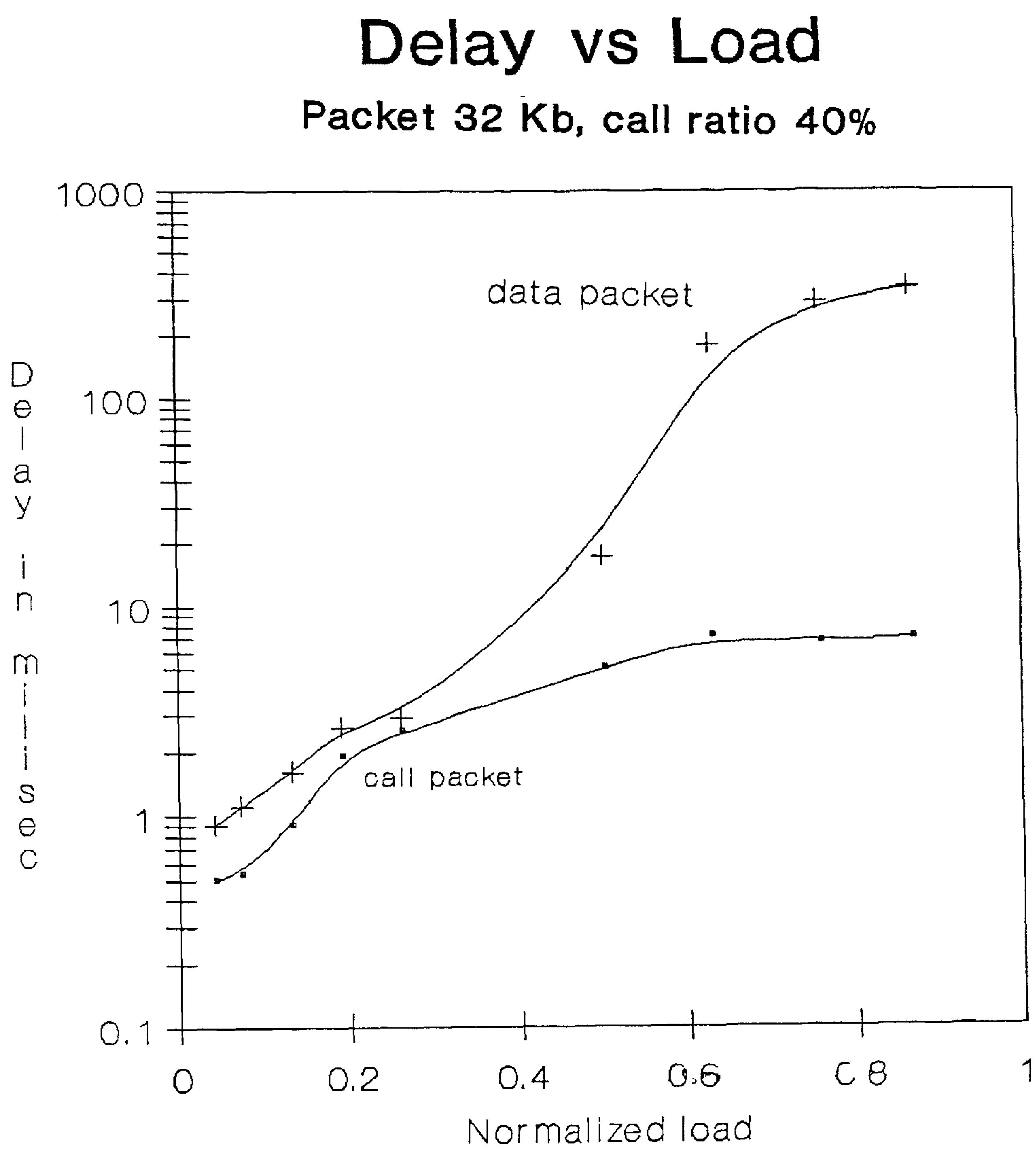

Delay vs Load
Packet 32 Kb, call ratio 40%
data packet
call packet
D e l a y i n m i l l i s e c
1000
100
10
1
0.1
0
0.2
0.4
0.6
0.8
1
Normalized load

Figure 3.12:

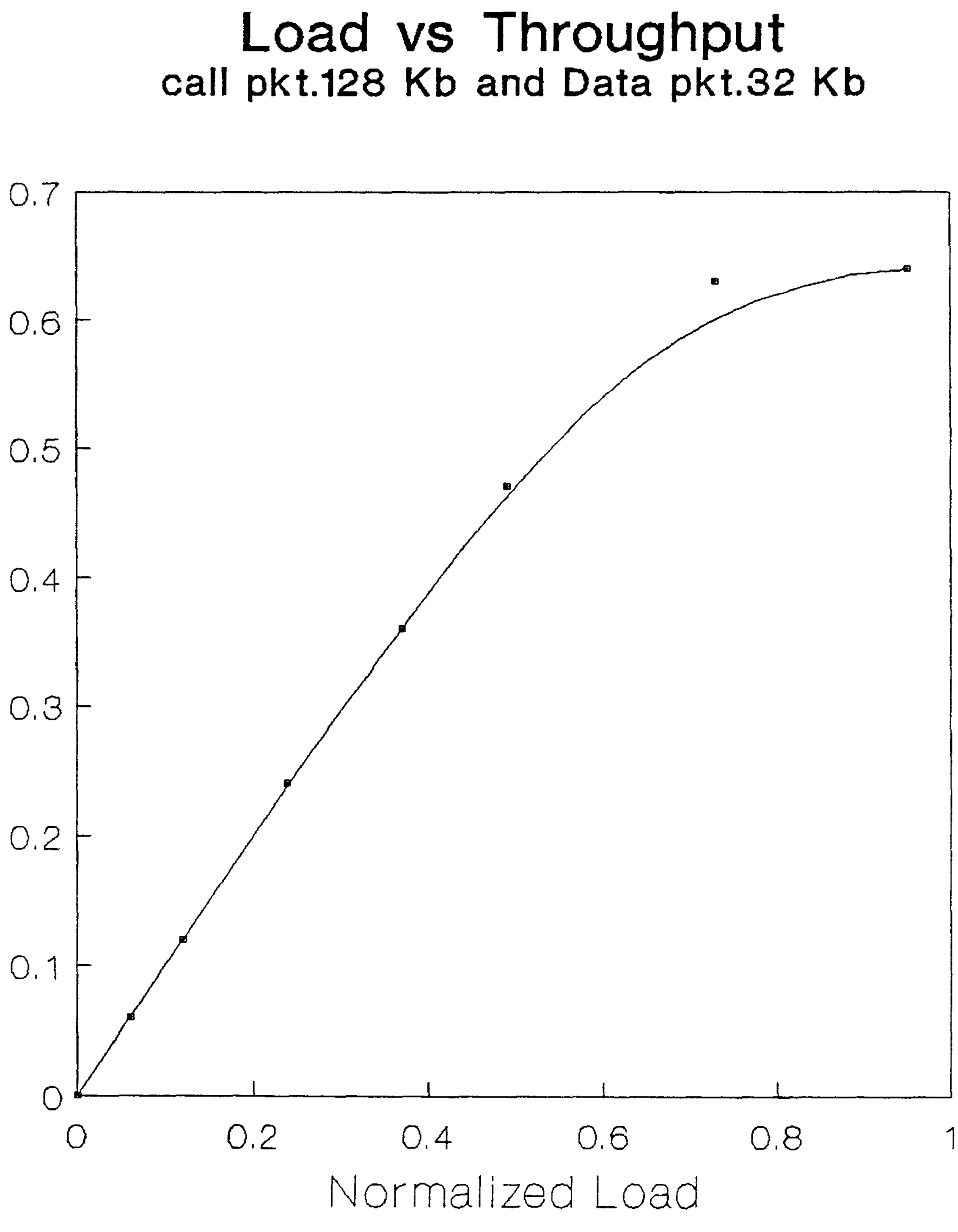

Load vs Throughput
call pkt.128 Kb and Data pkt.32 Kb
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
Normalized Load

Figure 3.13:

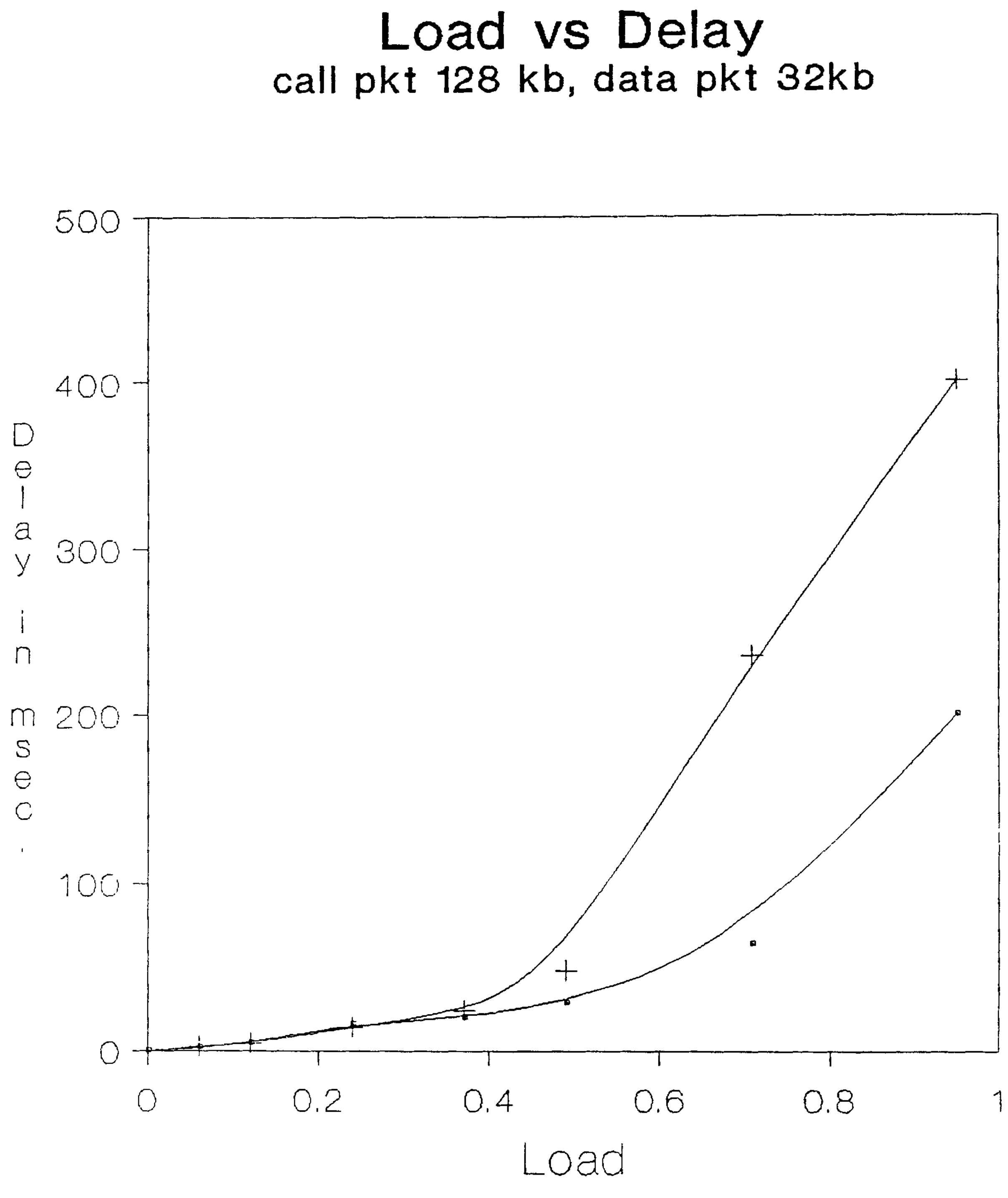

Load vs Delay
call pkt 128 kb, data pkt 32kb
500
400
300
200
100
0
Delay in msec.
0
0.2
0.4
0.6
0.8
1
Load

Figure 3.14:

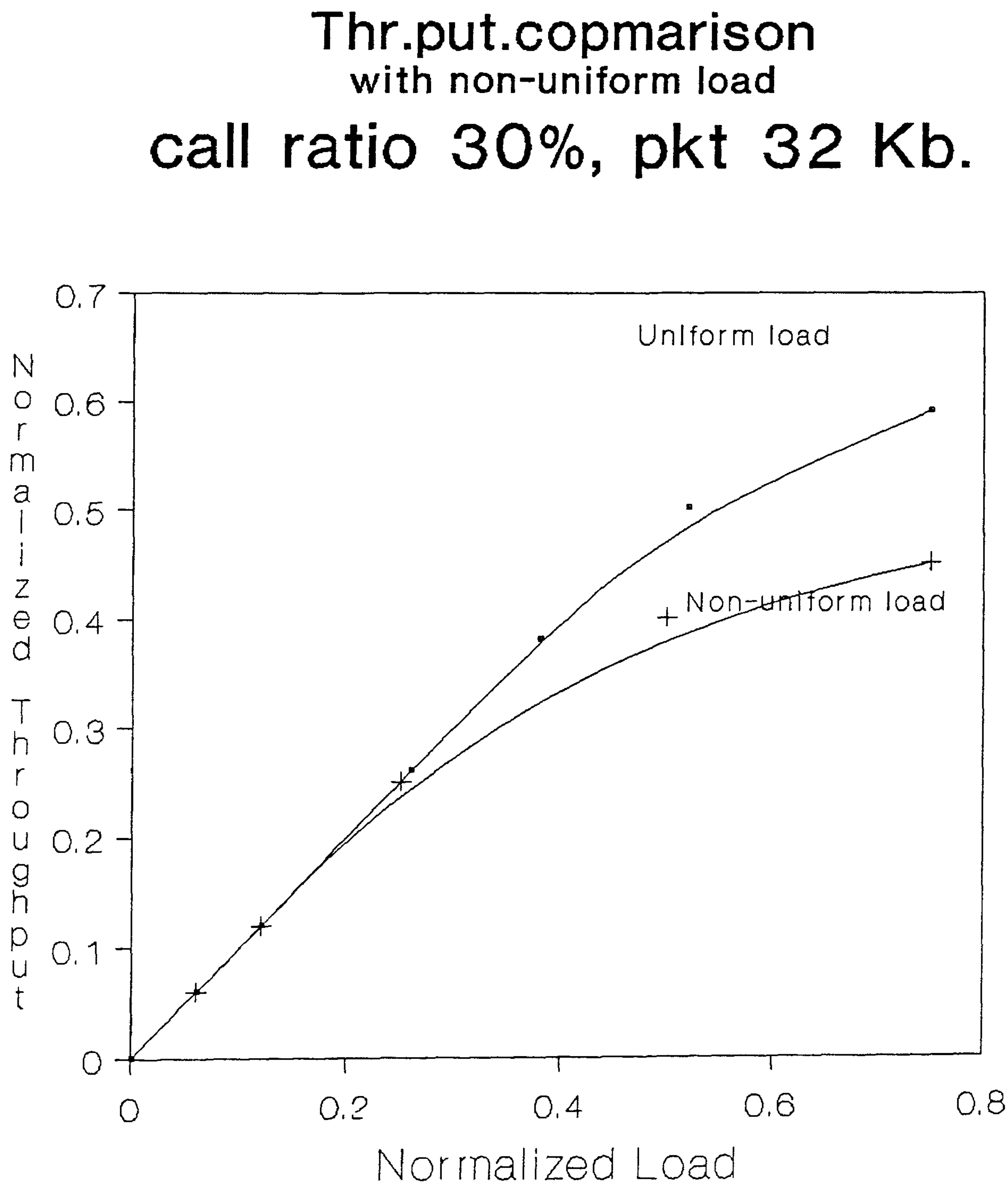
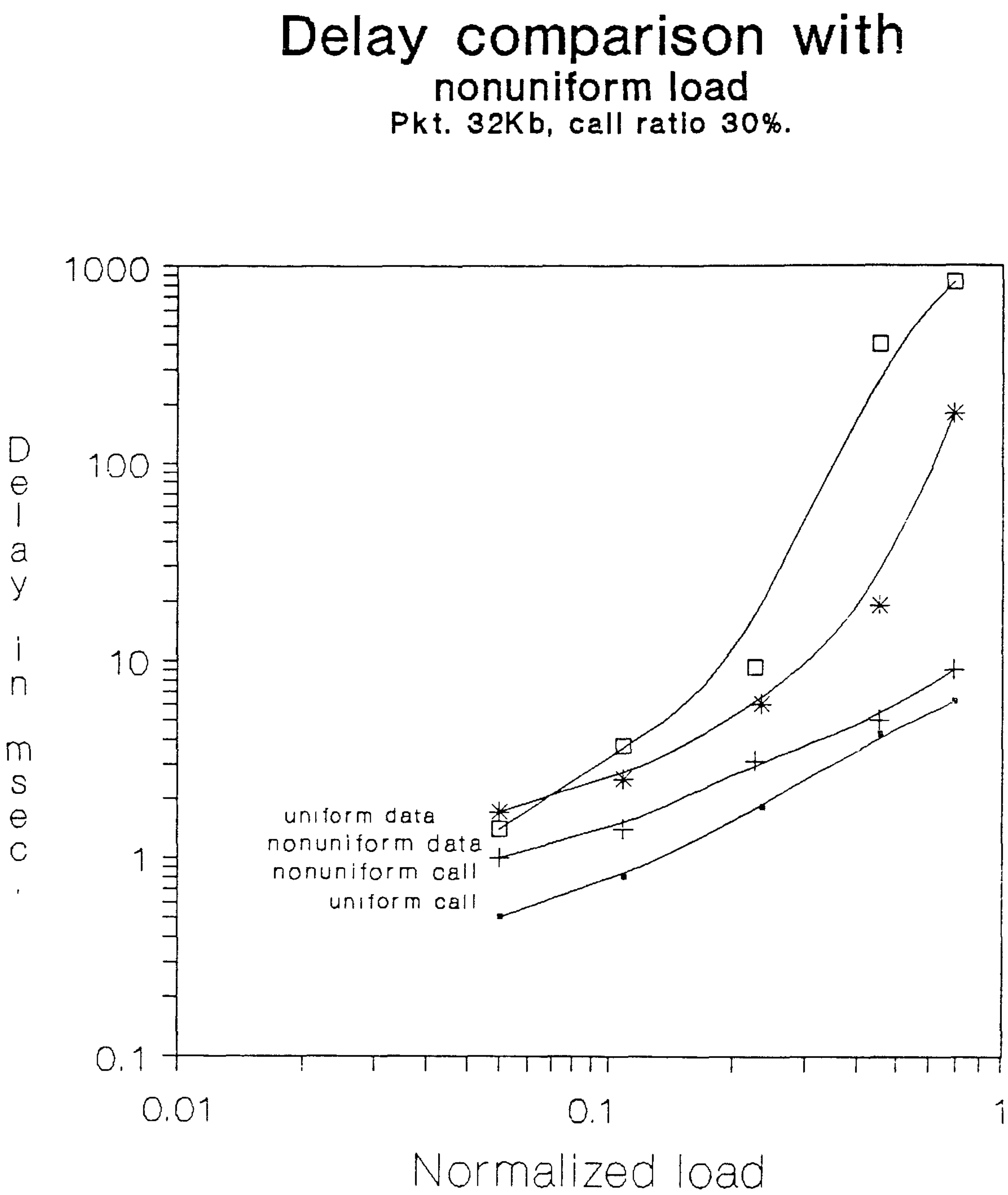

Thr.put.copmarison
with non-uniform load
call ratio 30%, pkt 32 Kb.
Uniform load
Non-uniform load
Normalized Throughput
Normalized Load
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
0.2
0.4
0.6
0.8

Figure 3.15:



Delay comparison with
nonuniform load
Pkt. 32Kb, call ratio 30%.
1000
100
10
1
0.1
Delay in msec.
0.01
0.1
1
Normalized load
uniform data
nonuniform data
nonuniform call
uniform call

Figure 3.16:

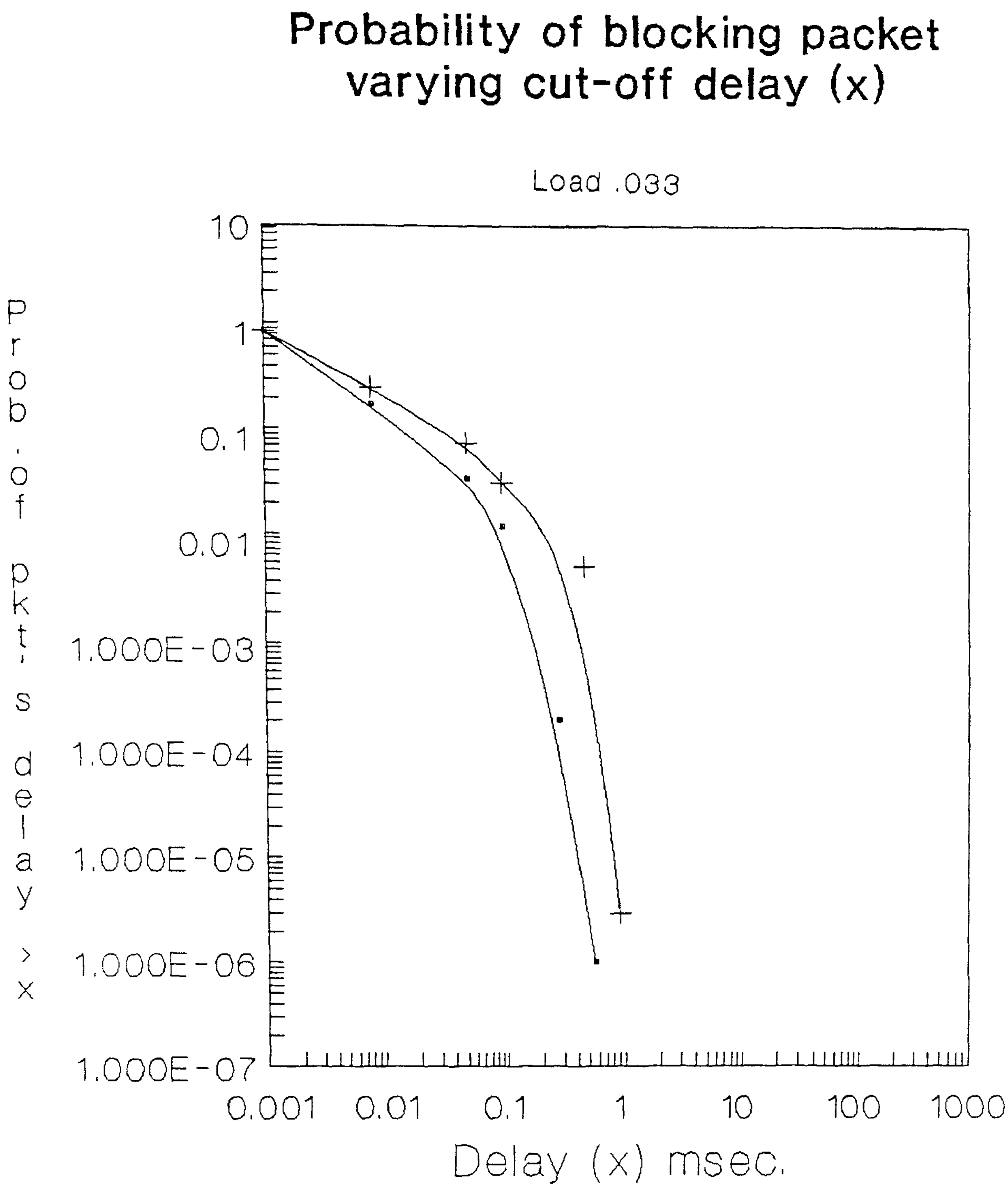

Probability of blocking packet
varying cut-off delay (x)
Load .033
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
Prob.of pkt's delay > x
0.001
0.01
0.1
1
10
100
1000
Delay (x) msec.

Figure 3.17:

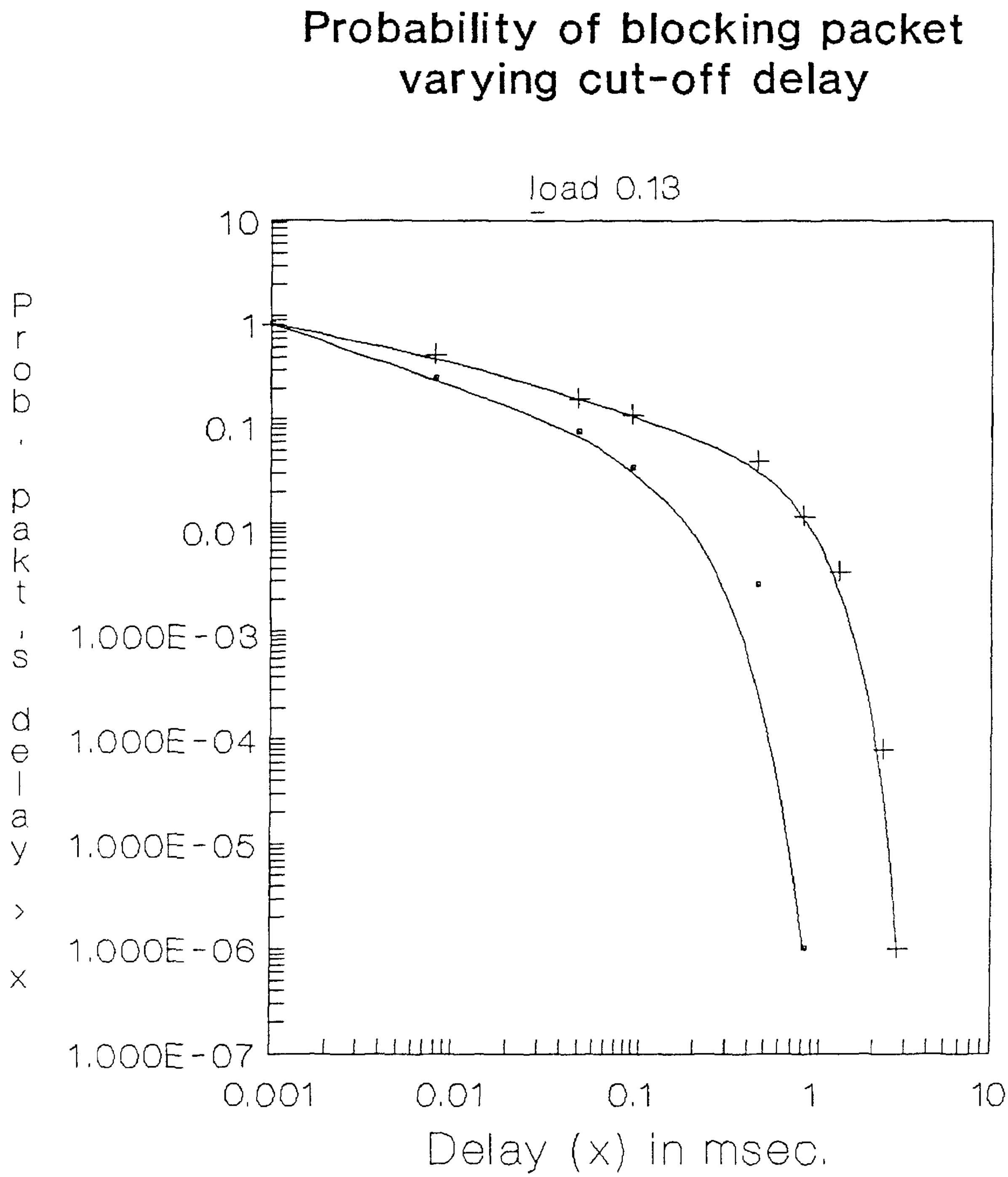
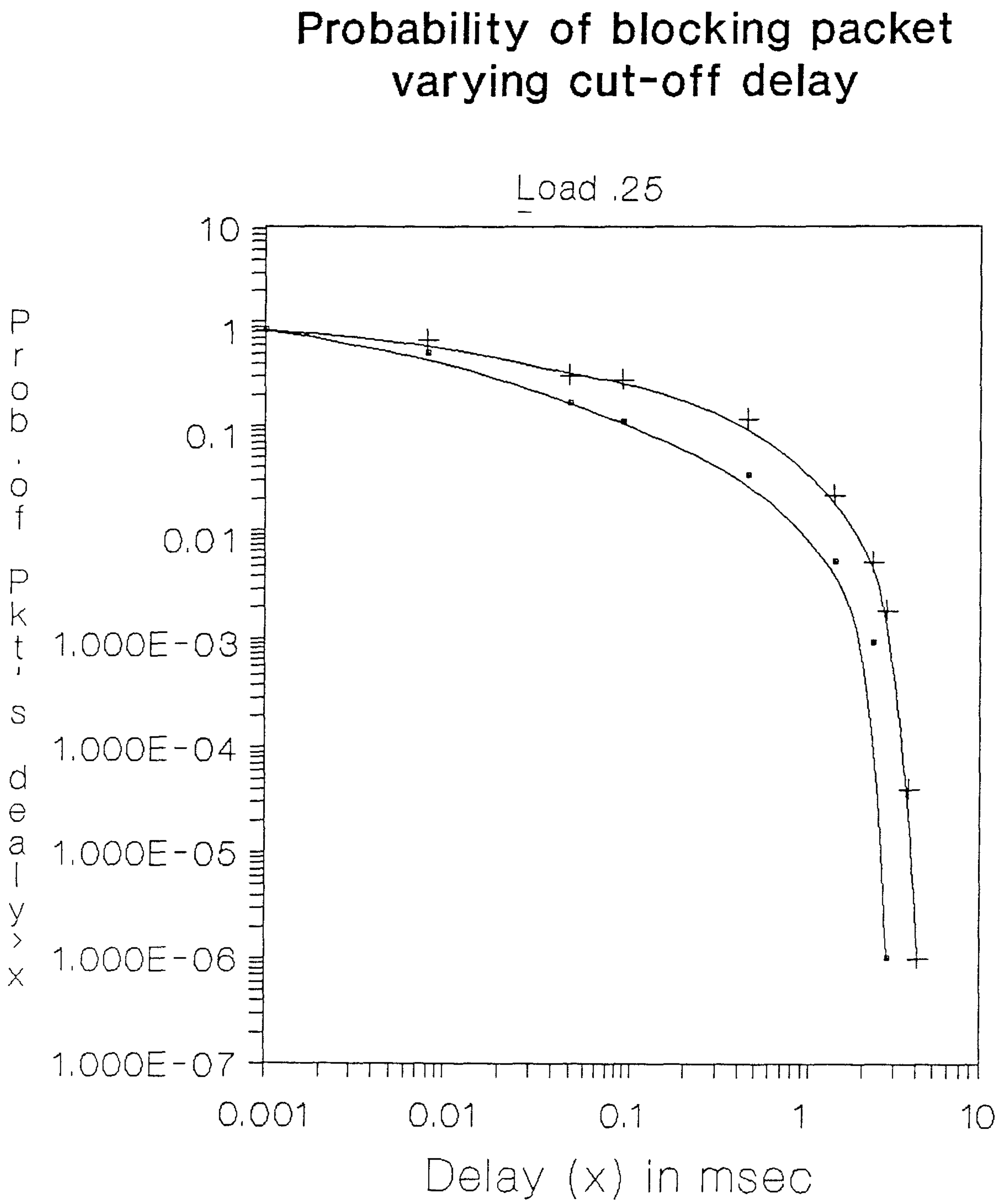

Probability of blocking packet
varying cut-off delay
load 0.13
Prob. pakt.'s delay > x
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
0.001
0.01
0.1
1
10
Delay (x) in msec.

Figure 3.18:



Probability of blocking packet
varying cut-off delay
Load .25
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
Prob. of Pkt's dealy>x
0.001
0.01
0.1
1
10
Delay (x) in msec

Figure 3.19:

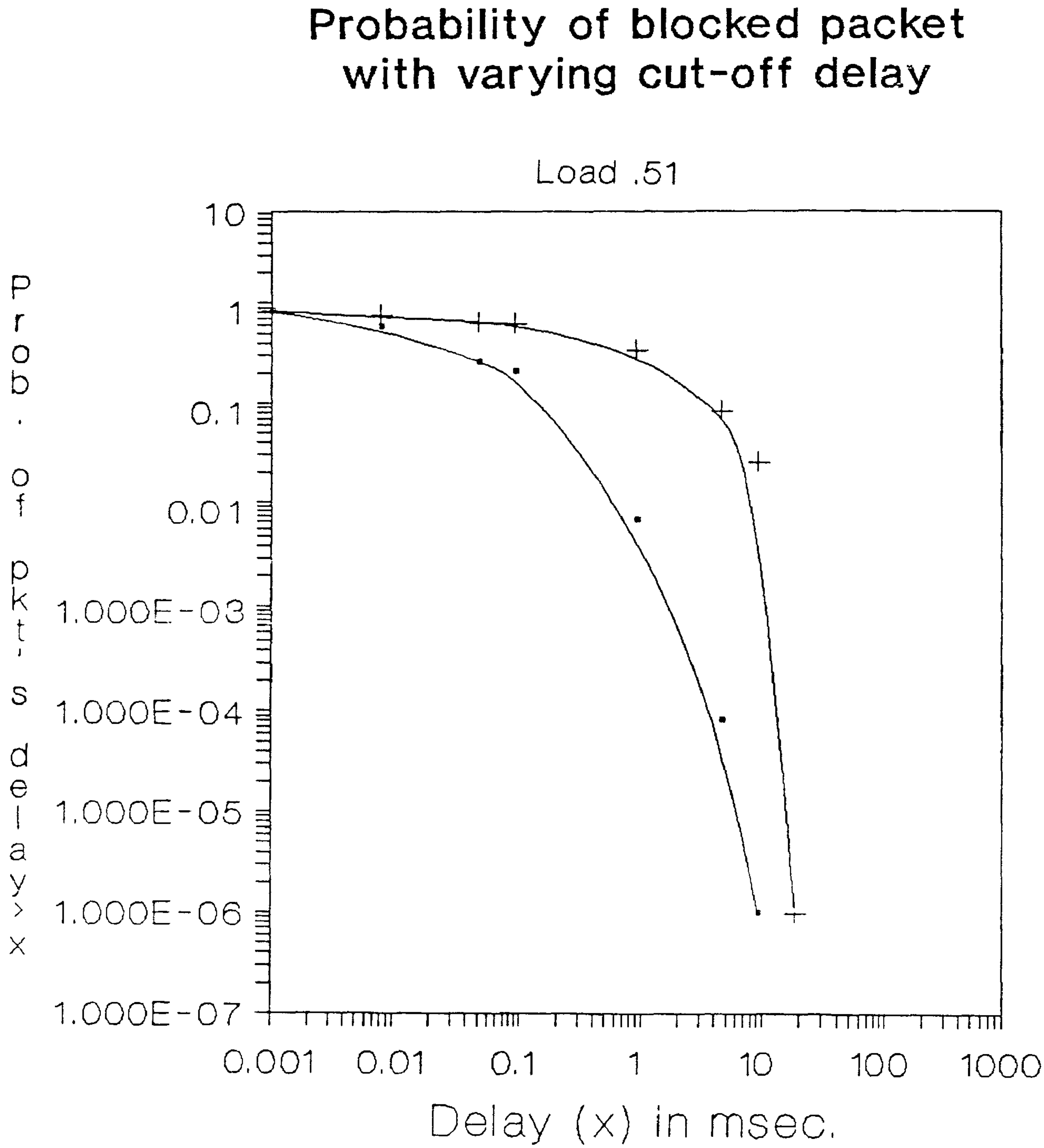

Probability of blocked packet
with varying cut-off delay
Load .51
Prob. of pkt's delay>x
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
0.001
0.01
0.1
1
10
100
1000
Delay (x) in msec.

Figure 3.20:

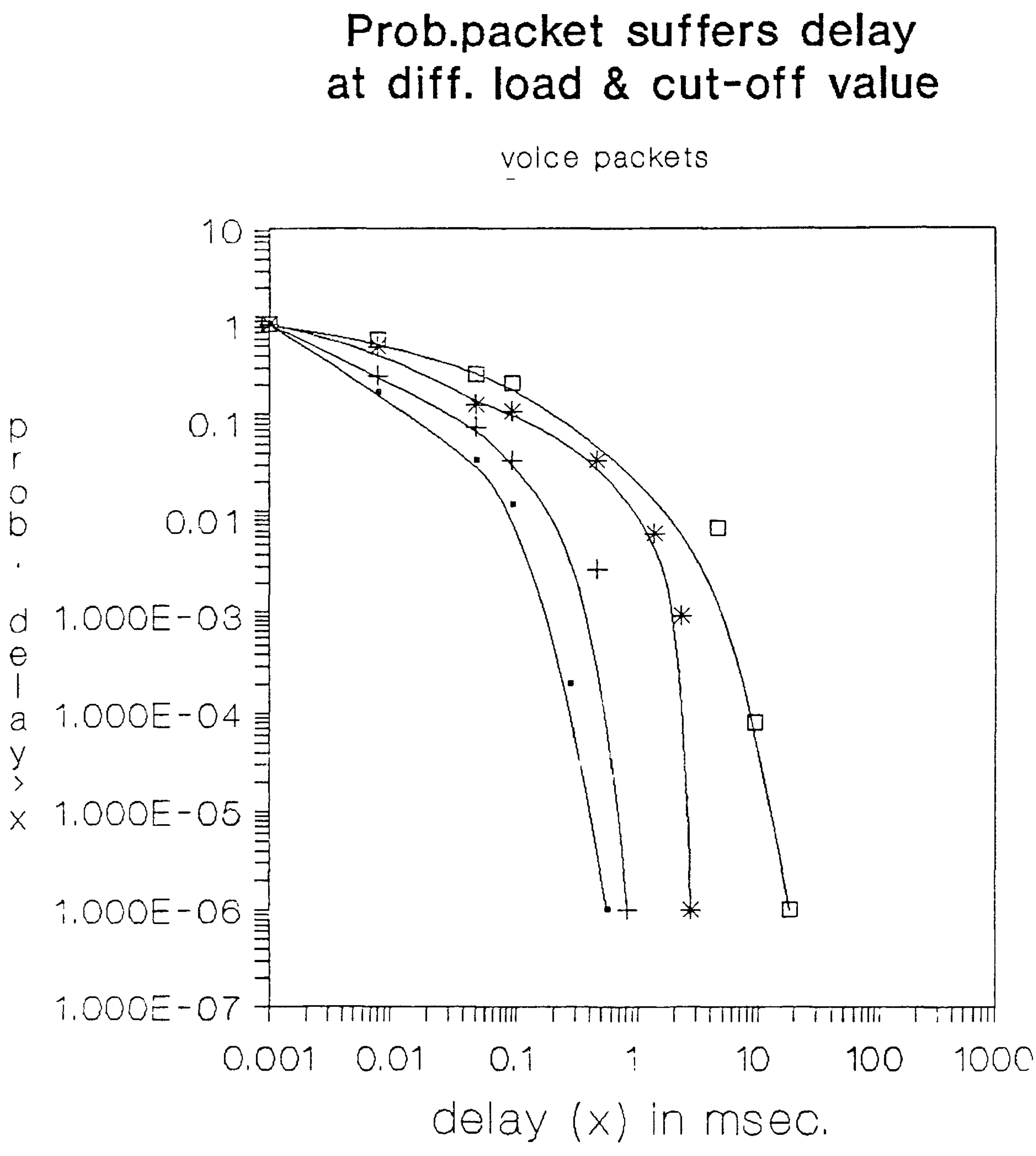

Prob.packet suffers delay
at diff. load & cut-off value
voice packets
prob. delay > x
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
0.001
0.01
0.1
1
10
100
1000
delay (x) in msec.

Figure 3.21:

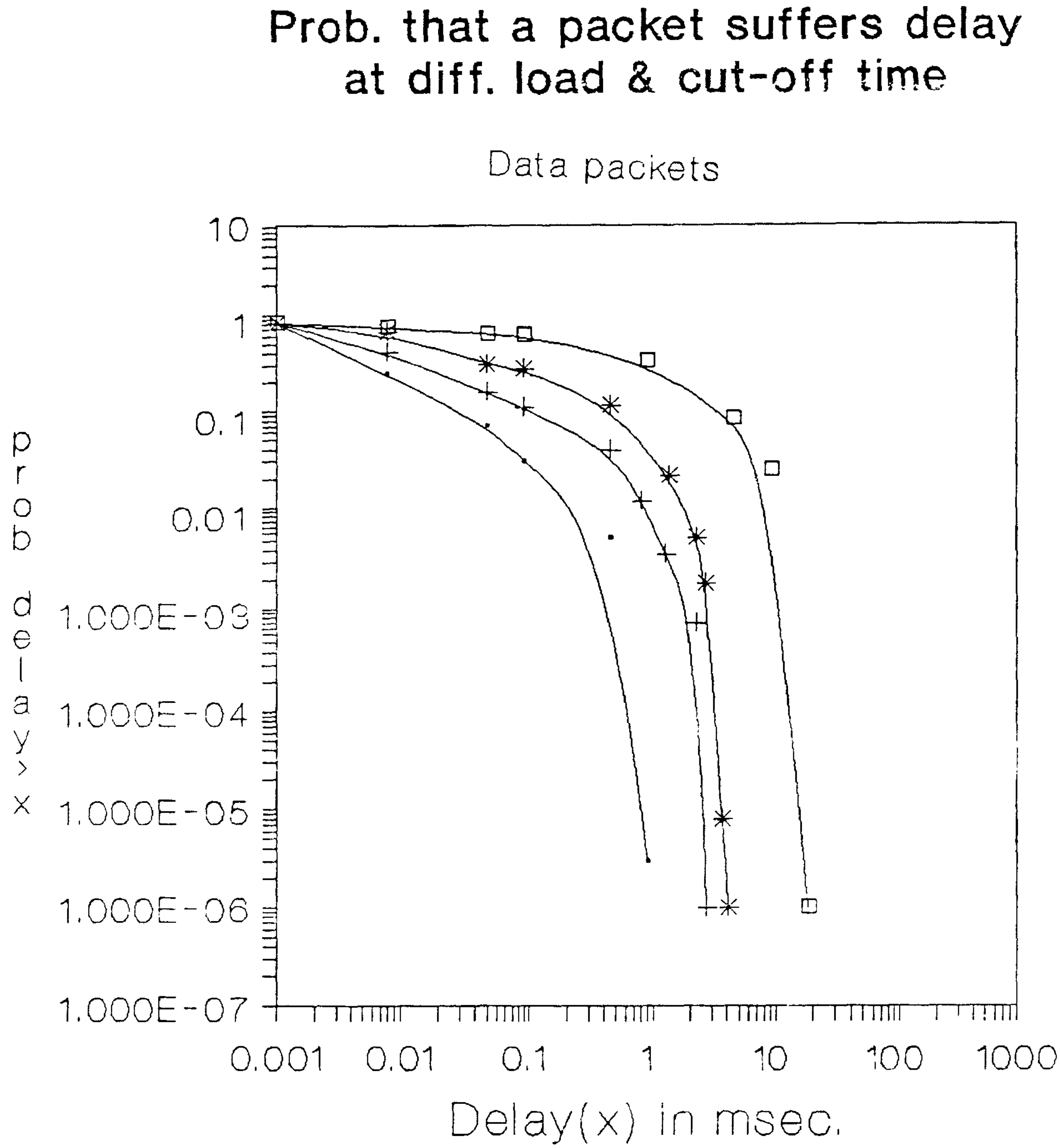

Prob. that a packet suffers delay
at diff. load & cut-off time
Data packets
prob delay>x
10
1
0.1
0.01
1.000E-03
1.000E-04
1.000E-05
1.000E-06
1.000E-07
0.001
0.01
0.1
1
10
100
1000
Delay(x) in msec.

Figure 3.22:

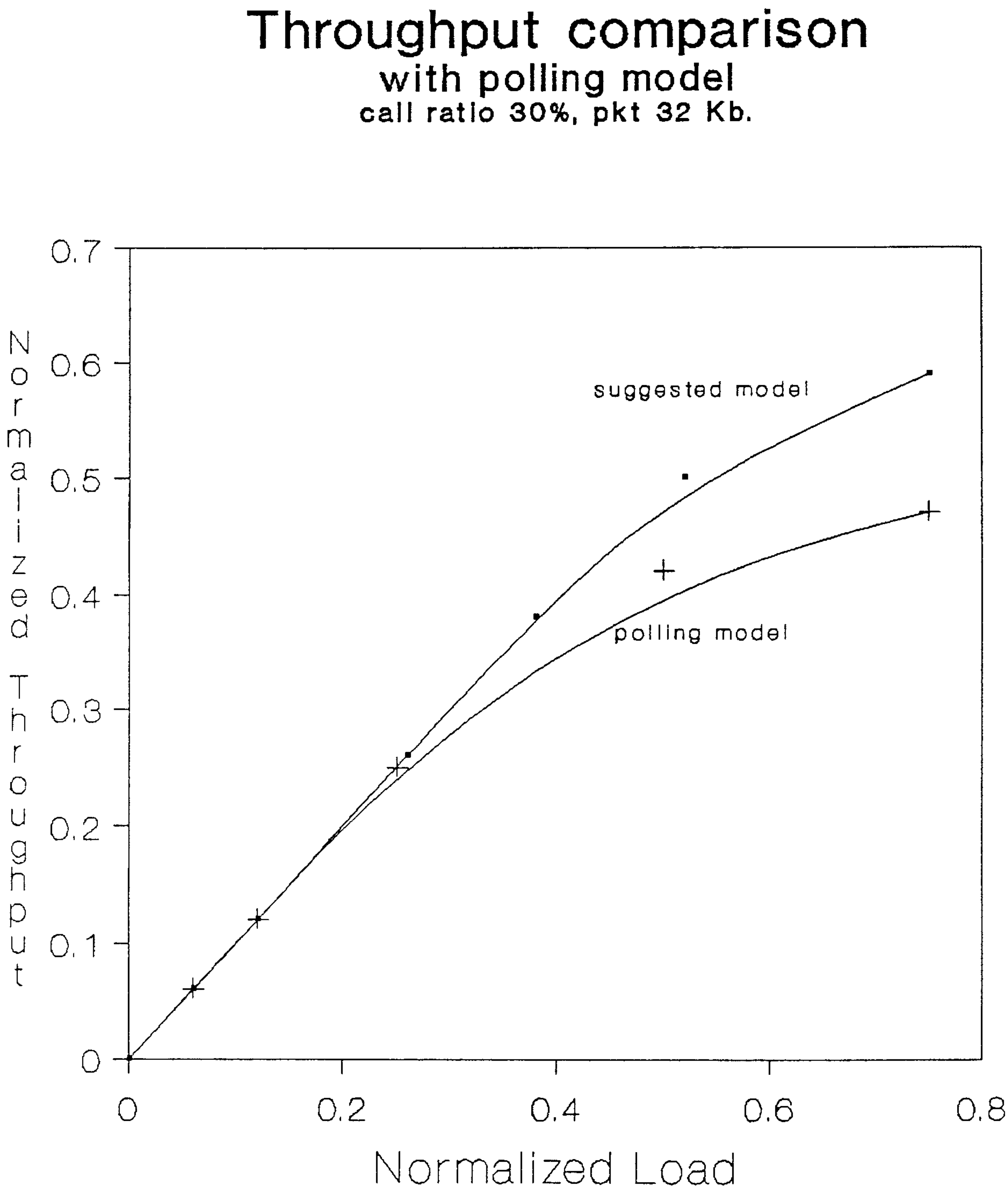

Throughput comparison
with polling model
call ratio 30%, pkt 32 Kb.
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Normalized Throughput
suggested model
polling model
0
0.2
0.4
0.6
0.8
Normalized Load

Figure 3.23:

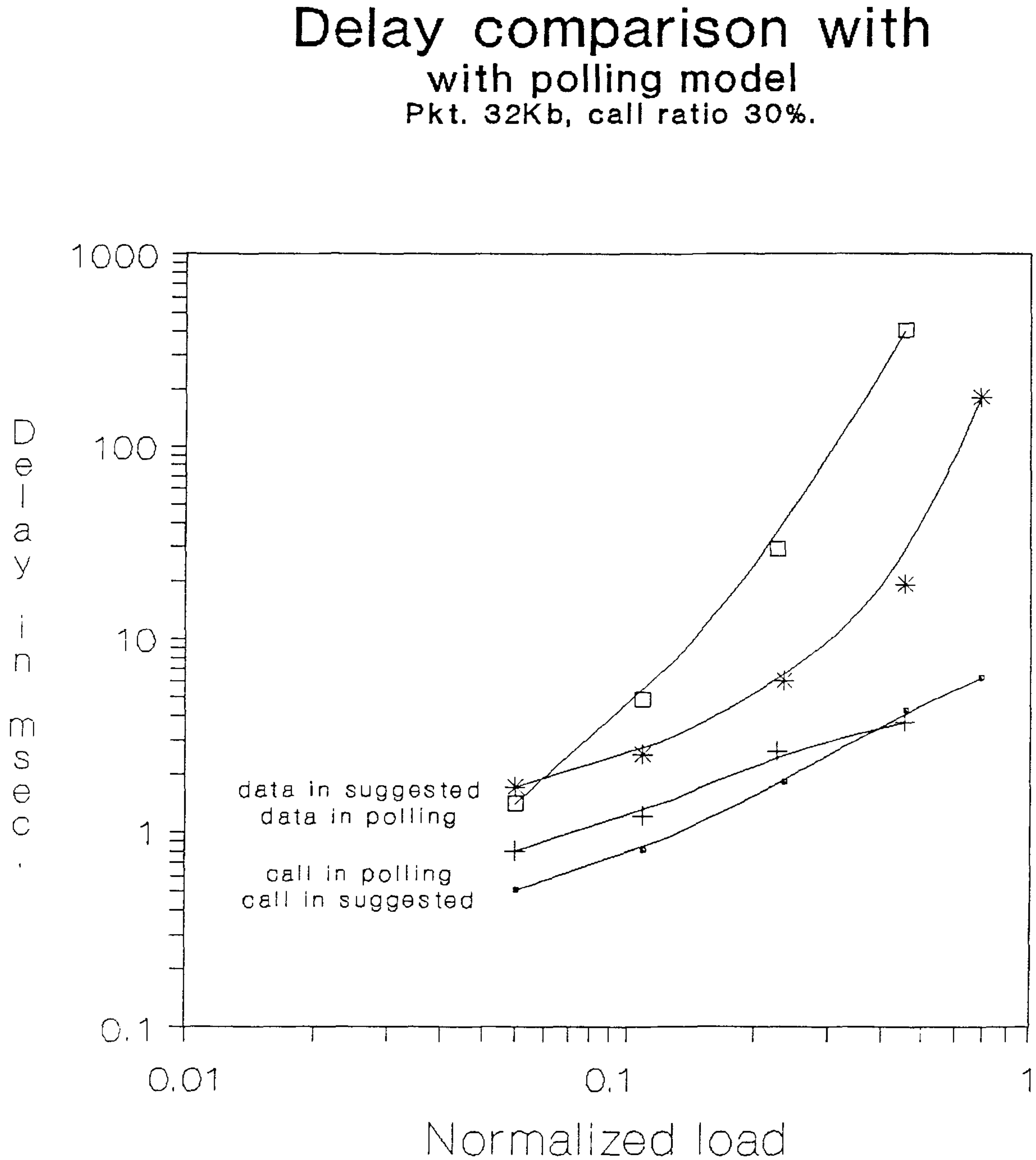

Delay comparison with
with polling model
Pkt. 32Kb, call ratio 30%.
Delay in msec.
1000
100
10
1
0.1
0.01
0.1
1
Normalized load
data in suggested
data in polling
call in polling
call in suggested

Figure 3.24:

Chapter 4

Observation and Conclusions

It has been observed in our results that at any packet length, normalized throughput increases at a constant rate (here it is 1) with normalized load at lower load range and starts saturating from 0.5 load. It reaches saturation at about 60-63% of load.

At a lower load, whatever packets the switch receives, it could switch because of its fast and non-blocking nature. At higher loads, because of blocking by the busy output port, incoming packets gradually start waiting in the input queues and more packets are blocked.

Considering input port and output port utilization it is shown that, with input queueing in a nonblocking switch the maximum throughput that can be achieved is 0.58 when the number of ports is extremely high. With one pair of input-output ports, the throughput is maximum and is linear with the load. In our model we have used 4 pairs of input-output ports and we obtained saturation at about 63% of the normalized throughput.

To have a better throughput-load ratio we should go for queueing at the output port. But in this case, to avoid collision and loss of packets, we must have a mechanism to stagger input packets before switching, then after switching these packets are to be placed in a register temporarily. From the register the packets could then be put in the output queues. This method increases the delay of the packets.

So either we have to sacrifice delay for throughput or throughput for delay. As our model is that of a fast packet switching architechture we can sacrifice some of the throughput.

We notice that at the higher load, the traffic with a larger average packet length has a slightly higher throughput. We can get the same load by using a high arrival rate of small packets or by using a low arrival rate of large packets. If we use the high arrival rate of the small packets virtually we are creating more non-switchable time. This will result in a fewer number of minipackets that could be transmitted and hence the throughput will be smaller than that obtained when the packets are longer and less frequent.

For the case of an increased voice/data ratio we obtained an inrease in throughput. This increase in throughput was expected because voice packets are already of higher priority. The presence of a larger number of voice packets caused an increase of maximum and average delay for voice packets.

Regarding delay we notice that because of priority, call packets have a very low delay when compared to data packets at any load, packet length, and voice/data ratio. For longer packet lengths the delay for both types of packets increase. This is obvious because we are calculating the service and queue time of each parent packet until all its minipackets are dispatched.

To have users-choice data we plot a graph with different cut off values of delay and measure the probability that a packet will be delayed more than this cut-off value. So any user can choose how much chance he/ she can take to lose a packet and what will be the corresponding delay at that tolerance limit. Figures 17-22 of the previous section show this kind of performance measurement with an average packet length of 32 Kbits.

The performance of the alternative method to select input for switching is shown in figures (23 and 24). We can compare the perfomance of this method with our neural net model.

Appendix A

Program Module

In our simulation program we have several function modules which simulate an environment, from generation to dispatch of packets in the form of series of minipackets-*A Fast Packet Switching System* . The programming steps are given in deteails as below:

random

Generate the random numbers which are uniformly distributed.

ran-arr

Genarate packets with random inter-arrival time. The distribution of the inter-arrival time is Poisson.

ran-len

Generate packets with random packet lengths. The packet-lengths are Exponentially distributed. Here the packet-lengths were considered in *Time Scale* . Different values of average packet lengths, such as 64 Kbits, 32 Kbits and 16 Kbits, were considered. For call packets (ie, circuit-switched type packets)

128 Kbits of average packet-length was generated.

ran-in

Generate the random numbers from 0 to 3 for the input port numbers.

ran-out

Generate the random numbers from 0 to 3 for the output port numbers.

ran-bit

In our simulation we were considering two types of packets. We set the priority bit 1 for circuit-switched packets and 0 for data-switched packet. This function generates randomly the bit 1 and 0.

create

This function actually acts like the **SFA** ie, switch fabric adapter that we have explained before. In this function packets are generated with random arrival time, random packet length (in time scale), random priority bit, random input and output port number. After generation the packets are divided into **minipackets** of length (0.000008 sec. including header of 0.0000001 sec) and put in the **call** or **data** queue of the corresponding input port. Each minipacket has its output port number, priority bit, arrival time and the minipacket number of each parent packet. If a packet is divided into say, p, packets then the minipackets of that particular parent packet will be numbered from p to 1, starting from the head-end of the respective queue. This fuction also maintains queue sizes of all the queues ie, 4 call queues and 4 data queues(like call[ ][MAX] and data[ ][MAX] where MAX is the maximum queue size).

dispatch

This function does the job like our queue managers in our model. Here we form a matrix OUTPUT[4][4] whose rows are the output port number and columns are the input port number. As call-packet queue has higher priority, first, for each output (ie, row) the call-queue size of the input (ie, col.) is put in the matrix. If for any output there is no value from call-queues then data-packet-queue size is put, if exists. This forms our request matrix as described in the model. Now for each output only one input port can despatch its packet. If there are more than one columns have their values in the same row of the request matrix then the problem is which input should be selected for switching. The function "neuron" is used to solve this.

neuron

This function actually performs the function of a neural net which works on "one-winner-take-all algorithm". The matrix elements are the external excitations for the neurons at the corresponding position in the neural net. In this functon, it takes each row at a time and through competition it accepts only one input which has maximum equivalent excitation. In case of equal values it does not matter which one will be picked up. So in our model we choose to pick up highest column number ie, the highest numbered input port.

From this matrix we get the configuration matrix which shows us what input ports can be switched at this cycle of switching operation. The switching positions are marked by 1.

othermodel

To compare our neural model with other exixting model, say polling, we use this function. In this function, each output polls all the inputs in ascending order of numbers to look for any packet for its own. Here, too, call queues are selected first and then data queues. If any eligible input queue is found it stops searching other input ports. Next output starts looking then. Ultimately matrix like our request matrix is formed. Only differnce with the previous one is that, here, only one input queue can take possession. Therefore, no competition rule exists here and deprive us from selecting the best queue for switching.

served

The configuraton matrix is passed through this function. When it reads 1 in a row of the configuration matrix, corresponding input has the permission to despatch its packet. First call-packet-queue is checked whether it has any packet or not. If not, data-packet-queue is selected. At each switching at most 4 minipackets, either call packet or/and data packet, can be switched. While dispatching, this function keeps track of the division number of the mini packets. When it encounters value 1 it can sense that one main parent packet has been despatched. So it can record total service and delay time of each parent packet.

update

After dispatching minipackets the queues are to be updated. This function pushes the minipackets from tail-end(ie, incoming-end) towards the head-end (ie, outgoing end) of the queues which have been served in the last switching operation.

main

In real field, the switch as keeps on switching packet as well keeps on checking if it has received any packet. If reeived, the queues are to be updated before switching. So in our simulating function, after creating packet it checks how many minipacket are in the parent packet and how many times it can switch before the next packet arrives in. If inter-arrival time is less than the one minipacket service time (ie, 0.000008 sec for 32 Mbits port capacity and 32 byte long minipacket), before switch can operate it accepts another stream of minipackets from the input buffer. Otherwise, it switches minipackets from the enabled input queues as many times as number of minipacket can be accomodated in that inter arrival time.

Bibliography

[1] Peter Bocker, "ISDN The Integrated Services Digital Network, concepts, method,system",*Spriger-Verlag*, Berlin, 1987.

[2] Mischa Schwartz, "Telecommunication. Networks protocol,modelling and analysis", *Addison Wesley*, Massachusets, 1987.

[3] Andrew S. Tanenbaum, "Computer Networks", *Prentice Hall*, New Jersey, 1988.

[4] Abdellatif Marrakchi, Terry Trondent. "A neural Net arbitrator for large Crossbar Packet Switches",*IEEE Trans. on Circuits and Systems*, Vol-36,No 7,pp 1039-41,July 1989,.

[5] Timothy X. Brown, "Neural Networks for Switching", *IEEE Communication Magazine*,pp 72-81, Nov 1989.

[6] Richard P. Lippman, "An intriduction to computing with neural nets", *IEEE ASSP Magazine*, pp 4-20, April 1987.

[7] E Majani, R Erlanson, Y Abu-Mustafa. "On K-Winners-Take-All Networks", *Advances in Neural Information System 1*, pp 634-43, Morgan Kaufman 1989.

[8] Atsuhi Hiramarsu. "ATM Communications Network Control by Neural Networks", *IEEE Trans. on Neural Networks*, Vol 1,no 1, pp 122-30, May 1990.

[9] Dr. D. W. Jacobson, "Design and Implementation of an Integrated Voice, Data and Video Services Network", *IEEE Pacific Rim Conference on Communication, Computers and Signal Processing*, pp 500-04, June 1989.

[10] Hamid Ahmedi, Wolfgang Denzel, Charter Murphy and Erich Porter, "A high Performance Switch Fabric for integrated Circuit and Packet Switching",*IEEE INFOCOM*, pp 0009-0018, 1988.

[11] Dan R. Spears, "Broadband ISDN Switching capabilities from a Services Perspective", *IEEE Journal on Selected Areas in Communication*, Vol SAC-5,no. 8, pp 1222-30, Oct,1987

[12] P Newman, "A Broadband Packet Switch for Multi Service Communication", *IEEE INFOCOM*, pp 0019-0028, 1988.

[13] Riccardo Milen and Jonathon turner, 'Non Blocking Networks for Fast Packet Switching", *IEEE GLOBCOM* pp 548-57,1989.

[14] J. S. C. Chen and R Guerin , "Input Queueing of an Internally Non-blocking Packet Switch with Two Priority Classes", *IEEE GLOBCOM*, pp 529 35, 1989

[15] Mark J. Carol, Michael G. Hluchyj and Moprgan, "Input vs Output Queueing on a Space Division Packet Switch", *IEEE Trans. on Communication*, Vol COM-35, No12, pp 1347-56, 1987.