ANALYTICAL SOLUTION OF QUEUES IN THE NODE INTERFACES FOR COMPUTER NETWORKS

A Project
by

Abu Sayed Md.Latiful Hoque

Submitted to the Department of Computer Science and Engineering of Bangladesh University of Engineering and Technology, Dhaka in partial fulfilment of the requirements of the degree

of

Call No.
001.64404
1991
ABU

POST-GRADUATE DIPLOMA IN COMPUTER SCIENCE AND ENGINEERING

AIT-BUET PROGRAMME

DECEMBER, 1991

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

001.64404
1991
ABU

Analytical Solution of Queues in the Node Interfaces for Computer Networks

A project
submitted by

Abu Sayed Md. Latiful Hoque

Roll No. 891804, Registration No. 80052
For the partial fulfillment of the degree of
Post-Graduate Diploma in Computer Science & Engineering.
Examination held on: December 28, 1991.

Approved as to style and content by:

[illegible] 28/12/91

i. Dr. Md. Shamsul Alam
Associate Professor,
Department of CSE,
BUET, Dhaka.

Chairman
and
Supervisor

[illegible]

ii. Dr. Mohammad Kaykobad
Assistant Professor,
Department of CSE,
BUET, Dhaka.

member

[illegible]

iii. Dr. M. Rezwan Khan
Associate Professor,
Department of EEE,
BUET, Dhaka.

member

Acknowledgment

The author is pleased to have the opportunity to express his gratitude to a number of people who have helped so much for the fulfillment of the project work.

The author is really owed to Dr. Md. Shamsul Alam, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka. The project work has been completed under his supervision. His keen interest and sincerity and the scholarly guidance have made it possible to complete the project work.

It would be ungrateful if the author does not express his utmost gratitude to Dr. Syed Mahbubur Rahman, Head of the Department of Computer Science and Engineering. His consistent encouragement has made it easier to complete the project.

The author also expresses his heart felt thanks for valuable discussions occasionally made with Dr. Mohammed Kaykobad, Assistant professor, Department of Computer Science and Engineering.

Finally the author wishes to express his indebtness to UNDP for providing financial support to the project work.

Abstract

The performance of a computer network depends on the performance of the network interface unit. In this thesis, analytical queuing models have been used for the study of the performance indices of the node interfacees for computer networks. Here the performance indices are throughput, utilization and the delay of the queuing network. Buzen algorithm has been used for the solution of single-chain queuing network with a number of processors and a number of controllers and Mean value analysis has been used for the solution of multi-chain queuing network. The effect of number of controllers and protocol processors on the performance indices have been studied.

Table of Contents

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 AN INTRODUCTION TO COMPUTER NETWORK

Although the computer industry is young compared to other industries (i,e automobiles,air transportation etc.), computers have made spectacular progress in a short time. During the first two decades of their existence, computer systems were highly centralized, using within a single room.

The merging of computers and communications has had a profound influence on the way the computer systems are organized. The concept of the "computer center" as a room with a large computer to which users bring their work for processing is rapidly becoming obsolete. This model has not one but at least two flaws: the concept of single large computer doing all the work and the idea of users bringing work to the computer instead of bringing the computer to the user.

The old model of a single computer serving all of the organization's computational needs is rapidly being replaced by one in which a large number of separate but interconnected computers do the job. These systems are called computer networks. There is a considerable confusion in the literature between a computer network and a distributed systems. The key distinction is that in a distributed system the existence of multiple autonomous computer is transparent to the user. The user of a distributed system is not aware that there are multiple processors;it looks like a virtual uniprocessor. Allocation of jobs to processors and files to disks, movement of files between where they are stored and where they are needed and all other system functions must be automatic. With a network a user must explicitly log onto one machine, explicitly submit jobs remotely, explicitly move files around and handle all the network management.

According to physical location, networks are classified as follows:

a. Local Area Network

Local area network (LAN) generally have three distinctive characteristics:

* A diameter of not more than a few kilometers.

* A total data rate 2-20 Mbps.

* Ownership by a single organization.

Local area networks are widely used because of the following reasons:

A collection of computers, terminals, and peripherals located in the same building or in adjacent buildings, may be allowed to intercommunicate and also allow all of them to access a remote host or other network. In the absence of local network, separate connections would be needed between the remote facility and each of the local machines, whereas with the local network the remote facility need only tap onto the local network in one place.

Another aspect of interest in local networks is to take the advantages of distributed computing. In this approach, some of the machines are dedicated to perform some specific functions, such as file storage, data base management, terminal handling, and so on.

b. Wide Area Network.

Wide area network (WAN) has a diameter from 10 km to 100 km and which lies within a country. Wide area network is also called long haul network. Local networks differ from long haul networks in several ways. The key difference is that the designers of long haul networks are often forced by economic or legal reasons to use the public telephone network, regardless of its technical suitability. Bandwidth consideration is very important in the case of long haul network but in the case of local network, nothing prevents the designers from laying their own high-bandwidth cables .

c. Metropolitan Area Network.

Metropolitan area network (MAN) is in between LAN and WAN. It covers the entire city but uses LAN technology. Cable television (CATV) networks are examples of analog MANs for television distribution. Most of LAN protocols also holds for MANs.

d. If the network is inter continental then it is called inter connection of long haul networks.

1.2 SOME APPLICATIONS OF COMPUTER NETWORK

Instead of using a single main frame, computer network is used for the following reasons:

Resource sharing i,e to make all programs, data and equipment available to anyone on the network without regard to the physical location of the resource and the user. Load sharing is another aspect of resource sharing.

High reliability is obtained by having alternative sources of supply i,e all files could be replicated on two or three machines So if one of them is unavailable (due to hardware failure), the other copies could be used. For military, banking, air-traffic control and many other applications, the ability to continue operating in the face of hardware and software problems is of great importance.

So it is found that replacing a single mainframe by workstations on a LAN (Local area network) improves the reliability and performance[1]. The availability of WAN(wide area network) makes many new applications possible. Some of this new applications may have important effects on society as a whole. Some important uses of computer network are access to remote program, access to remote databases, communication facilities etc.

A company that has produced a model simulating the world economy may allow its client to log on the network and run the program to see how various projected inflation rates, interest rates, and currency fluctuation might affect their business. The use of remote databases is a major area of network usage. It may soon be easy for a person sitting at home to make reservation of airplanes, trains, buses, boats, hotels, restaurants, theaters and so on any where in the world with instant confirmation. Home banking and automated newspaper also falls into this category.

Present newspaper carries a little bit of everything but the automated newspaper can easily be tailored according to the readers personal taste.

A widespread network use is as a communication medium. At present computer to computer communication in different countries can be done by electronic mail by only computer personnel. In future, it will be possible for everyone, not just people in the computer business, to send and receive electronic mail. Furthermore, this mail will also be able to contain digitized voice , still pictures and possibly even television and video images. In the present state of development of science and technology, communication engineering plays the most important role because now a days there is a race between communication technology and transportation. The question is now that people should move physically to fulfill his requirement and getting all kinds of information or the information should be available to him staying at home. All the developed countries are choosing the second one and the evaluation of ISDN (integrated Services Digital network) is for this purpose [2]. In future the telephone system will be digital transmission from end to end and will carry both analog and digital signals of all kinds and this will be done by using broadband integrated services digital network (BISDN).

1.3 INTERFACING OF COMPUTER NETWORK

Local area networks are rapidly emerging as a major and distinct class of computer communication networks and their salient characteristics make them particularly attractive as an efficient and economic solution to high speed unregulated connections within a limited distance geographical area. There is an interfacing between the LAN user and the LAN. The main limitations of LAN performance is due to the interface processing time and queuing delay while its sensitivity to the cable speed and access protocol is rather limited. Broadband integrated services digital network has been designed to carry high volumes of real time information such as voice, video and interactive data etc [3]. For transmission of voice packets from CODEC (which converts analog signals into digital forms), and data packets from terminals there must have some interfacing between the network and the local node generating the voice and data packets. Figure 1.1 shows how local interface interacts with the network and the users. To serve different users there are queues for voice packets, queues for data packets and queues for control packets and the interface unit may be represented by a network of queues.

1.4 NETWORK INTERFACE UNIT

The NIU (Network interface unit), as the name suggests, plays the role of interfacing between the constituent units of the local node and the network. There are three functional units in a broadband packet switch e.g. packet sorting unit (PSU), packet despatch unit (PDU) and network interface unit (NIU). Figure 1.2 shows the block diagram of a packet switch with different units and their interaction with each other. All incoming packets into

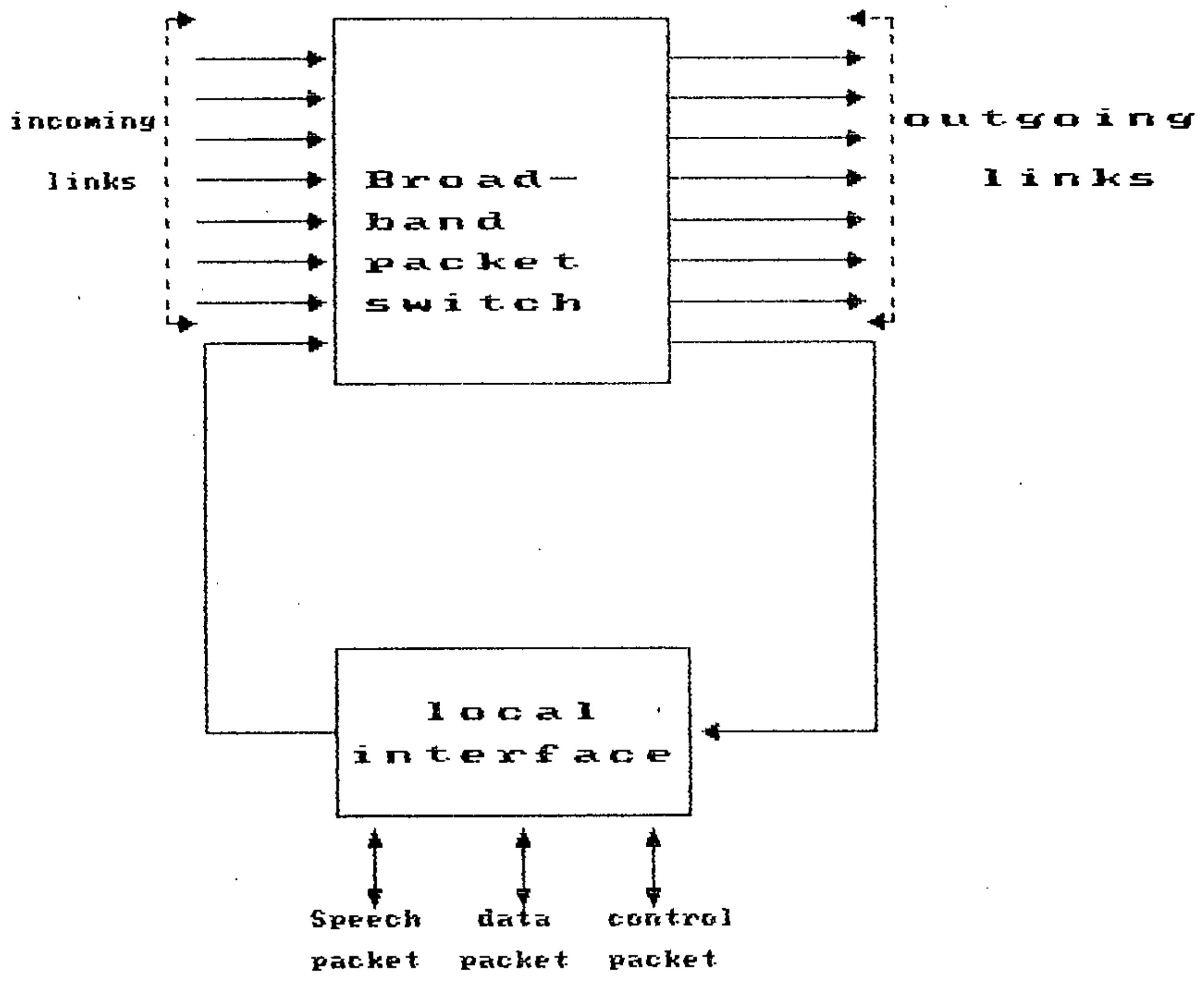


Figure 1.1 : Broadband packet switch configuration with local interface [3].

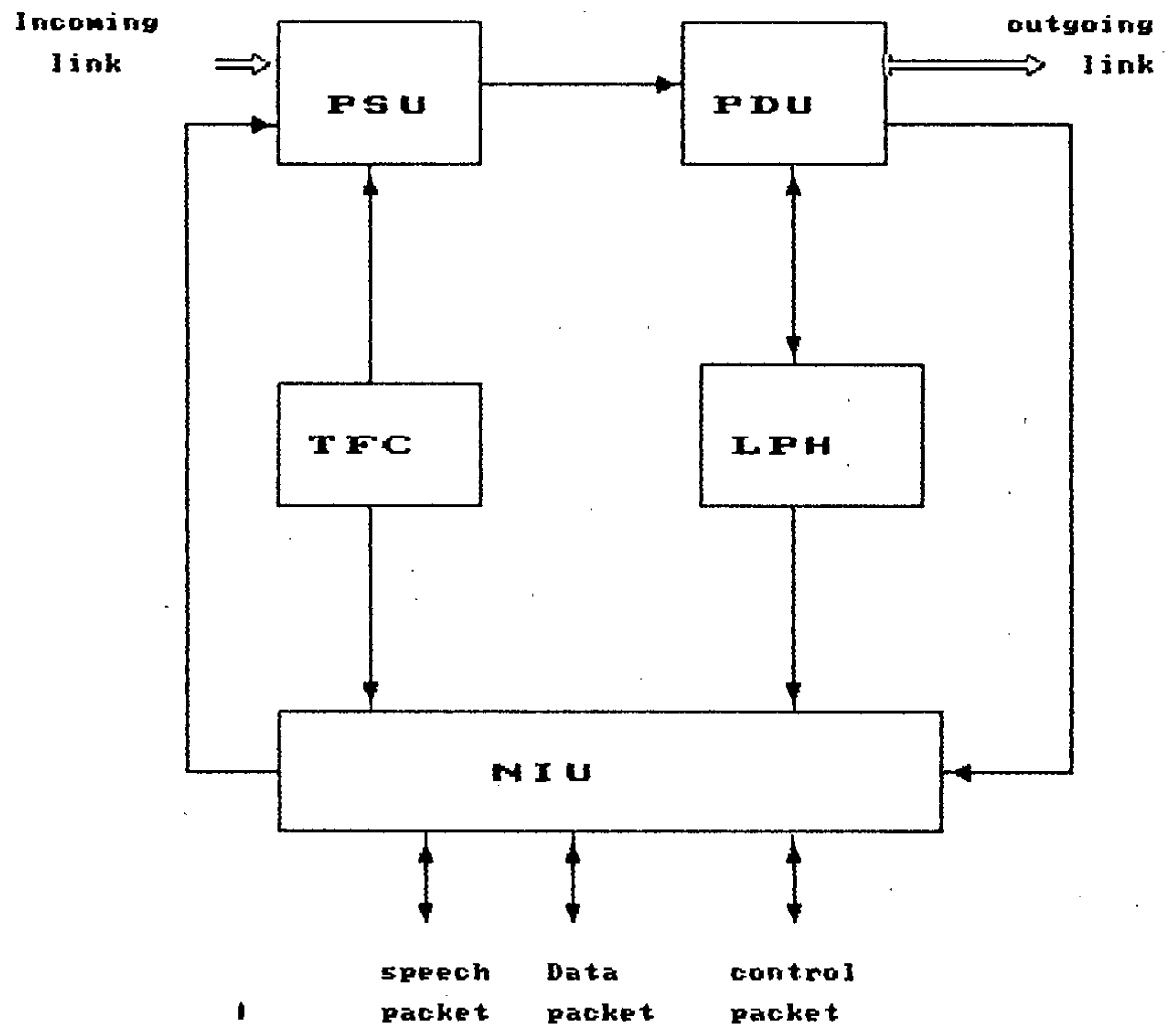


Figure 1.2: Block diagram of broadband packet switch configuration.

the packet switch are sorted in the packet sorting unit and classified according to their address. The packets destined to another node is given to the packet despatch unit for transmission to the desired node. The packets for local exchange or local node is given to the network interface unit. Packet sorting unit is assisted by traffic flow controller (TFC) unit which controls the flow of incoming packets to the packet switch. The voice and data packets are also classified in the packet sorting unit. The packets from the packet sorting unit are transmitted to another node by the packet despatch unit. The packet despatch unit transfers the header of the packet from the header store and the body of the packet from the packet store. The packet despatch unit is assisted by local packet handler (LPH) to have an error free transmission and to maintain the generation sequence of the voice packets. All outgoing packets from the local nodes have to pass through the NIU, which assembles the header and the checksum fields. The packets converging on the NIU comprise the voice packets from the speech buffers, the call control packets from the call control processor, the data packets from the host computer, and the flow control and diagnostic packets from the TFC (traffic flow controller). All incoming packets from CODECs (voice packets), host computer or terminals (data packets), enters into the NIU to have entrance into the network. In a broadband packet switch the NIU interacts with the traffic flow controller to control the flow of packet, local packet handler and packet despatch unit to have an error free transmission. Figure 1.3 shows the

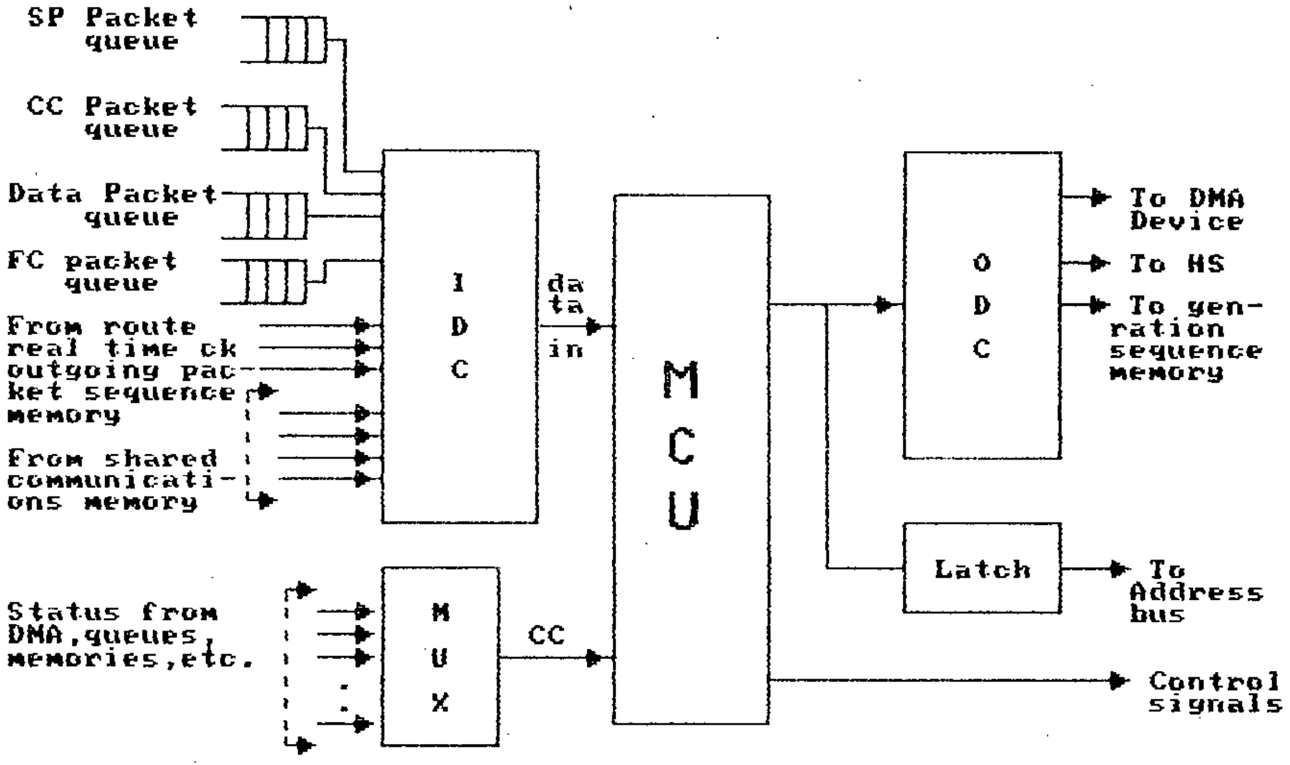


Figure 1.3: Network interface unit architecture [3].

architecture of a network interface unit which consists of the following blocks.

a. The input queues.

There are four input queues e.g. speech packet queue, control packet queue, data packet queue, flow and diagnostic packet queue. Speech packet queue which contains the voice packets from CODECs for digital voice communication. Data packet queue contains the data from the host computers or an interactive terminal. Flow and diagnostic control packet queue contains the packets from the terminal flow controller to control the flow of packet.

b. Input data controller(IDC).

For transmission of both voice packets and data packets using broadband integrated services digital network, there must be a priority controller in the network interface unit to give the highest priority to the voice packets. Because the voice packets must reach the destination according to the generation sequence of the packets to be understandable to the listener. Input data controller controls the priority of the packets and maintains the generation sequence of the voice packets. Real time clock input of the input data controller provides the time stamp for voice packets. The implementation of explicit routing scheme requires the saving of the explicit route address for every established voice connections. The local packet handler saves and updates this information in the route record. The input data controller takes this record to assemble the header of the outgoing packets.

There is a shared communication memory to store the header of the incoming packet and the body of the packet and the input data controller is informed the arrival of the packet.

c. Microprogrammed control unit (MCU). Packets from the input data controller enters into the microprogrammed control unit where the main functions of layer 2 protocol and some layer 3 protocol are processed. Status from the DMA, different queues, and memories passes to the MCU through the multiplexer. MCU performs the buffer management and supplies the address of a free buffer for loading a new packet. Headers and checksum of a packet are added in the MCU. Control of voice traffic involves two tasks: reducing the number of calls set up through the congested node by the required percentage and preventing the setting up of new calls through the congested node. TFC sets a bit in the route record and the MCU interprets this bit and seeks a new route for further voice packets generated from this connection.

d. output data controller (ODC).

The output data controller controls the DMA transfer of the packet to the common buffer pool which is accessible to the packet switch. The header of the packet is stored in the header store and the packet is stored in the packet store. Output data controller also maintains the generation sequence number of the voice packet.

Delay is very important for transmission of voice packets. In a packet switched network, there can exist multiple routes between two nodes and if the voice packets are allowed to take different routes to the destination, they may reach the destination out of sequence due to different delays and the reconstruction of speech may become cumbersome. To overcome this problem, speech packets are allowed to follow a fixed route and there is a provision for alternate path in the event of congestion of the previous path. To achieve this an explicit route number (ERN) is used in the header of a voice packet. To minimize the time delay for voice packets priority controller is used to give highest priority to the voice packets.

1.5 SCOPE OF THE THESIS

The purpose of the thesis is to have an analytical solution of node interfaces of computer networks using different queuing models. In the study of the queuing models, no priority controller has been included in the model and all the packets have been considered of the same priority. To transmit voice packet and data packet through same node priority must be given to the voice packet to maintain the generation sequence of the voice packets. As no priority controller is included in the model so the transmission of voice packets is not included in the study.

The node interface is seen as a network of queues and the performance has been studied by analyzing the delay and throughput of the queuing network. In chapter 2 the theory of the queuing model has been described and shown how a queuing model is to be solved by using buzen algorithm and mean value analysis. Different types of single chain queuing network has been studied in chapter 3. Firstly a simple closed queuing network has been studied by using mean value analysis. Then a more complex queuing network having multiple controllers has been studied. Often it is the case that multiple protocol processors is used in parallel to increase the throughput and to decrease the delay of the packet transmission. So queuing network with multiple protocol processors also has been studied in the last part of the chapter 3. When different types of stations communicate through a packet switch to transmit packets to destination stations then multi-chain queuing network arises and the solution of multi-chain queuing network has been given in chapter 4. Thus throughout the thesis, node interface of computer network has been analyzed by using different queuing models.

CHAPTER 2

MODELING QUEUING NETWORKS: BUZEN ALGORITHM AND MEAN VALUE ANALYSIS

2.1 INTRODUCTION

Queuing models for computer systems are constructed by considering each relevant resource as a server, which receives requests for service from the programs processed by the systems. When a request finds the server busy, it joins a queue where it waits its turn to be served. Depending on the models degree of details, a request may represent an entire program, a program step an interactive command, an i/o instruction and so on.

A service center or station may contain more than one server and is therefore capable of serving more than one request simultaneously.

Furthermore a station may have more than one queue. Figure 2.1 shows the component of a station together with some of the variables that are used to describe the phenomena occurring when the station is working.

Queuing systems are usually studied in steady state conditions i.e. the assumption is made that the system has been working for a long period of time so that the distribution descriptors of the random variables we use to represent it are no longer influenced by the systems initial conditions and are therefore independent of time. A pre-requisite assumption that is always at least implicitly made is that the system is stationary.

A source is characterized by its request generation process. A station is characterized by its number of server and of queues, by the maximum admissible length of the queues (queue capacity), by the speed of its server and by service discipline.

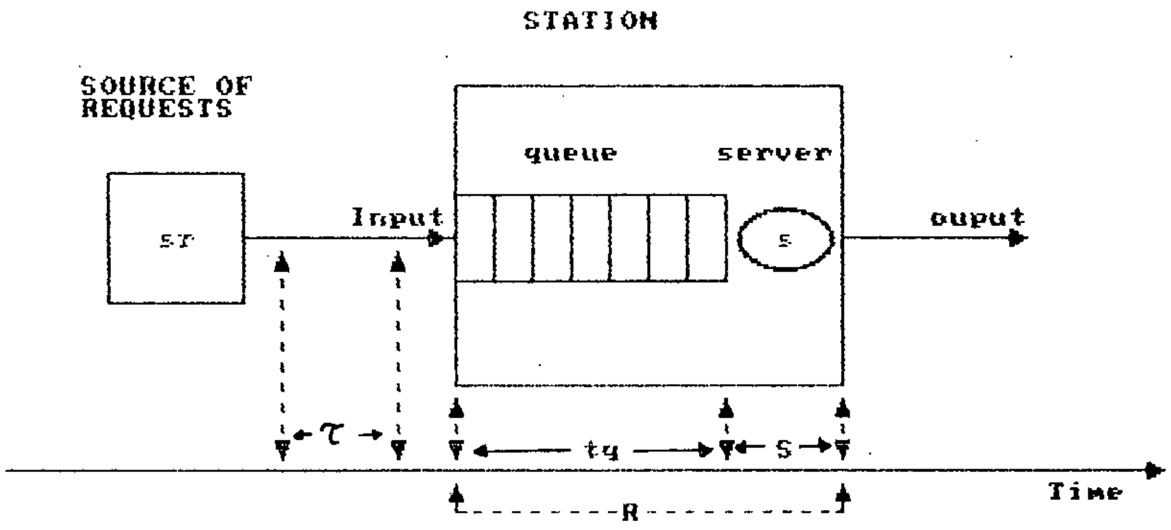


Figure 2.1: Components of a single server station and some of the variables used to describe the behavior : τ (the inter-arrival time), tq (mean queuing time), S (mean service time), R (mean response time of the station)[4].

2.2 ANALYSIS OF THE BEHAVIOR OF A STATION

To analyze the behavior of a station, detailed description of its arrival process, service process and service discipline must be provided. Arrival and departure phenomena are represented by random variables. A random variable is characterized by its probability distribution function. This function for random variable x and for each real x_0, is given by

$$F_x(x_0) = P[x <= x_0]$$

i,e by the probability that the value of x is not greater than x_0. The derivative of this function with respect to x is the probability density function $f_x(x_0)$.

The arrival process is usually assumed to be a Poisson process, an assumption that is supported by the result of several experimental studies of interactive systems. In a Poisson arrival process the probability of having k arrivals within a given time interval of length t is,

$$P_k\ (t) = \frac{(\lambda * t)^k}{k!} * e^{-(\lambda * t)} \qquad [\ k >= 0,\ t >= 0]$$

Where λ is the mean arrival rate.

Average no of arrivals occurring during each time interval t is equal to $\lambda * t$.

The probability that no arrivals will take place in time t

$$P_0\ (t) = e^{-(\lambda * t)}$$

So the probability of at least one arrival is

$$1 - P_0(t) = 1 - e^{-(\lambda * t)}$$

thus the probability that a new arrival will occur τ time units after the most recent arrival with τ <= t is

$$F_\tau(t) = P[\ \tau <= t\ ]$$
$$= 1 - e^{-(\lambda * t)} \qquad (\ t >= 0)$$

which is negative exponential distribution with parameter λ .

A crucial property of the exponential distribution is the markov or memoryless property whose importance in simplifying the solutions of the probabilistic models is fundamental. In an arriving process having this property the probability that a new request will arrive during the next t units of time is independent of the amount of time already elapsed since the last arrival. Service process has characterization similar to those that are used to model the arrival process. Service time are always assumed to be statistically independent of inter-arrival time.

The probability that a generic request will not need more than t time units of service is given by

$$F_s(t) = P[\ s <= t\ ]$$

$$= 1 - e^{-(\mu * t)}$$

where μ = mean service rate.

The fundamental performance index for queuing model is server utilization which may be defined as the steady state probability that the server is busy. For single server model, utilization may be defined as the steady state probability that the server is busy and which equals the ratio λ / μ [4]. Another performance index is the mean response time which is defined as the total service time plus the total waiting time of a request.

If N = mean no of requests in the system

R = mean response time

then according to Little's formula

$$N = \lambda * R$$

where λ is mean arrival rate of the system.

Another form of Little's formula is as follows:

$$N_q = \lambda * t_q$$

where N_q = mean number of queued requests.

t_q = mean waiting time of a request.

Little's formula holds for any service discipline, any inter-arrival time and service time distributions and any number of queues in the station.

For single server station with Poisson arrival and exponential service times, classical result of queuing theory states that at the equilibrium we have

$$N = \frac{U}{1 - U}$$

using Little's formula, we have

$$N = \lambda * R$$

so $$\lambda * R = \frac{U}{1 - U}$$

$$R = \frac{U}{\lambda * (1 - U)}$$

but we have $$U = \frac{\lambda}{\mu}$$

so $$R = \frac{1}{\mu * (1 - U)}$$

Considering first come first served discipline mean waiting time

$$t_q = R - \frac{1}{\mu}$$

$$= \frac{1}{\mu * (1 - U)} - \frac{1}{\mu}$$

$$= \frac{U}{\mu * (1 - U)}$$

2.3 NETWORK OF QUEUES

Queuing network are much better suited to the construction of intermediate level and micro level models of computer system. An open network has at least one source of request external to the network, and as well as at least one exit that requests can use to leave the network.

Network having no external sources as shown in figure 2.2 is called closed queuing network. In many practical problems, the assumptions of an unlimited number of requests in the system is certainly unrealistic. So to model more accurately multiprogramming systems, where the maximum no of active programs are limited by the size of the main memory or in interactive systems where the maximum no of commands in execution is limited by the number of active terminals, one can make use of closed queuing network in which the number n of requests (or programs or commands) in the model is kept constant.

The state of a network consisting of m stations is defined if the state of each station is known; thus the state is represented by the vector

$$N = \{ N1, N2, N3 \ldots\ldots N_m \}$$

where N_i is the number of requests in i-th station. A network of m stations is said to have a product form solution if the equilibrium distribution of network state probabilities exists and can be written as

$$P(N) = \frac{P_1(n_1) . P_2(n_2) \ldots . P_m(n_m)}{G}$$

where $P_i(N_i)$ is the probability that the i-th station is in state N_i. G is the Normalizing constant that forces the sum of all state probabilities to be equal to 1.

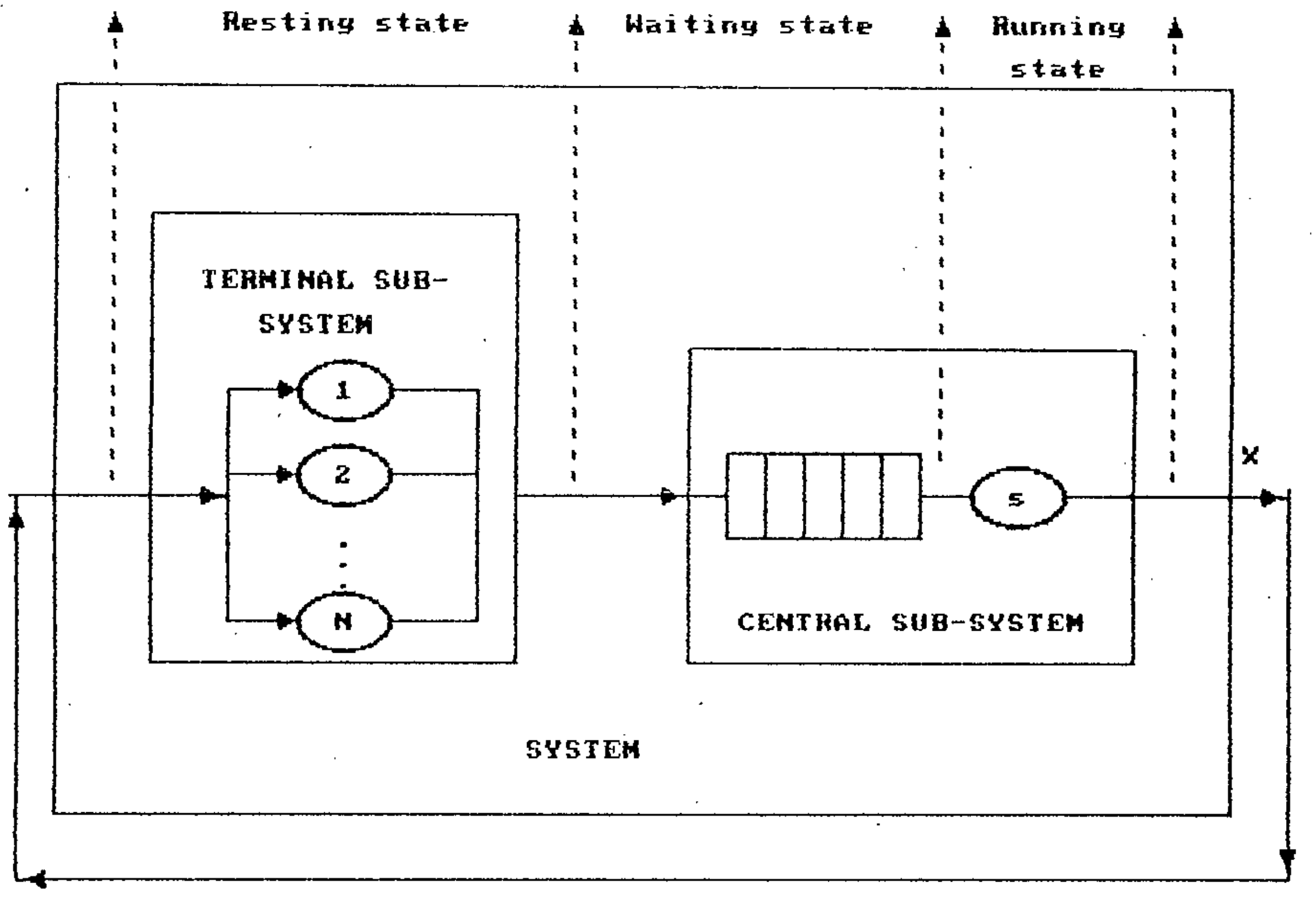


Figure 2.2: Closed network macrolevel model of an interactive system[4].

2.4 BUZEN'S ALGORITHM

The central sub system shown in figure 2.2 with a greater level of detail may be replaced by a network of stations representing various resources and their interconnections. A closed network model often used to represent multiprogramming systems at the intermediate level is the central server model shown in figure 2.3. This model has been studied by Buzen using the approach to the solution of exponential closed networks proposed by Gordon and Newell.

From the central server model if k is the number of server and N is the number of programs, balancing the input and output rates of station i yields

$$\mu_i * U_i = q_{1i} * \mu_1 * U_1 \qquad (i = 2, \ldots k) \qquad (2.1)$$

where q_{1i} is the rate of transition from server 1 to station i.

If we define

$$y_1 = 1 \quad \text{and} \quad y_i = \frac{1}{\mu_i} * q_{1i} \qquad (i = 2, \ldots k) \qquad (2.2)$$

then from equation 2.1 we get

$$U_i = y_i * U_1 \qquad (i = 1, \ldots k) \qquad (2.3)$$

From (2.1) we have k-1 homogeneous equations with k unknowns and their solutions can be determined except for a constant which can be assumed to be equal to U_1. To derive the value of the constant the balance equation for all the states of the model must be

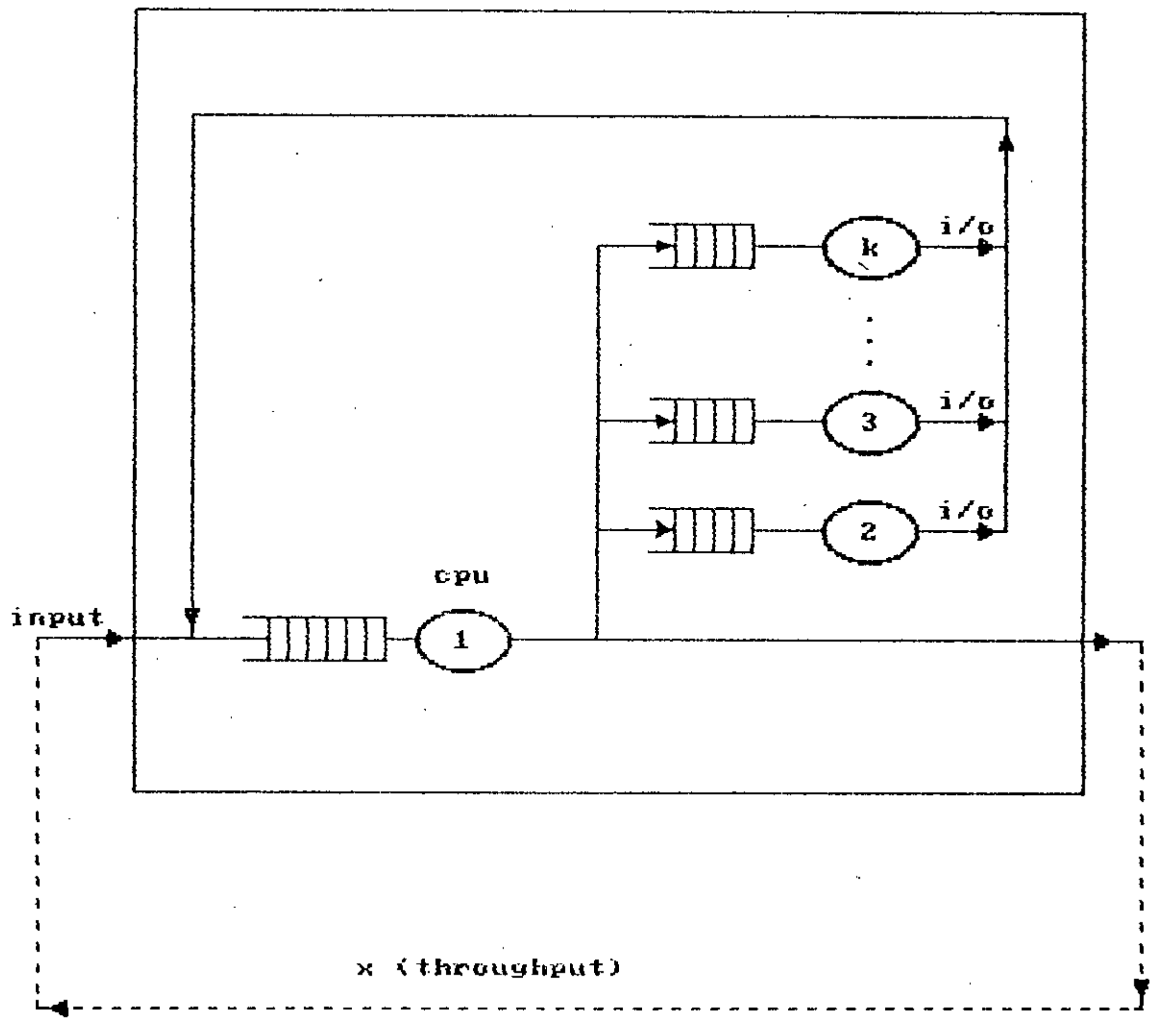


Figure 2.3 Closed model of the central server type[4].

written. These yield a product form solution of the type

$$P(N) = \frac{1}{G(N)} * y_1^{n1} * y_2^{n2} * \ldots\ldots * y_k^{nk} \quad \ldots \quad (2.4)$$

where G(N) is given by

$$G(N) = \sum_{\text{all } N} y_1^{n1} * y_2^{n2} \ldots\ldots y_k^{nk}. \quad (2.5)$$

Then we can write

$$U_1 = \frac{G(N-1)}{G(N)} \quad (2.6)$$

It is the value of the unknown constant.

Now to find utilization U_1, G(N) have to be computed. Now the utilizations of all other stations may be obtained from equation 2.3,

$$\text{i,e} \quad U_i = y_i * U_1 = \frac{\mu_1}{\mu_i} * q_{1i} * U_1 \quad (2.7)$$

The models mean throughput rate x at the equilibrium is defined as the mean number of programs that go through the loop around the central server. This number may be obtained by multiplying the cpu's output rate times the probability of departure q_{10}.

$$\text{i,e} \quad x = \mu_1 * U_1 * q_{10} \quad (2.8)$$

The mean response time R is

$$R = \frac{N}{X}. \quad (2.9)$$

To find the values of the performance indices, it is important to compute the values of function G . A computationally efficient algorithm has been developed by Buzen .This algorithm requires kN additions and kN multiplications and requires the auxiliary function $Y_i(n)$ to be recursively defined as follows :

$$Y_i(0) = 1$$

$$Y_1(n) = y_1^{\,n} = 1 \qquad (2.10)$$

$$Y_i(n) = Y_{i-1}(n) + y_i * Y_i(n-1) \quad \ldots \quad \ldots \quad [\, n = 1,\ldots N; \; i = 2,\ldots K]$$

It is seen that

$$Y_k(n) = G(n)$$

2.5 MEAN VALUE ANALYSIS

A new approach to the solution of product form queuing network called mean value analysis has been introduced by Reiser and Levenberg [5]. Instead of calculating G and deriving from it the values of the performance indices this approach computes the indices by the recursive relationships that exists among them. The validity of Mean Value Analysis extends to all cases in which the normalizing constant approach can be applied. From normalizing constant approach we found that the mean queue length at server i is given by

$$N_i(N) = \sum_{n=1}^{N} n * p_i \; (N_i = n)$$

$$N_i(N) = \sum_{n=1}^{N} y_i^{\,n} * \frac{G(N-n)}{G(N)} \qquad \ldots \qquad (2.4)$$

Where N is the number of programs in the network. By applying some simple algebra this expression can be transformed into the recursive form

$$N_i(N) = U_i(N)[1 + N_i(N-1)]. \qquad \ldots . \qquad (2.5)$$

With $N_i(0) = 0$

By Littles formula, the mean response time can be derived

$$R_i(N) = \frac{N_i(N)}{X_i(N)}$$

Putting the value of $N_i(N)$

$$R_i(N) = \frac{U_i(N)[1 + N_i(N-1)]}{X_i(N)}$$

Again

$$U_i(N) = X_i(N) * S_i \qquad \ldots \qquad (2.6)$$

Where S_i = Mean service time of station i

Putting the value of $U_i(N)$ we get

$$R_i(N) = \frac{X_i(N) * S_i * [1 + N_i(N-1)]}{X_i(N)}$$

so $R_i(N) = S_i * [1 + N_i(N-1)] \qquad \ldots \qquad (2.7)$

From the above equation we see that the mean time taken by station i to process a request equals the sum of the mean time required to process the requests already queued up and the requests mean service time.

An expectation of the mean throughput rate in which the normalizing constant G is not used can be obtained by applying Littles formula to the whole network. Since the systems mean response time R is the sum of the mean times spent by a program in each of the k stations, i,e

$$R(N) = \sum_{i=1}^{K} V_i * R_i(N). \quad \ldots \quad (2.8)$$

So the systems mean throughput rate

$$X(N) = \frac{N}{R(N)}$$

Applying Littles formula to station i

$$N_i(N) = X_i(N) * R_i(N)$$

So starting from the initial condition $N_i(0) = 0$, it is possible to calculate $R_i(N)$,$R(N)$,$X(N)$,$U_i(N)$ and $N_i(N)$ iteratively.

CHAPTER 3

SOLUTION OF SINGLE-CHAIN CLOSED QUEUING NETWORKS FOR UNIPROCESSOR AND MULTIPROCESSOR INTERFACES APPLYING BUZEN ALGORITHM AND MEAN VALUE ANALYSIS

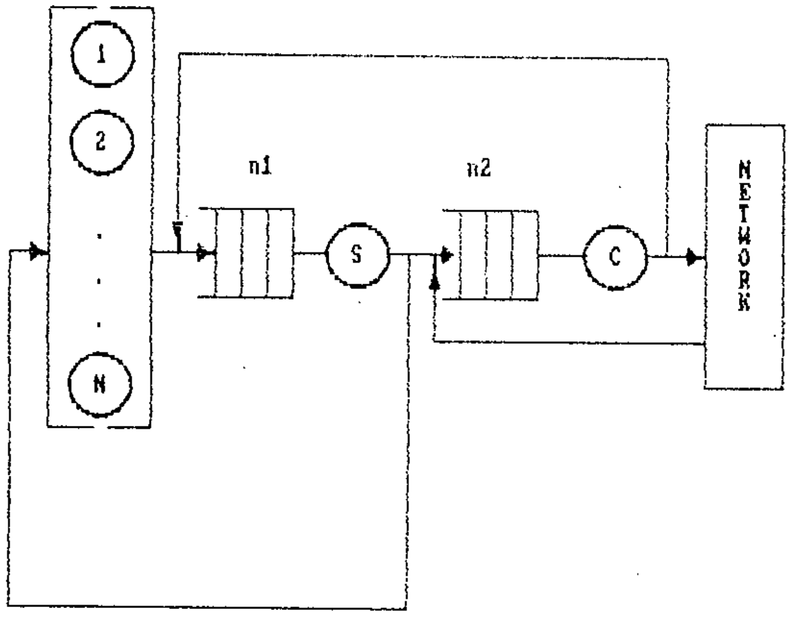
3.1 INTRODUCTION

In this chapter starting with a simple queuing network, more complex queuing network has also been solved by using Mean Value Analysis and Buzen algorithm. Firstly a simple queuing network has been solved by using Mean Value Analysis and shown how the utilization of the protocol processor, delay of the network of queues and the throughput varies with the number of terminals. More complex queuing network with multiple controller has also been solved by using Buzen algorithm with flow equivalent aggregation method. The performance indices used for this purpose is the delay and throughput of the network and the utilization of the protocol processor. Here it is shown that how the performance indices are affected with the number of terminals and the number of controllers.

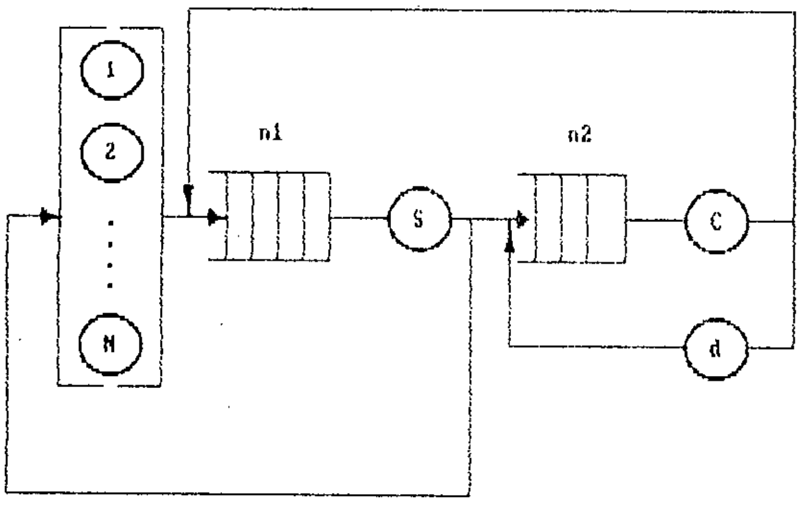
Multiple protocol processors are often used in the interface of computer network to increase the speed of operation i,e to decrease the delay of the queuing network. Towards the end of this chapter a queuing network with multiple number of protocol processors has been studied and shown how the delay of the network decreases as the number of protocol processors is increased.

3.2 DESCRIPTION AND SOLUTION OF THE SIMPLE QUEUING NETWORK

In figure 3.1 (a) simple queuing network has been shown. Here s represents the protocol processor, n1 represents the protocol



(a)



(b)

Figure 3.1: (a) The output of the controller is directed to network.
(b) Network is replaced by a delay.

processor queue. The controller is represented by c and n2 represents the controller queue. There are total N number of terminals. A message is generated by a terminal and entered into the protocol processor. If the protocol processor is busy the message waits in the processor queue. As the waiting time of the message in the processor queue is elapsed the massage enters into the processor. Protocol processor breaks the data into frames, maintains the sequence of the frames and processes the acknowledgment frames. The controller handles buffer linking and management, interrupt generation, several types of frame checking and error detection [6,7,8]. The controller operates as a receiver in one of the four modes:

* Single mode: A specific physical address programmed into the controller's initialization block must match exactly the address field of the packet.

* Multicast: When the multicast bit in the address field of the transmitted message is set, it will be recognized by all the controller devices in the network.

* Broadcast: A hash filter maps the physical address into one of the logical address groups. The controller then recognizes all packets that are addressed to a single logical group.
* Promiscuous: Accepts any data packet regardless of the contents of the transmitted message's address field.

To carry out DMA, the controller must operate as a bus master. Setting a control bit in one of the internal controller registers, the controller selects the DMA mode.

In figure 3.1 (a) it has been shown that the message from the controller goes to the network and there is a delay called bus delay which is represented by d. This delay consists of the transmission delay across the bus, transceiver delay and queuing delay at destination node. After reception of the message the destination station generates an acknowledgment which returns to the controller again. In the analysis the delay in the closed queuing network represents the summation of the delay in the protocol processor, the delay in the controller and the bus delay in the network.

The mean value analysis described in chapter 2 has been used to solve the queuing network in figure 3.1 (b). So the solution procedure is not given in this chapter.

3.3 DESCRIPTION OF THE MULTI-CONTROLLER QUEUING NETWORK.

Figure 3.2 shows the queuing network with multiple number of controllers. The number of terminals varies from 1..N and number of controllers varies from c1...cc and s represents the protocol processor .The functions of the protocol processor and the controllers have been described in section 3.2. The bus delay is represented by d. To find the delay of the queuing network the message passes three stages. One in the protocol processor, one in the controllers and the last one is the bus delay. So the

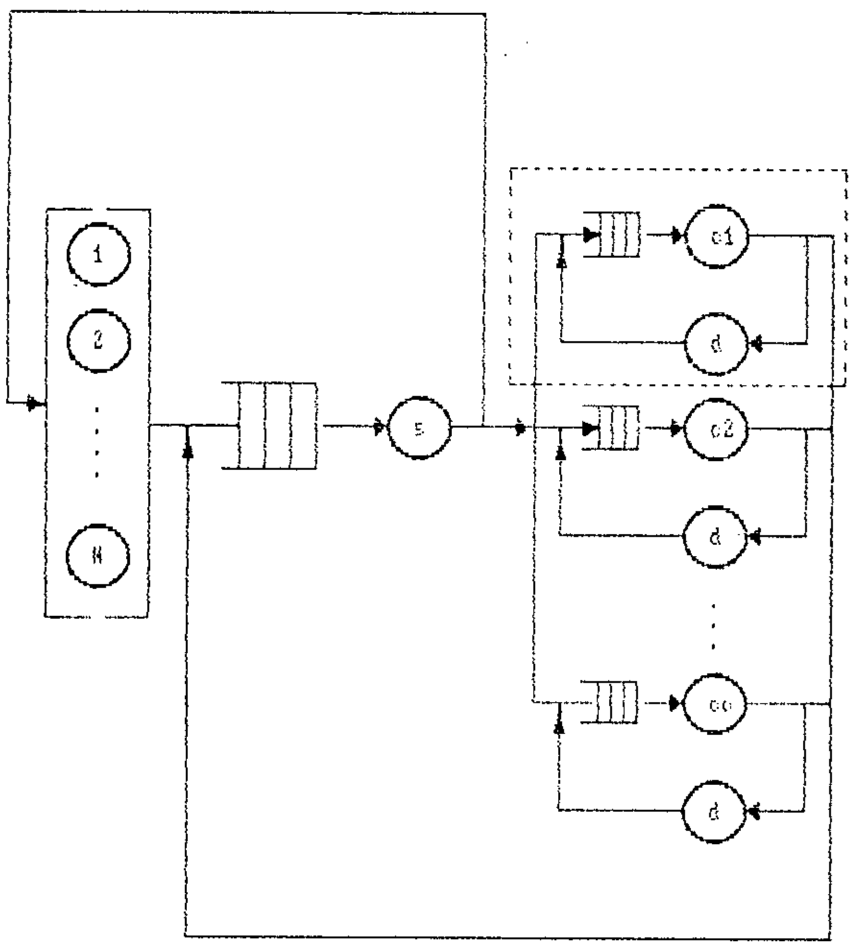
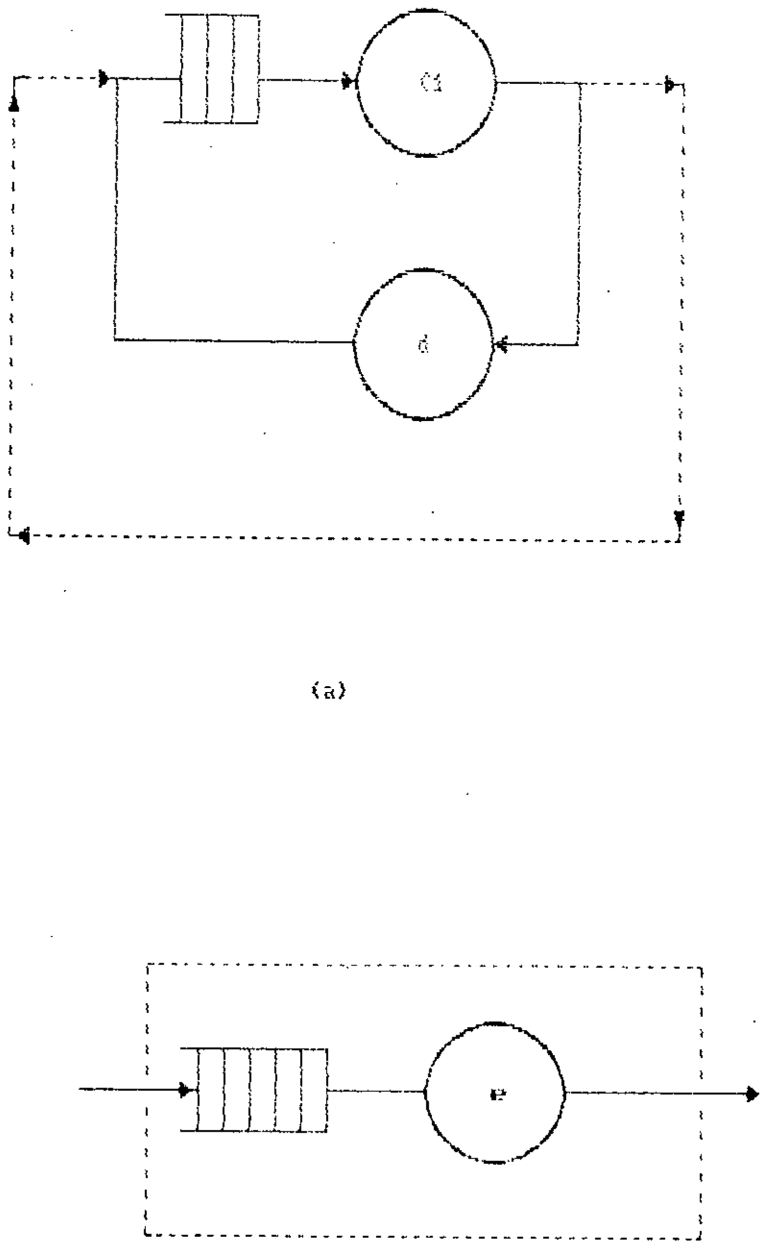


Figure3.2 :Multi-controller queuing network.

delay of the queuing network cannot be found directly applying the Buzen algorithm because Buzen algorithm cannot be applied to the queuing network with more than two stages. To make the queuing network solvable using Buzen algorithm simplification of queuing network using flow equivalent aggregation has been used. By using flow equivalent aggregation the queuing network has been simplified so that Buzen algorithm may be applied. The subnetwork shown by the dotted lines in figure 3.2 has been separately analyzed that is shown in figure 3.3 (a).To find the equivalent of the subnetwork it has been considered as a closed queuing network. The subnetwork has been studied in isolation and replaced by simpler equivalent component as shown in figure 3.3 (b).These components will produce exactly or approximately the same effect as the subnetwork .So the equivalent component may be replaced in exchange of the subnetwork. The equivalent component is called the flow equivalent station whose mean throughput rate is equal to the throughput of the subnetwork in isolation. After replacing all of the subnetworks in figure 3.2 with their flow equivalent stations we get the queuing network of figure 3.4 which can be solved by using Buzen algorithm.

3.4 SOLUTION OF THE MULTI-CONTROLLER QUEUING NETWORK

To find the solution of the subnetwork in figure 3.2 it has been studied separately in figure 3.3. The output of the subnetwork is connected with the input and then calculated the mean throughput



(b)

Figure 3.3: (a) subnetwork of a controller.
(b) Flow equivalent of the subnetwork.

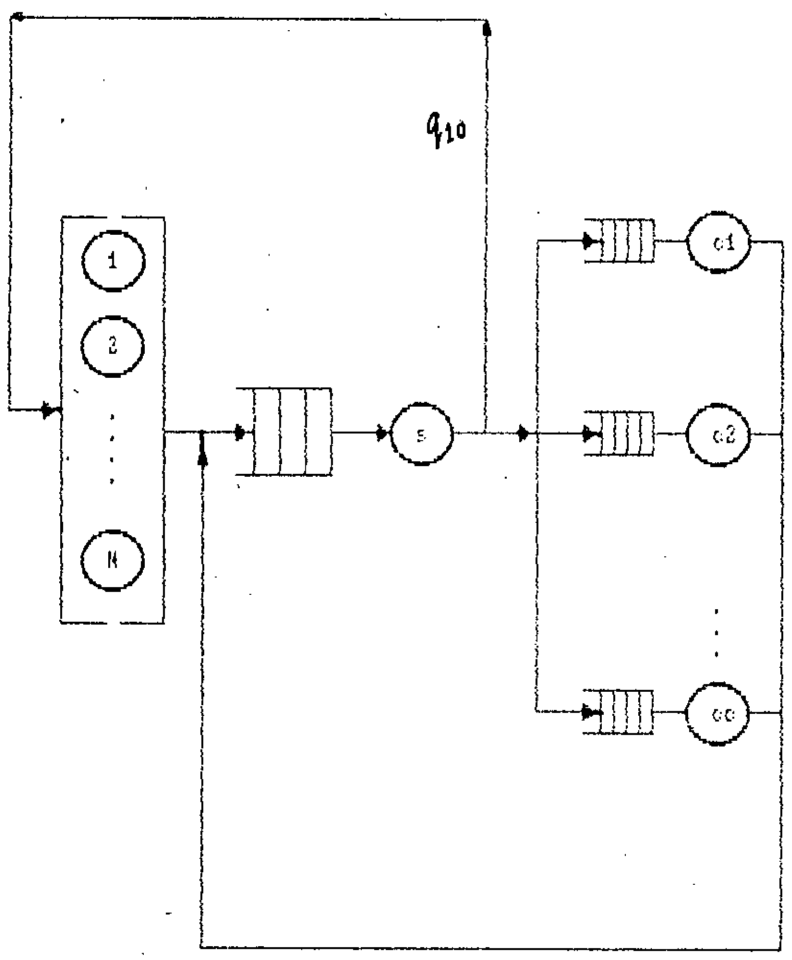


Figure 3.4: Multi-controller flow equivalent queuing network.

rate of the subnetwork for all the values of N to be considered. After replacing the subnetwork with its flow equivalent counterpart it becomes load dependent and the load dependent mean service rate of the single flow equivalent station of figure 3.3 (b) is set equal to the mean throughput rate of the subnetwork. To find the mean throughput rate of the sub-network for all values of N, Mean Value Analysis has been used. Since the equivalent station becomes a load dependent, for the complete solution of the network, Buzen algorithm with load dependent behavior has been used. In stations with load dependent behavior, the mean service time s_i of the station cannot be expressed by a single value, but is to be assigned as a function of the number of requests in the station. The service time of a request, with N_i requests in station i is often assumed to be an exponentially distributed random variable with mean

$$S_i(N_i) = \frac{1}{a_i(N_i) * \mu_i}$$

where $a_i(N_i)$ is a function that can take any positive value and is called **capacity function.**

In our analysis we shall consider the queuing networks with exponential service times, FCFS scheduling discipline, and a single class of jobs to be processed. The equilibrium state probability distribution of a network with k stations is of the form

$$p(N) = \frac{1}{G(N)} \frac{y_1^{N1}}{A_1(N_i)} \frac{y_2^{N2}}{A_2(N_2)} \cdots\cdots \frac{y_k^{Nk}}{A_k(N_k)}$$

where y_i's are the quantities which are functions of the network's topology and parameters and the $A_i(N_i)$'s are auxiliary functions that may be obtained from the $a_i(N_i)$'s as follows :

$$A_i(N_i) = a_i(1) * a_i(2) \cdots\cdots a_i(N_i).$$

with $A_i(0) = 1$. The normalizing constant G(N) is given by

$$G(N) = \sum_{\text{all } N} \frac{y_1^{N1}}{A_1(N_1)} \frac{y_2^{N2}}{A_2(N_2)} \cdots\cdots \frac{y_k^{Nk}}{A_k(N_k)}$$

Though the network includes load dependent stations, the main performance indices can be derived from the knowledge of G(N) only. The computation of G(N) requires a large no of operations. Buzen has proposed an effective algorithm to calculate the normalizing constant G(N) . In this algorithm the auxiliary function $Y_i(n)$ can be recursively computed as follows:

$$Y_i(0) = 1$$

$$Y_1(n) = \frac{y_1^n}{A_1(n)}$$

$$Y_i(n) = \sum_{j=0}^{n} \frac{y_i^j}{A_i(j)} * Y_{i-1}(n-j) \qquad [n=1,\ldots.N; \; i = 2,\ldots k]$$

From the above we have

$$Y_k(n) = G(n) \qquad [n = 1,....N]$$

So the utilization of the protocol processor is

$$U_1(N) = \frac{G(N-1)}{G(N)}$$

Finding the utilization of the protocol processor we get the throughput as follows:

$$X_1(N) = \mu_1 * U_1 * q_{10}$$

where μ_1 = service rate of protocol processor.

q_{10} = the probability of departure of a job from the processor.

3.5 SOLUTION OF MULTI-PROCESSOR QUEUING NETWORK

Figure 3.5 shows a queuing network with multiple processors. The solution of multi-processor queuing network is similar to that of the queuing network of resources with load dependent behavior that has already been solved in the previous section but the difference is to find the value of the auxiliary function $Y_1(n)$ where $n = 1,...N$ [9]. The calculation of the auxiliary function is as follows:

$$Y_1(0) = 1$$

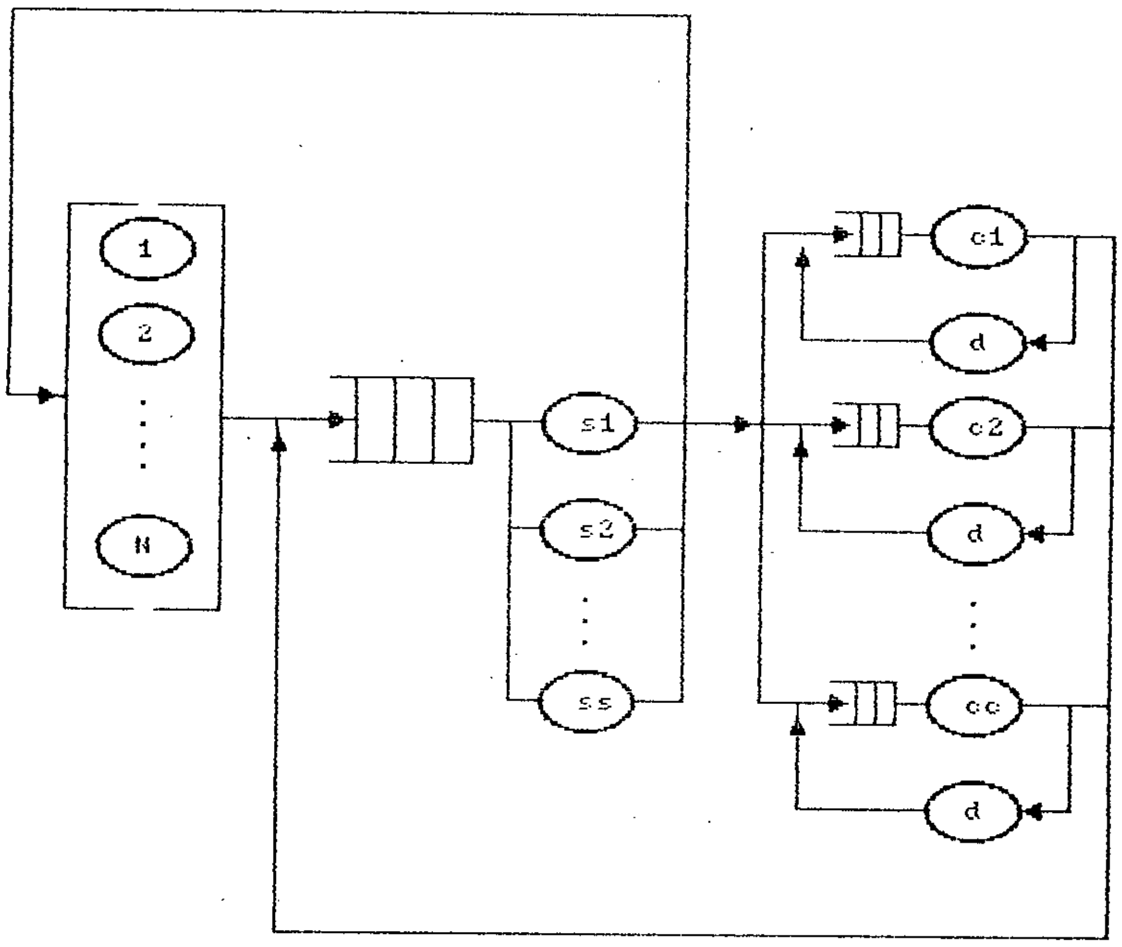


figure 3.5 : Multi-processor queuing network.

$$Y_1(n) = 1/\beta_1(n) \qquad [n = 1,....N]$$

where $\beta_1(n) = n!$ $\quad [\, n < ns \,]$

and $\beta_1(n) = ns! * ns^{(n-ns)}$

where ns = number of processors.

3.6 RESULT AND DISCUSSION

Simple queuing network shown in figure 3.1 has been solved by using mean value analysis. The model has been studied for 10 terminals and found the utilization of the protocol processor, delay and throughput of the queuing network. The bus delay has beem considered 5 mili-seconds which may be the delay of IEEE standard LAN. It is observed from figure 3.6, 3.7 and 3.8 that processor utilization, delay and throughput increases as the number of terminals increases.

Multiple controllers often used to increase the throughput and to decrease the delay of the network. In the solution of multi-controller queuing network, Mean Value Analysis has been used to solve the sub-network and an equivalent network is found which is solved by using Buzen algorithm. Figure 3.9 shows how the throughput of the protocol processor varies with the number of terminals. As the number of terminals increases, throughput increases. But for a fixed number of terminal throughput increases as the number of controller increases. It is also observed that when number of controller is increased from two to three, throughput changes a large amount but if it is changed from three to four the throughput increases a very small amount. This is because that the processor can serve three controllers efficiently and if the controller is increased more, the processor becomes saturation. So in the study of the model, it is found that for single processor it is economical to use three controllers than to use four controllers. If the processor utilization and network delay are considered as shown in figure 3.10 and 3.11 the same result is found.

In the case of multi-processor queuing network number of controllers used for the analysis is five. Number of processors has been varied from two to four. From figure 3.12 it is seen that the processor utilization increases as the number of terminals increases but for a fixed number of terminals the processor utilization decreases as the number of processors increases. This is because in all the cases the processors are running in unsaturated conditions and as number of processors is increased in parallel definitely the utilization of the processor will be decreased. But in the case of throughput as shown in figure 3.13, it increases as the number of processors is increased. Network delay as shown in figure 3.14, also decreases as the number of processors is increased keeping the number of terminals fixed. But it is interesting to note that the change in delay and throughput for changing the number of processors from two to three is wide but it is narrow for changing from three to four. This is because the number of controller is fixed for both the cases and it is five and three processor can serve five controllers efficiently and if the number of processors is increased more, the controllers become saturated and throughput does not increase accordingly. So for this configuration and for given parameters three processors can serve five controllers efficiently and economically.

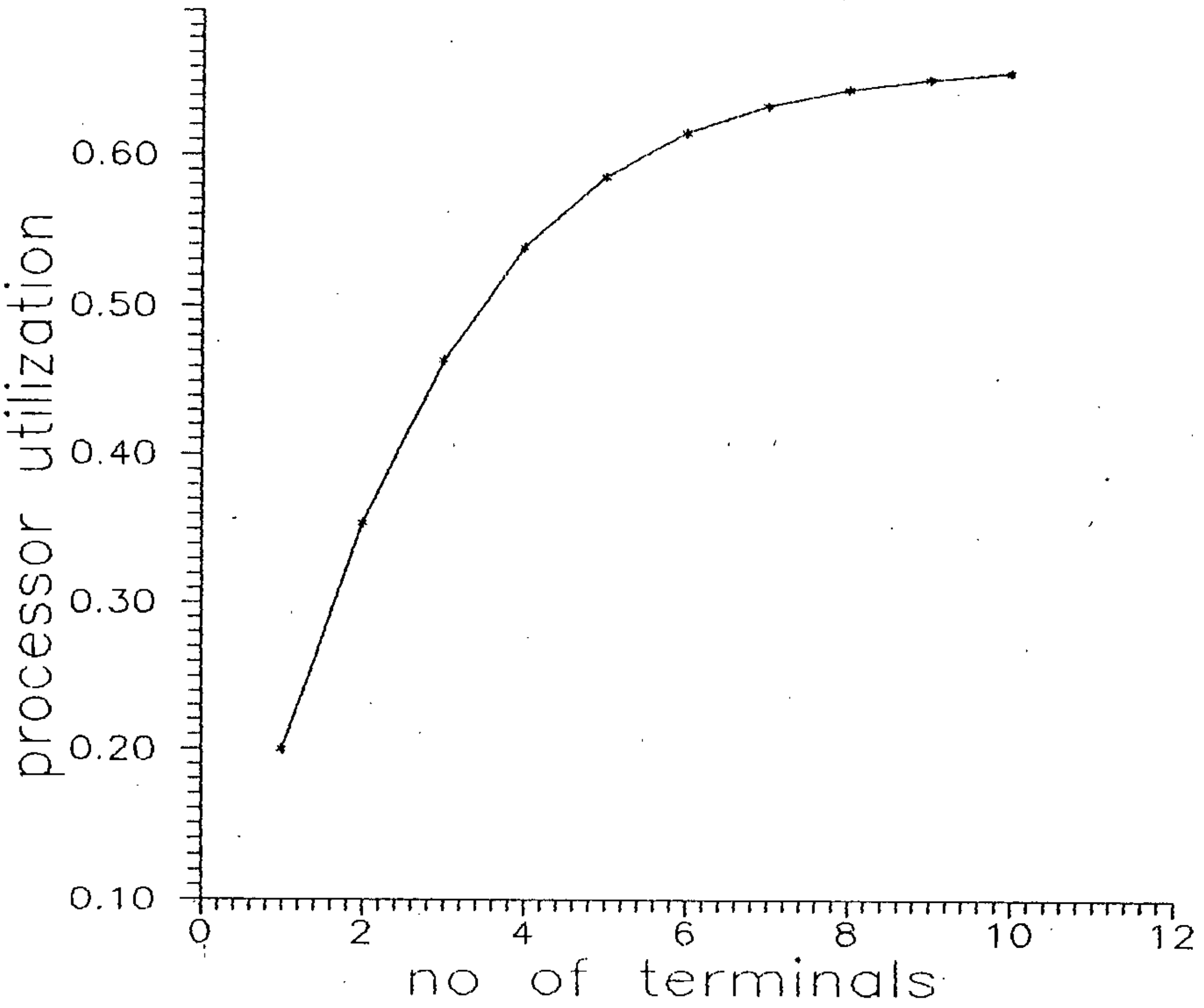


Figure 3.6: Protocol processor utilization vs number of terminals.

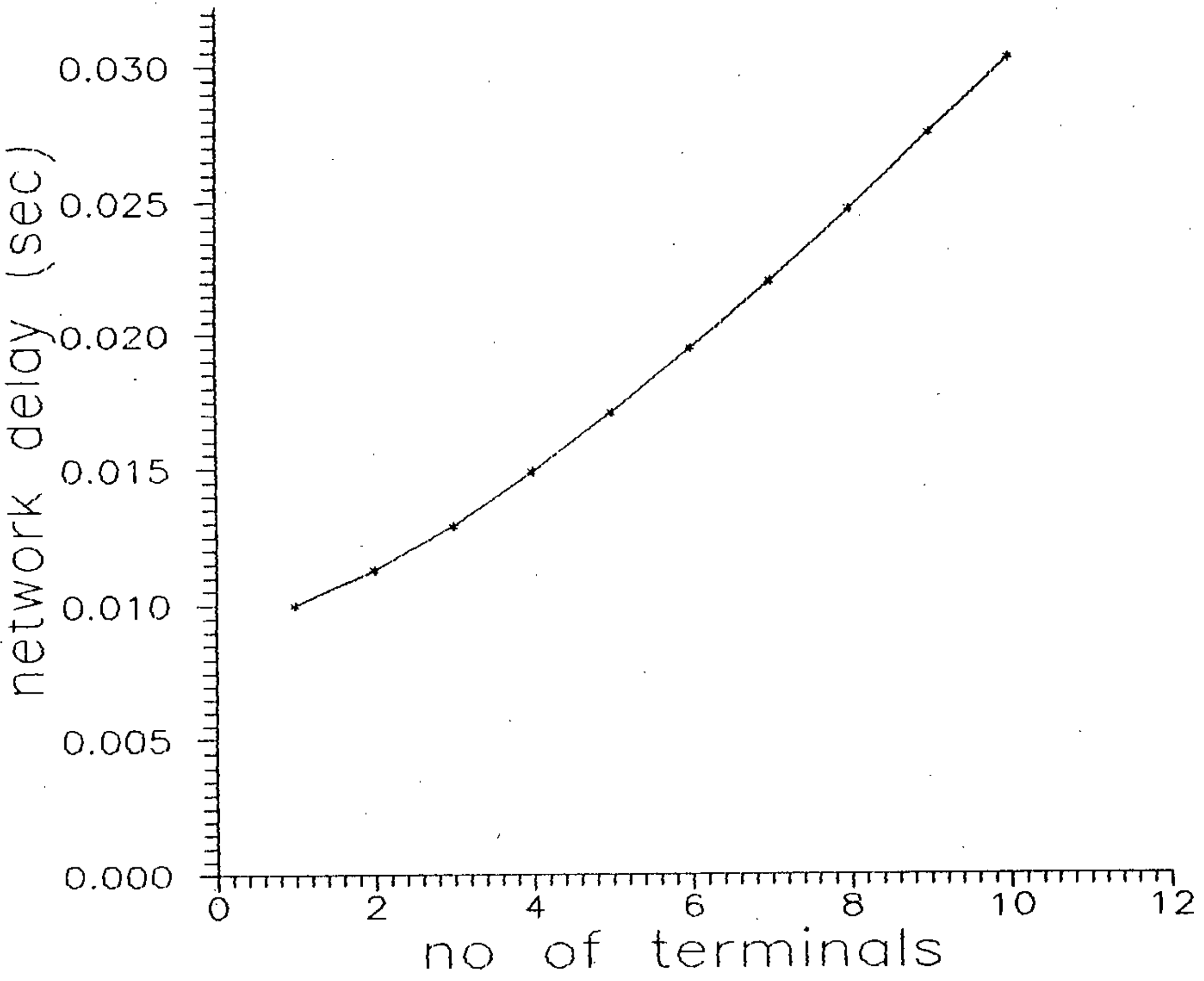


Figure 3.7 : network delay vs no of terminals for simple queuing network.

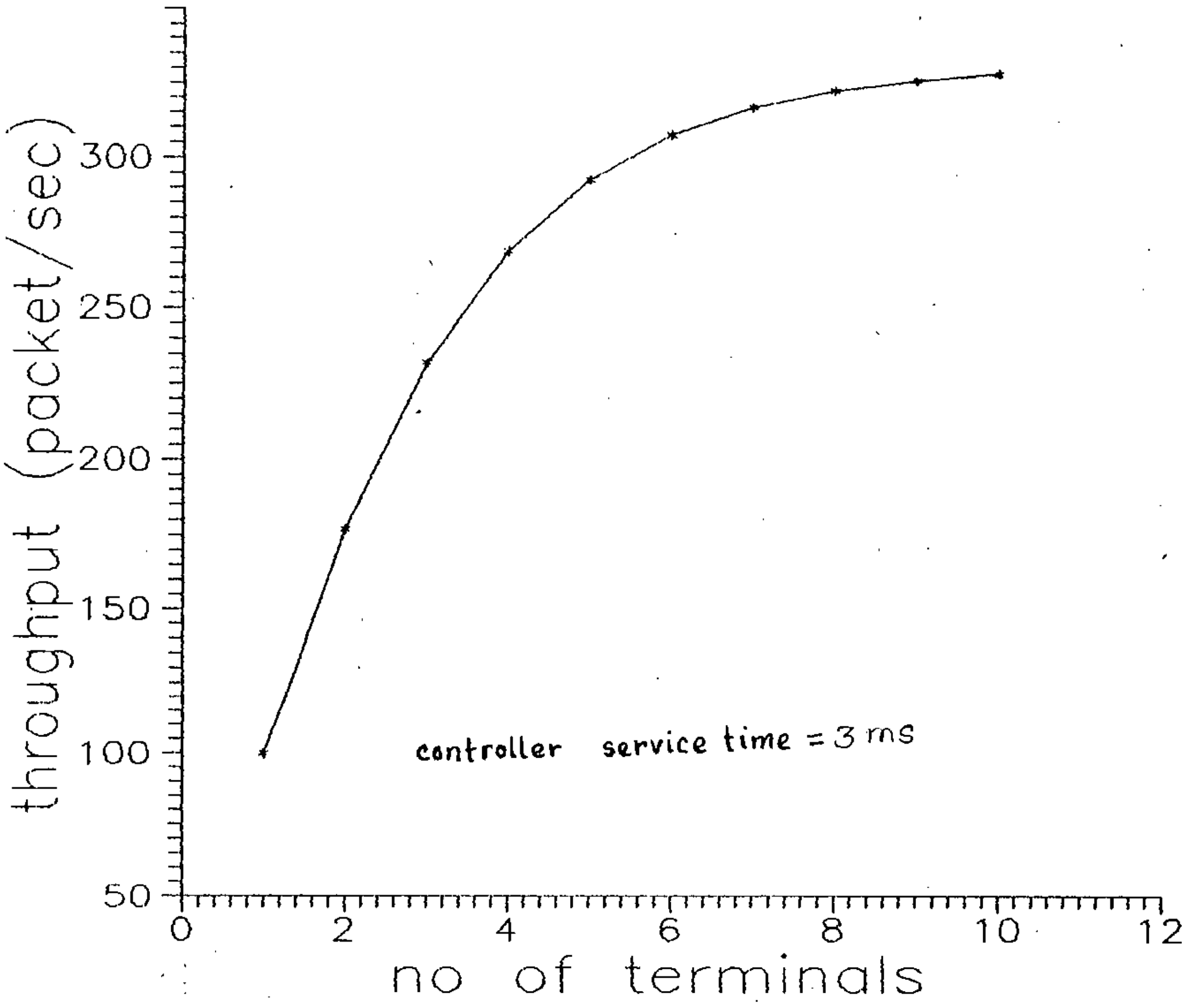


figure 3.8 : Throughput vs no of terminals for simple queuing network.

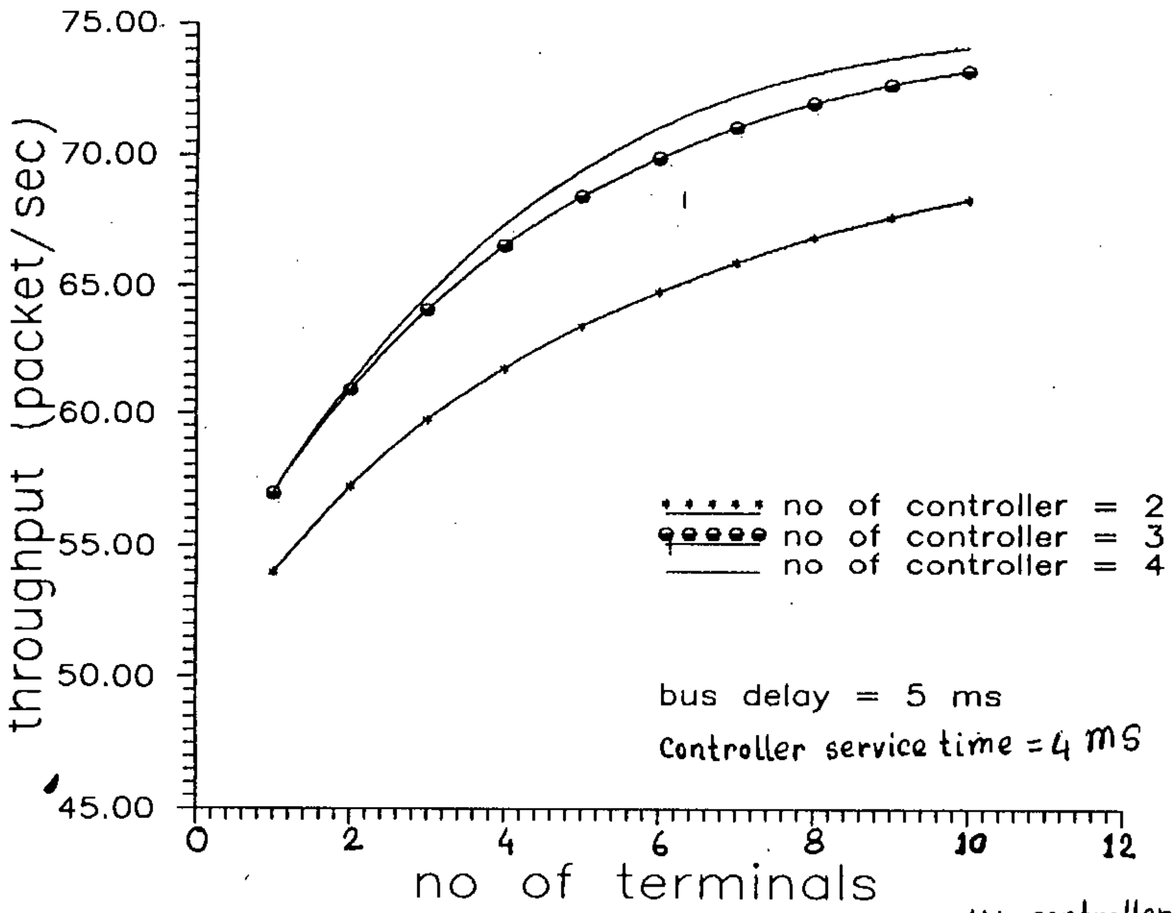


figure 3.9 : throughput vs no of terminals for multi-controller queuing network.

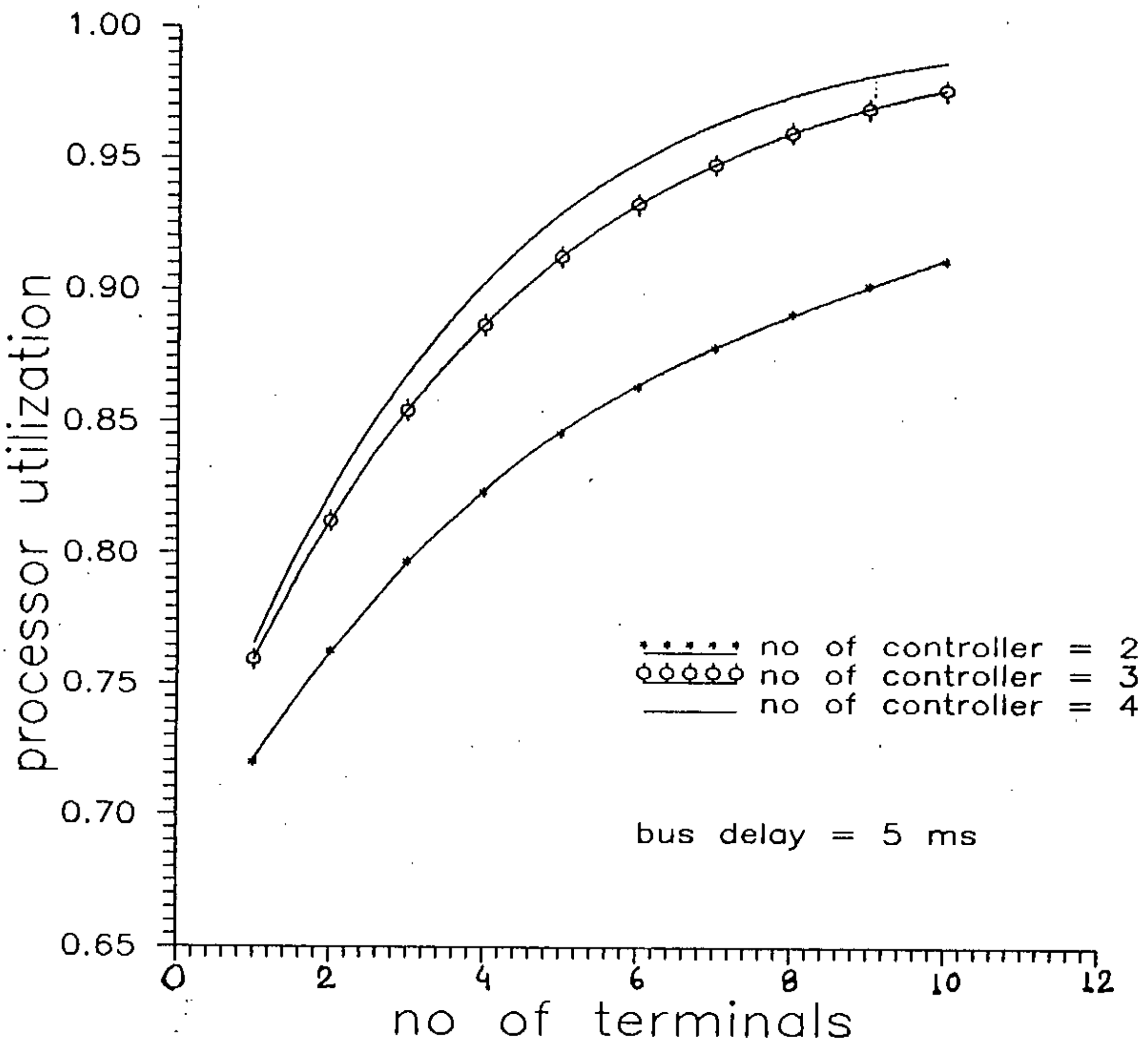


figure 3.10: protocol processor utilization vs no of terminals

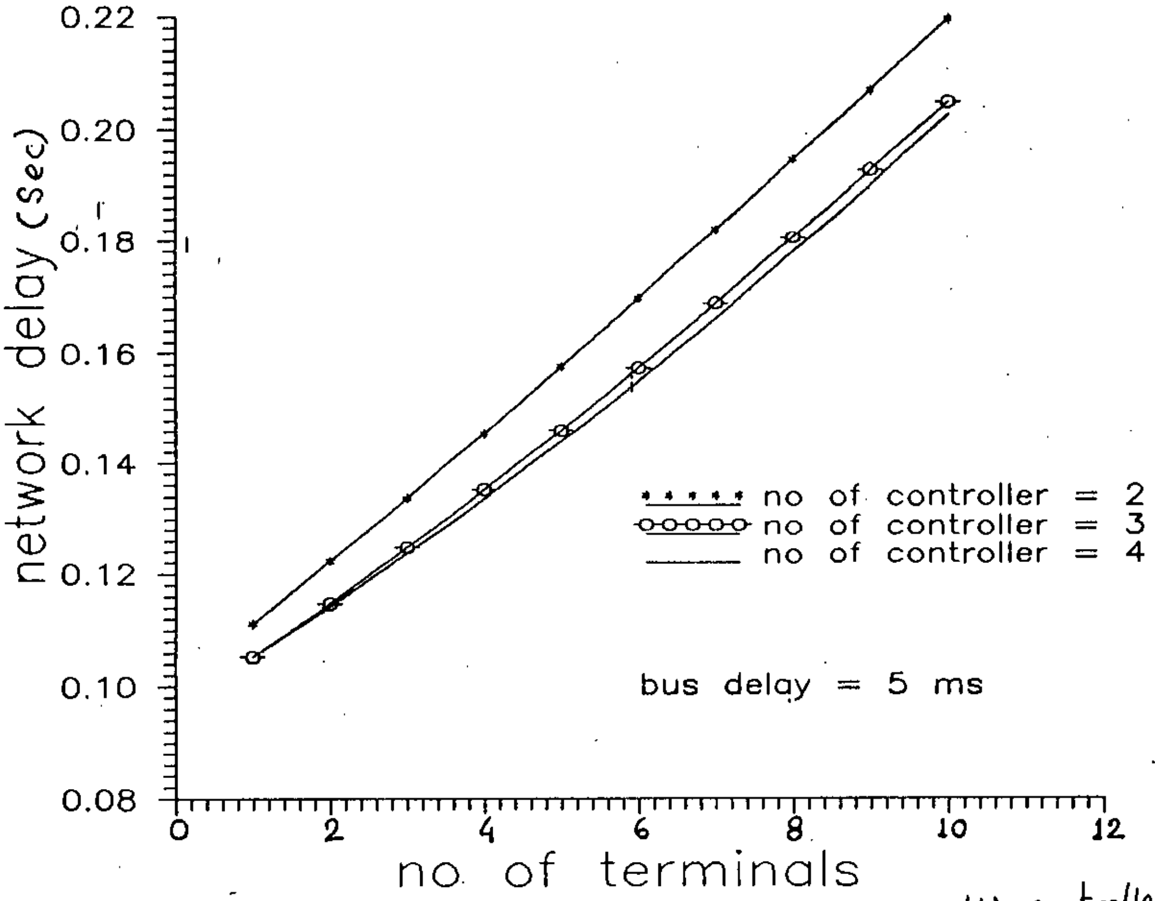


figure 3.11: network delay vs no of terminals for multi-controller queuing network.

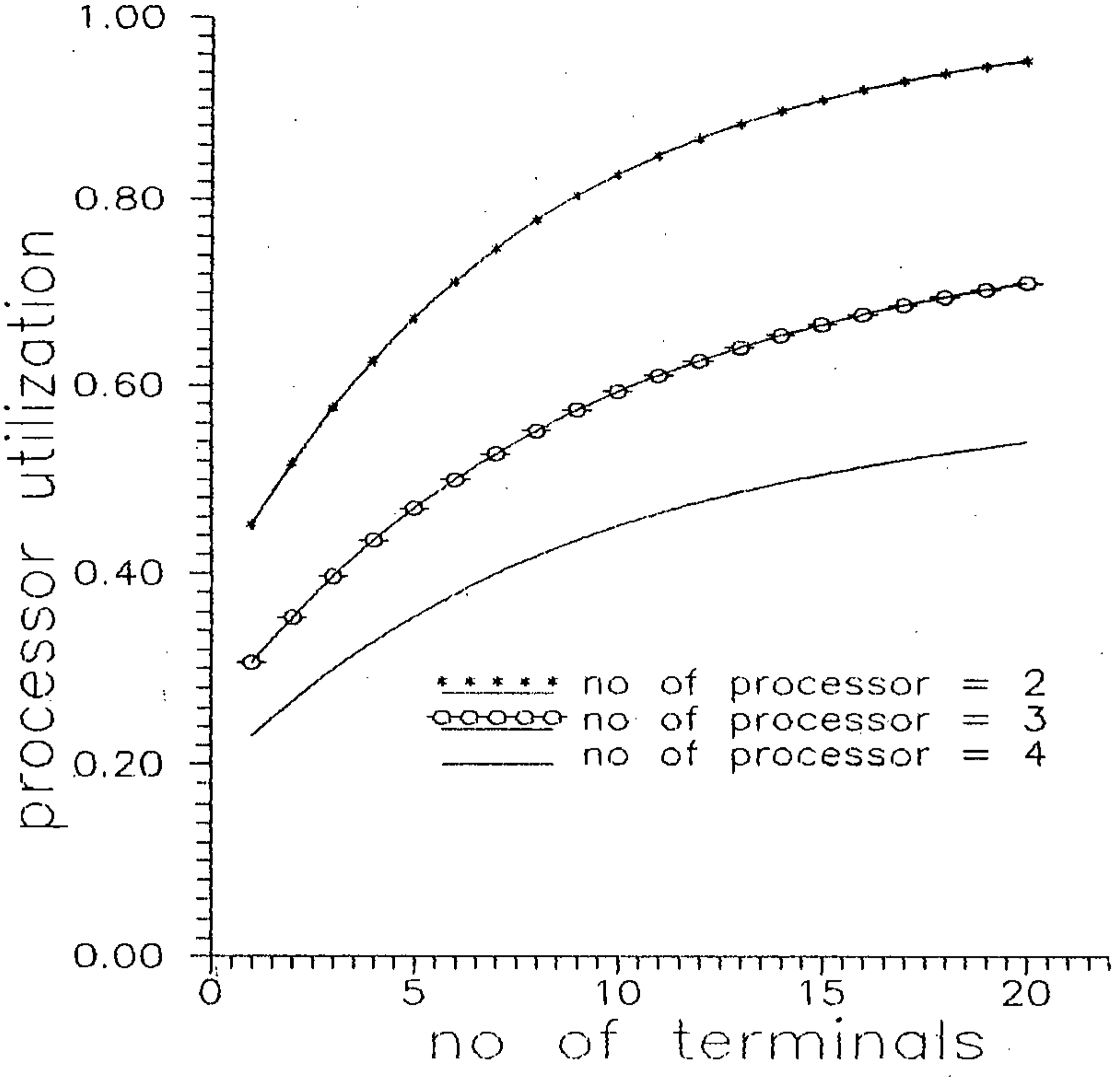


figure 3.12: protocol processor utilization vs no of terminals for multi-processor queuing network.

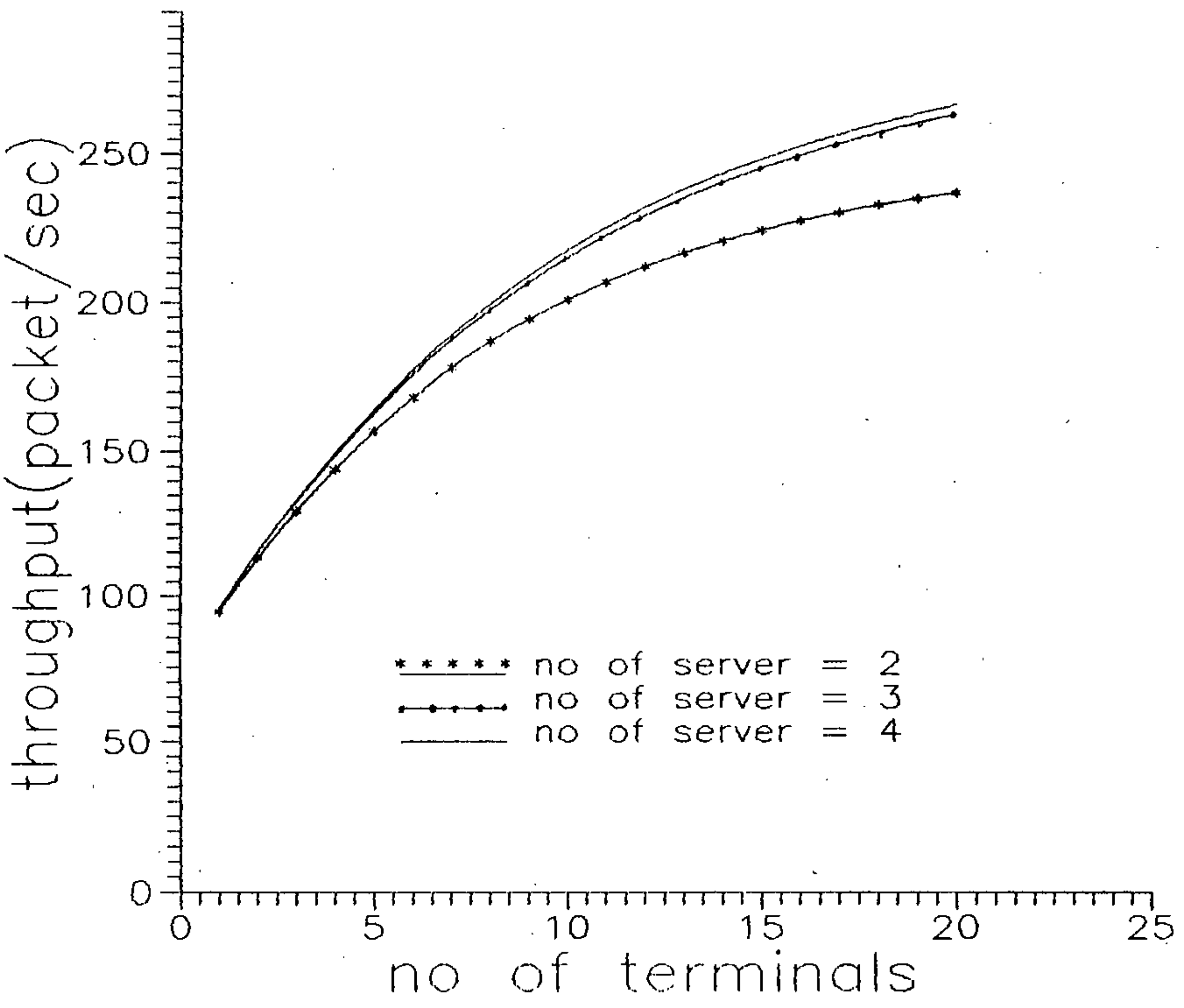


figure 3.13: throughput vs no of terminals for multiprocessor queuing network.

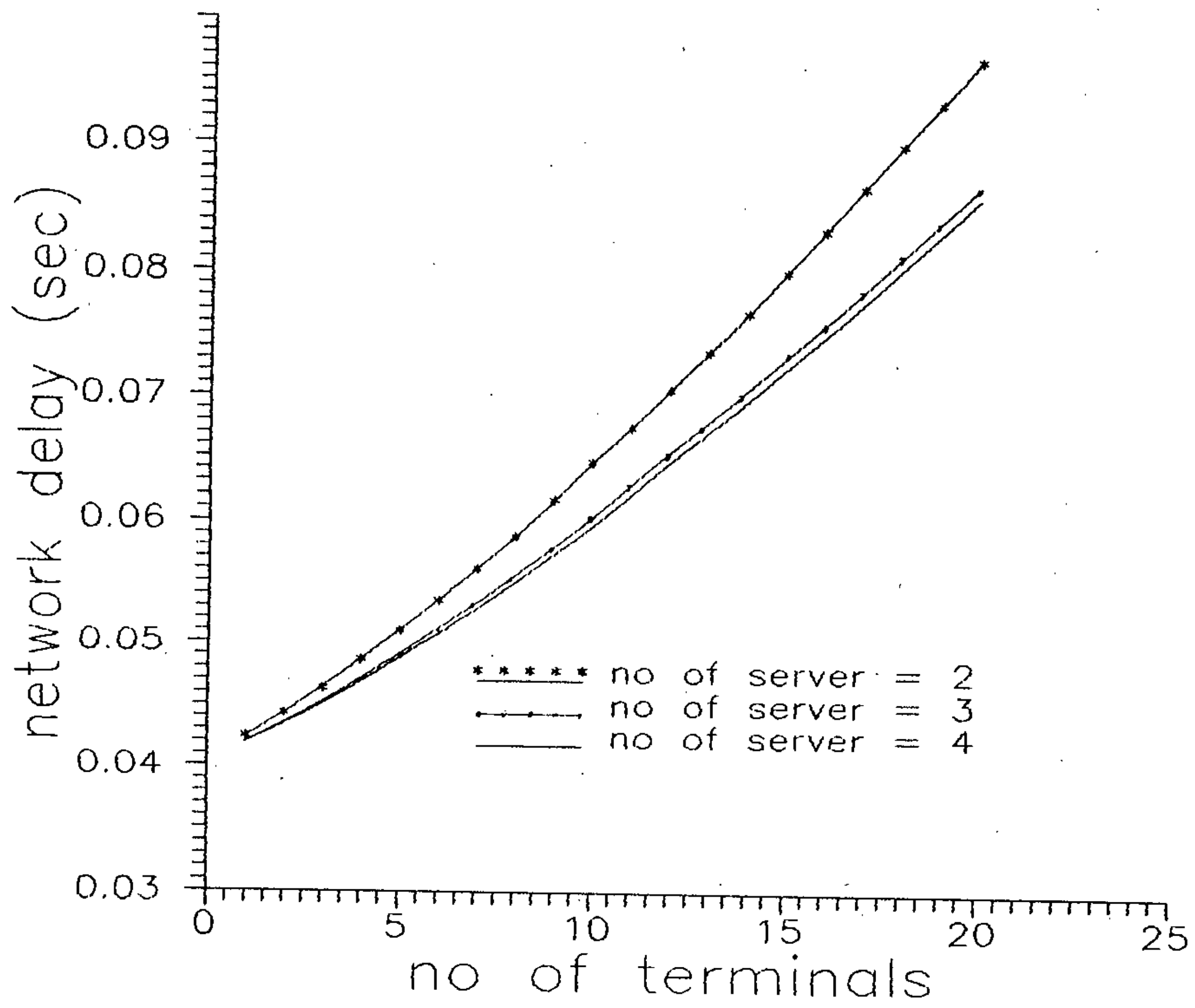


figure 3.14: network delay vs no of terminals for multi-processor queuing network.

CHAPTER 4

SOLUTION OF MULTI-CHAIN CLOSED QUEUING NETWORK

4.1 INTRODUCTION

A communication network model can be represented by an open queuing system if the system runs without flow control. But if the flow control is used then to represent the system, closed queuing network must be used. The most widely known protocol to control the flow in a network is called the window flow control. In window flow control, when a certain no of messages in a region of the network is unacknowledged then the new traffic is halted. A communication system with N number of switching nodes and R unidirectional virtual channels can be represented by a multi-chain queuing network. Each virtual channel has a source and a sink. Both source and sink are modeled by a simple queue. Each virtual channel is transformed into a closed chain consisting of source queue, sequence of forward route link queues, and back to source queue. The multi-chain queuing network studied in this chapter does not represent the entire network but only the interface to the network with different channels to serve different types of users. In the solution of multi-chain queuing network, it has been shown how the delay and throughput of each chain varies with the population of each chain.

4.2 DESCRIPTION OF THE MULTI-CHAIN QUEUING NETWORK

Figure 4.1 shows the multi-chain queuing network of a network interface unit having two channels to serve two types of users.

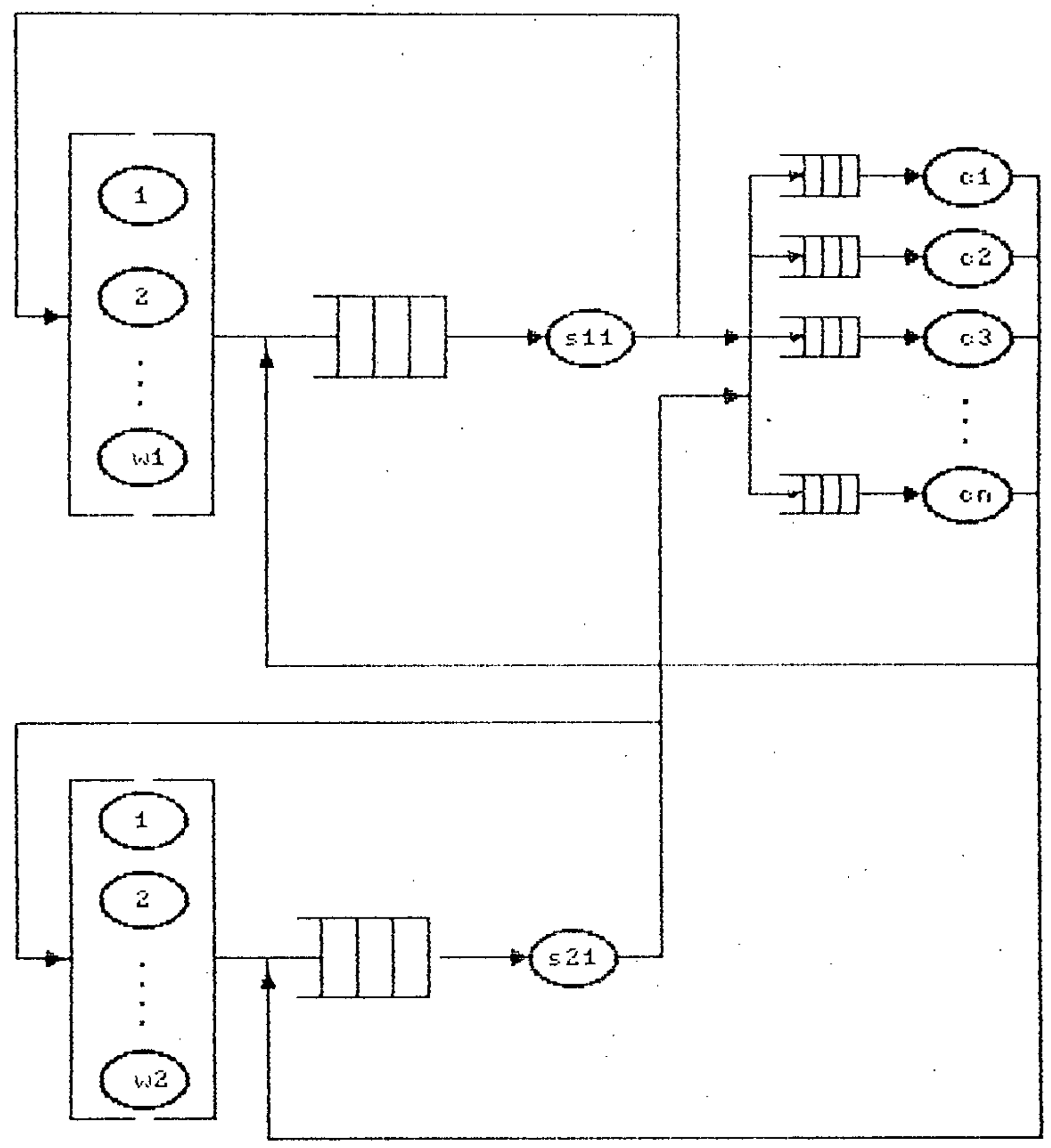


Figure4.1: Multi-chain queuing network.

While different types of users are connected through different channels with the network via network interface unit and there is a traffic flow control then multi-chain closed queuing network arises [10]. In figure 4.1 there are two chains representing two channels. There are processors for each chain to process each chain's messages and both the chain uses the common controller. Multiple common controllers have been used in parallel to increase the throughput of the network and to decrease the delay of the network. All the controllers are assumed to be identical in nature and there is equal probability of a message from the processor of each chain to the controllers. The controllers shown in figure 4.1 are the same as the equivalent controllers described in section 3.3.

4.3 SOLUTION PROCEDURE

In a closed multi-chain queuing network with product form solution an arriving customer observes the equilibrium solution of the queuing network with one less customer in the arriving customer chain.

In multi-chain case we have

R = Number of closed chains.

W^r = population of chain r, $r = 1,2, \dots R$.

$R(i)$ = set of chains visiting queue i $(i = 1,2,\dots N)$

$Q(r)$ = set of queues in chain r.

τ_i^r = mean service time of chain r message at queue i.

n_i = Mean queue size of queue i.

λ^r = Throughput of chain r.

t_i^r = Mean queuing time of chain r message at queue i.

We shall also use the notation $n_i^j(r^-)$ to denote the mean no of chain j message at queue i upon arrival of a chain r message. It is convenient to introduce $W = (w^1, W^2, \ldots W^R)$ and $W - e_r = (W^1, W^2, \ldots, W^{r-1}, W^r - 1, W^{r+1}, \ldots W^R)$.

To have product form solution of a multi-chain queuing network, some restrictions must be introduced on the service discipline and on the service time distribution.

If the scheduling is first in first out and all chains visiting queue i have the exponential service time distribution at that queue i,e $\tau_i^j = \tau_i$ for all $j \in R(i)$, in this case

$$t_i^r = \tau_i + \sum_{j \in R(i)} \tau_i * n_i^j(r^-) \qquad \ldots \qquad (4.1)$$

If the scheduling is processor sharing or last come first served preemptive-resume with a phase-type service time distribution then $_i^r$ may be different for different values of r. Then the delay time equation can be written as

$$t_i^r = \tau_i^r \left[1 + \sum_{j \in R(i)} n_i^j(r^-) \right] \qquad \ldots \qquad (4.2)$$

If it is a pure time delay with phase type then the delay time equation can be written as

$$t_i^r = \tau_i^r \qquad \ldots \qquad (4.3)$$

Equation (4.1) and (4.2) are identical. Summarizing the above equations

$$t_i^r = \tau_i^r \quad \text{if queue is a time delay.}$$

otherwise

$$t_i^r = \tau_i^r [1 + \sum_{i \in R(i)} n_i^j (W - e_r)]$$

$$\lambda^r = W^r / \sum_{i \in R(i)} t_i^r$$

$$n_i^r = \lambda^r * t_i^r$$

The above equations can easily be solved by an R dimensional recursion, starting with $n_i^r(0) = 0$, where

$i = 1,2,...N; r = 1,2,....R.$

Using the above equations it is easy to solve the multi-chain queuing network shown. There will be a two dimensional recursion.

4.4 RESULT AND DISCUSSION

In the solution of multi-chain queuing network, the scheduling has been considered to be of first come first served and all chains visiting to a queue have the same exponential service time distribution and the delay and throughput for varying number of terminals have been evaluated using Mean Value Analysis. To solve queuing network, Mean Value Analysis is very simple to implement and can be used for a large number of chain population.

The multi-chain queuing network in figure 4.1 has two chains and there are variable number of controllers in the network and the network has been solved for 3, 4 and 5 number of controllers. Number of parallel controllers have been used to increase the throughput and to decrease the delay of the queuing network. User request generation time for chain 1 terminals and chain 2 terminals are 6 ms and 4 ms respectively. Mean service time of each controller has been considered 5 ms i.e. mean service rate is 200 packets/sec which is compatible to a 2 - 10 Mbps LAN. Chain 1 supports 15 terminals and chain 2 supports 10 terminals and the service time of the protocol processor in chain 1 and chain 2 have been considered 2 ms and 3 ms respectively.

While considering the delay of a chain it includes the delay of the terminal, delay of the protocol processor and the delay of the common controllers. Figure 4.2 shows the delay of chain 1 with number of terminals in chain 2. As both the chain uses the common controllers, so increasing the number of terminals in chain 2 affects the delay in chain 1. For a fixed number of terminals in chain 2 it is observed that delay decreases as the number of controllers are increased. This is the aim of using multiple parallel controllers. If a network allows a fixed delay then to serve more terminals multiple controllers must be used. Figure 4.6 shows chain 1 delay with the number of terminals in chain 1 and the delay increases with the number of terminals. While considering chain 1 throughput, it is observed that the throughput decreases with the increasing number of terminals in chain 2 as shown in figure 4.3. But if we consider the chain 1 throughput with the increasing number of terminals in chain 1 as shown in figure 4.7 it is seen that throughput increases. This is because, in the previous case chain 2 message increases the queue length of the common controller for chain 1 message and accordingly the delay of the chain 1 message increases but as the number of terminals in chain 1 remains fixed so the throughput decreases. In the second case delay of the chain 1 message increases but as the terminals in chain 1 increases so the throughput increases. Similarly if the delay of the chain 2 has been considered as shown in figure 4.4 and 4.8 it is observed that there is no matter whether the number of terminals in chain 1 or chain 2 increases the delay increases and in both the case, for a fixed number of terminals delay decreases as the number of controllers increases. This is exactly the case for chain 1 also. If chain 2 throughput is considered as shown in figure 4.5 and 4.9 it is seen that chain 2 throughput increases as the number of terminals in chain 2 increases and it decreases as the number of terminals in chain 1 increases. This is for the same reason as in the case of chain 1 throughput as described previously.

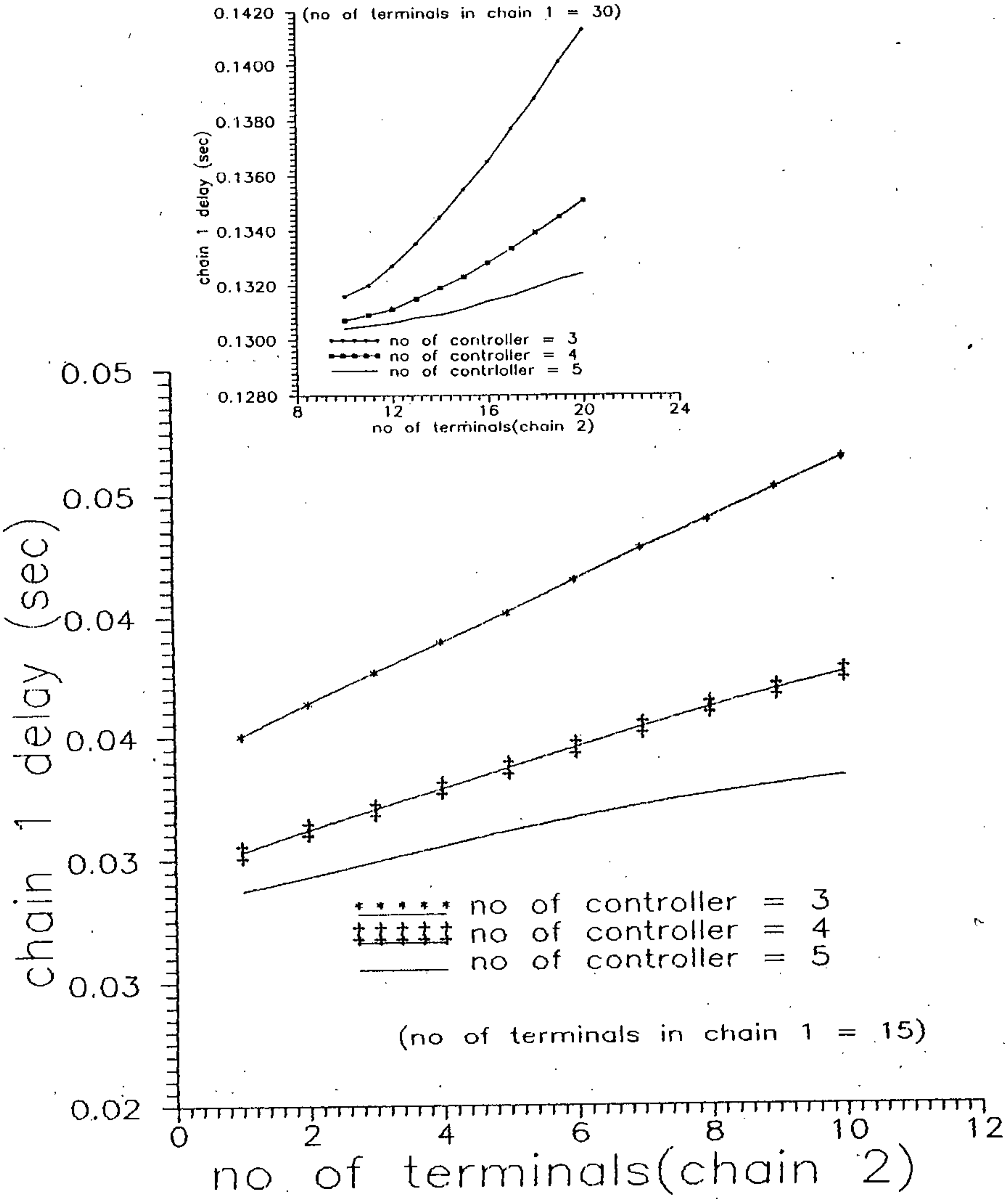


figure 4.2: chain 1 delay vs no of terminals in chain 2.

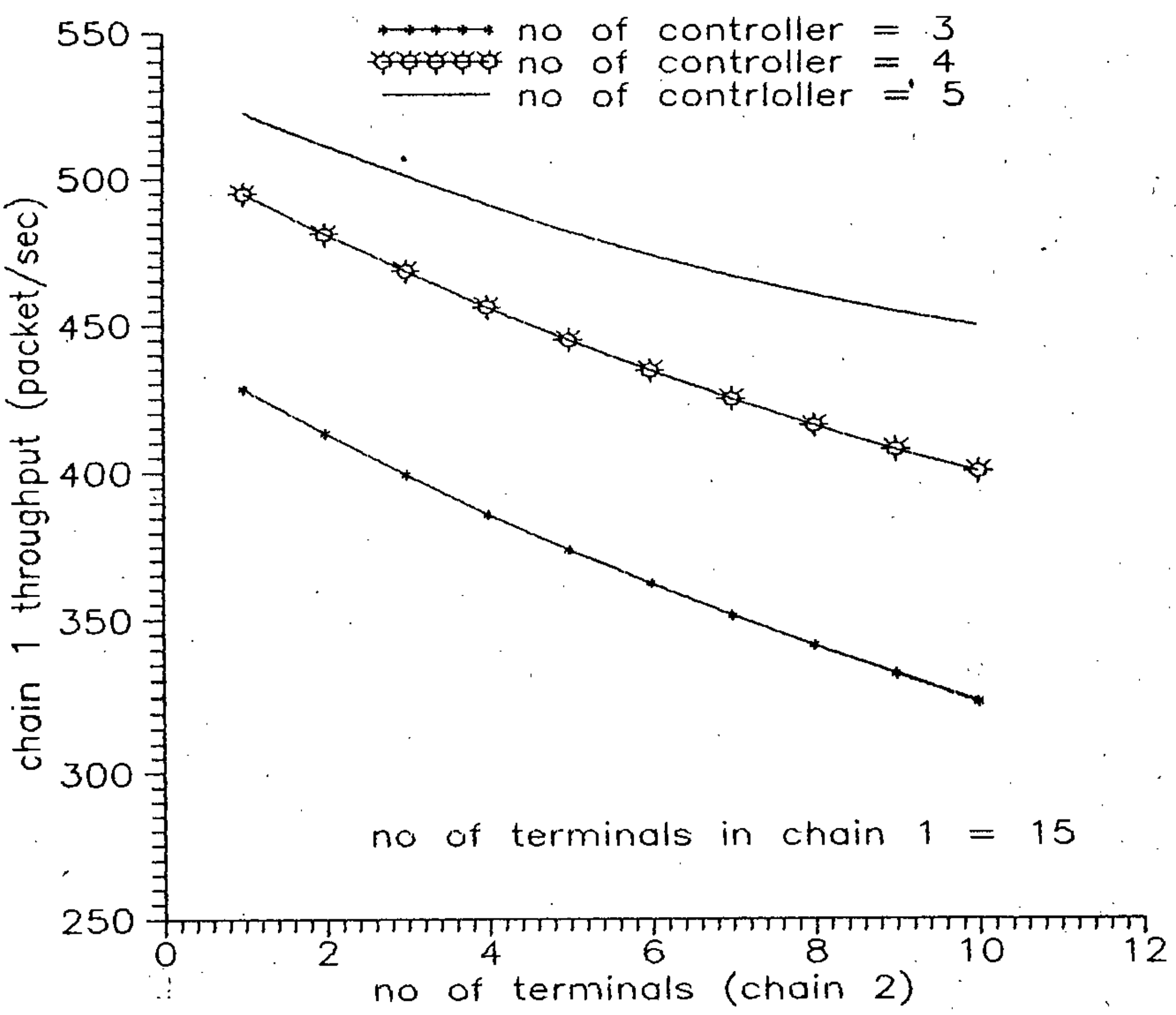


figure 4.3: chain 1 throughput vs no of terminals in chain 2

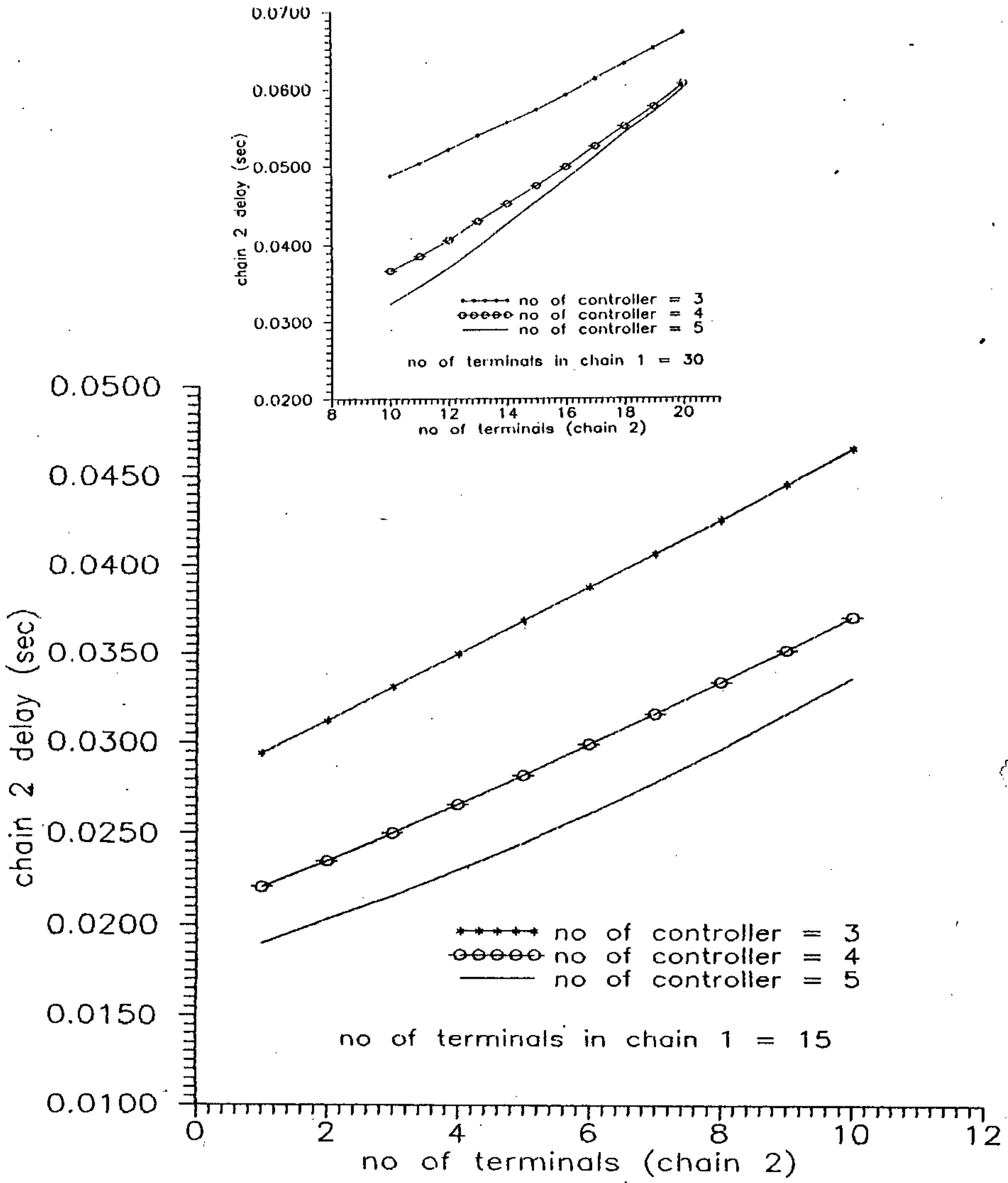


figure 4.4: chain 2 delay vs no of terminal in chain 2.

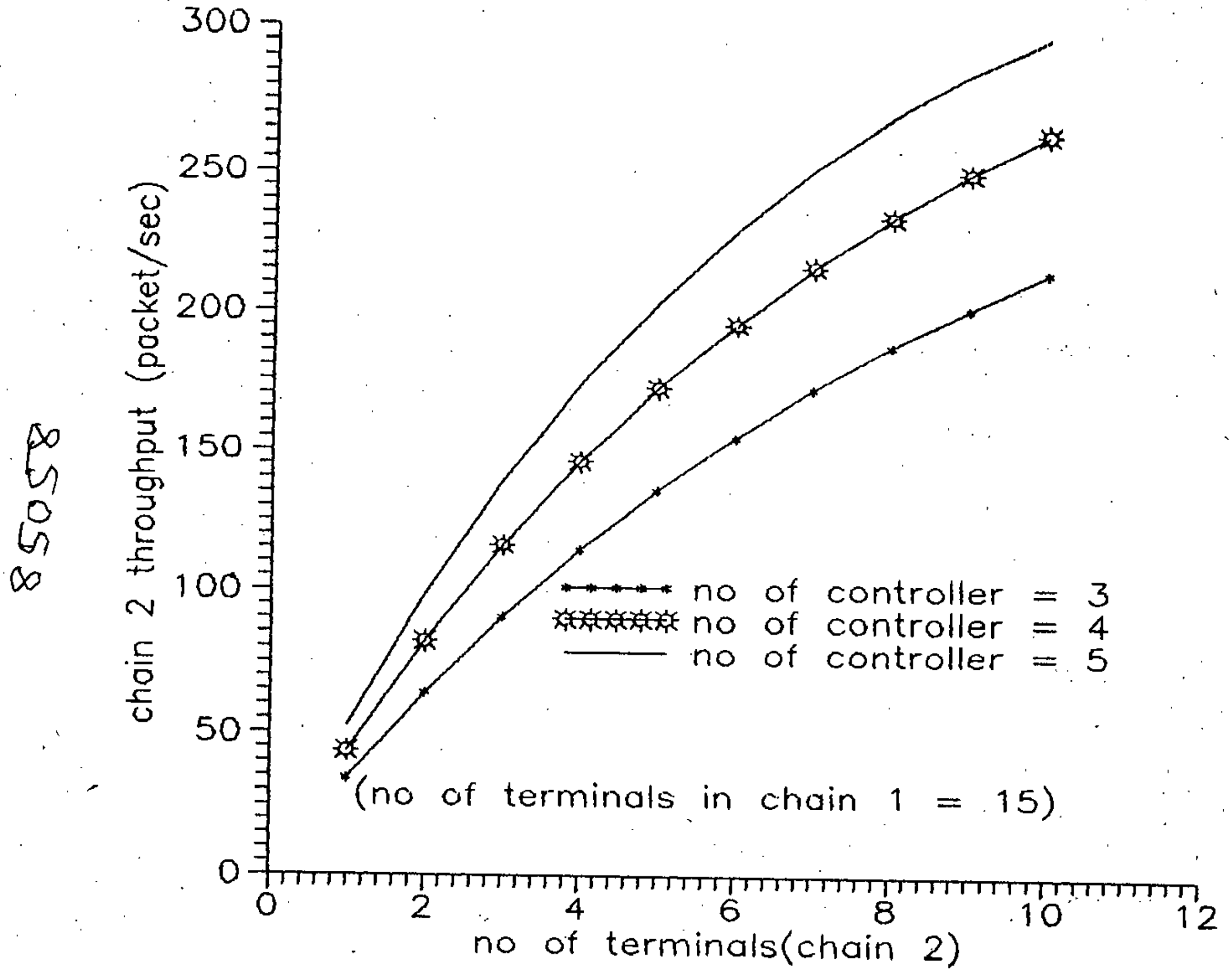
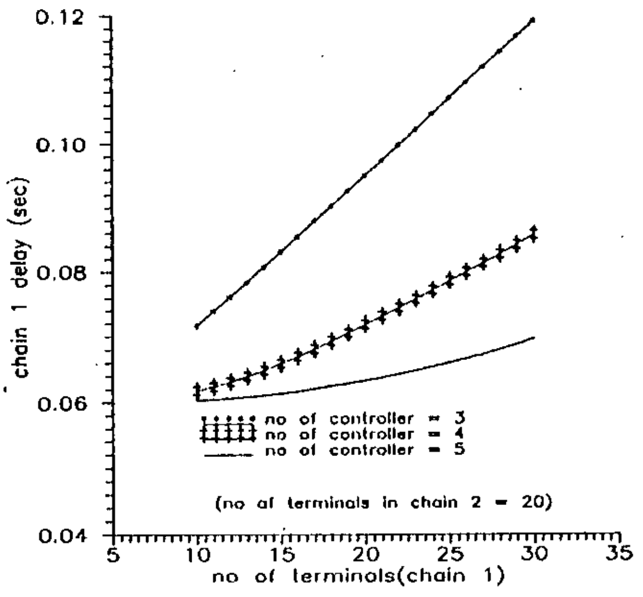


figure 4.5: chain 2 throughput vs no of terminals in chain 2.



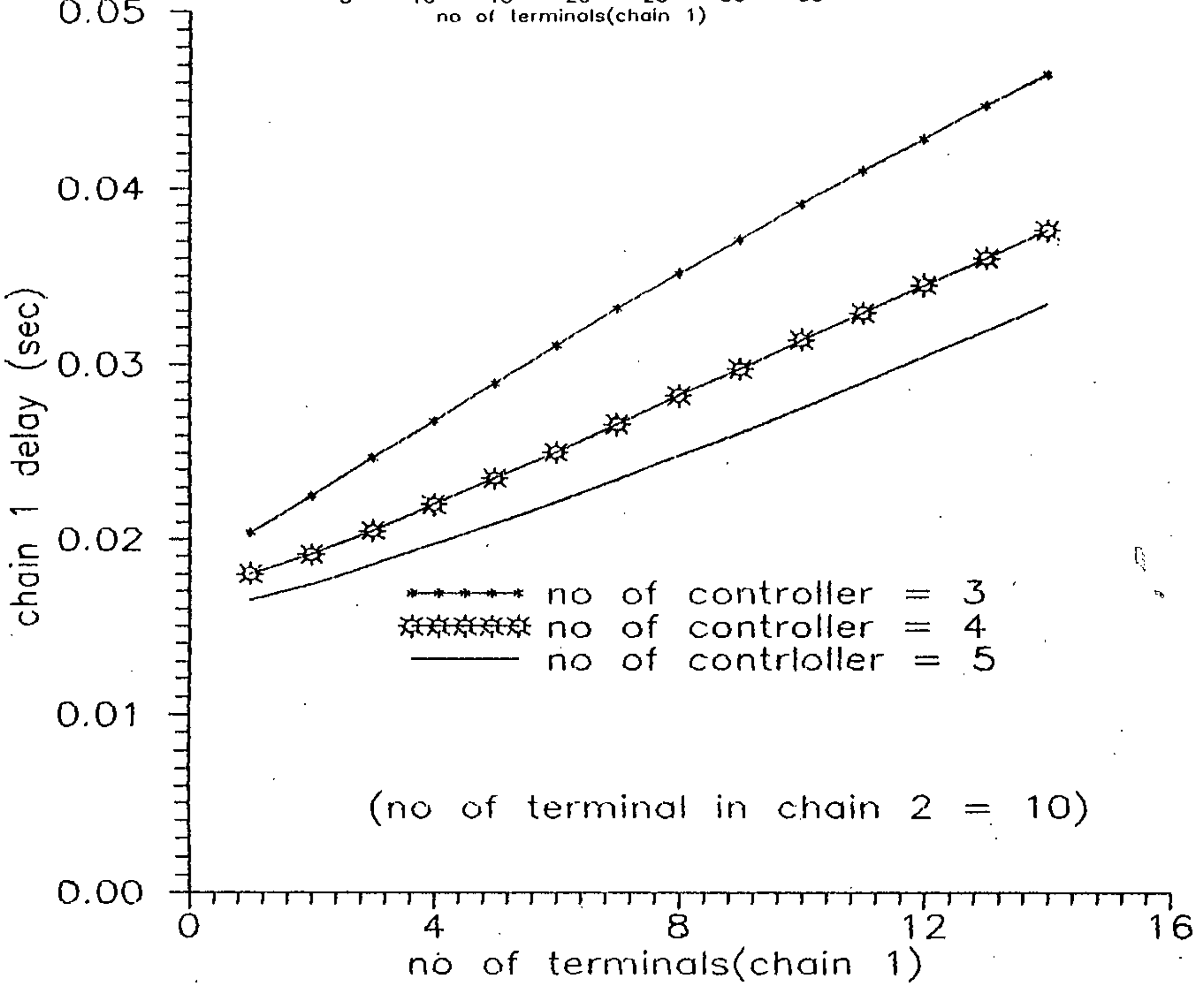


figure 4.6: chain 1 delay vs no of terminals in chain 1.

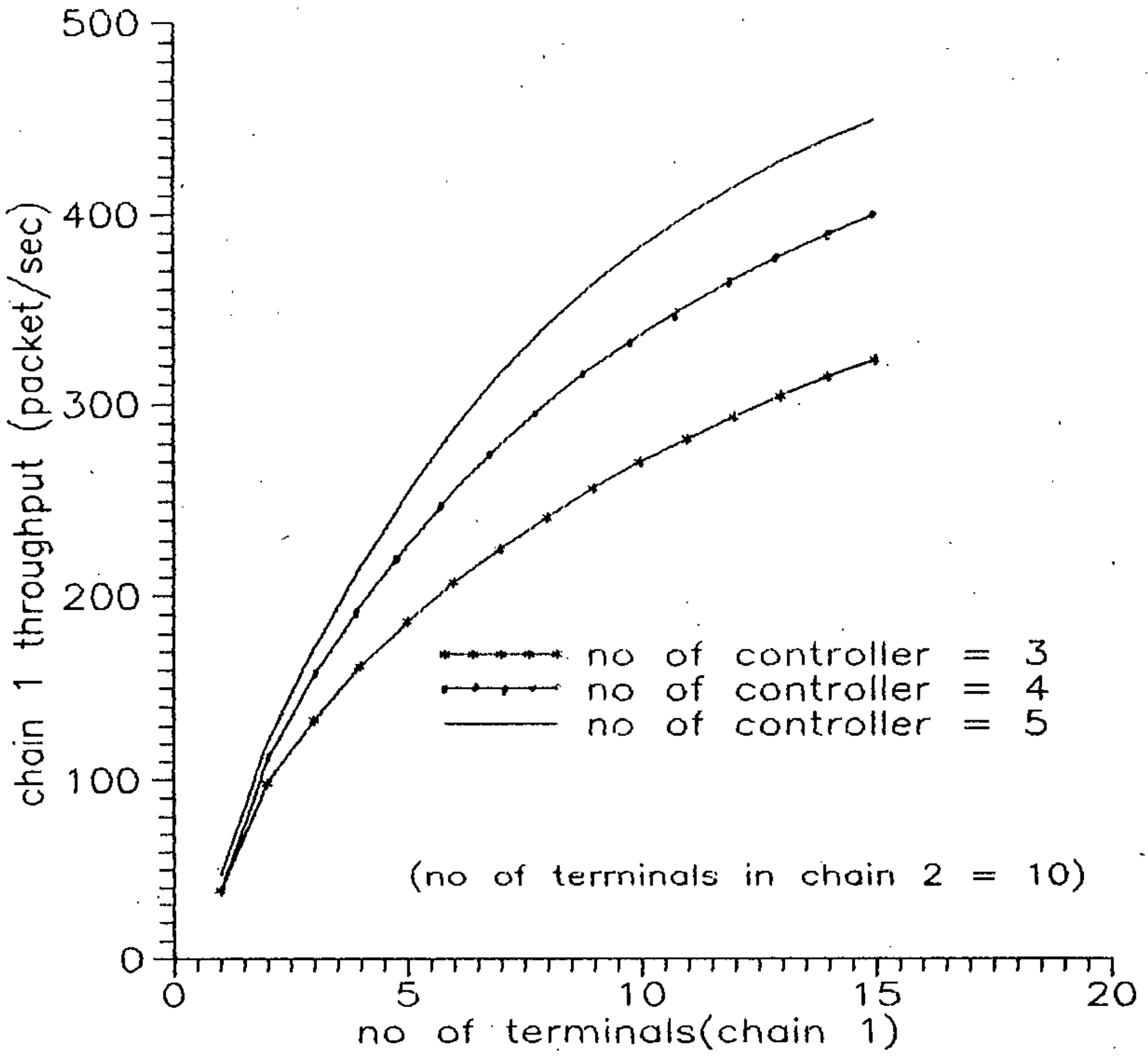


figure 4.7: chain 1 throughput vs chain 1 no of terminals

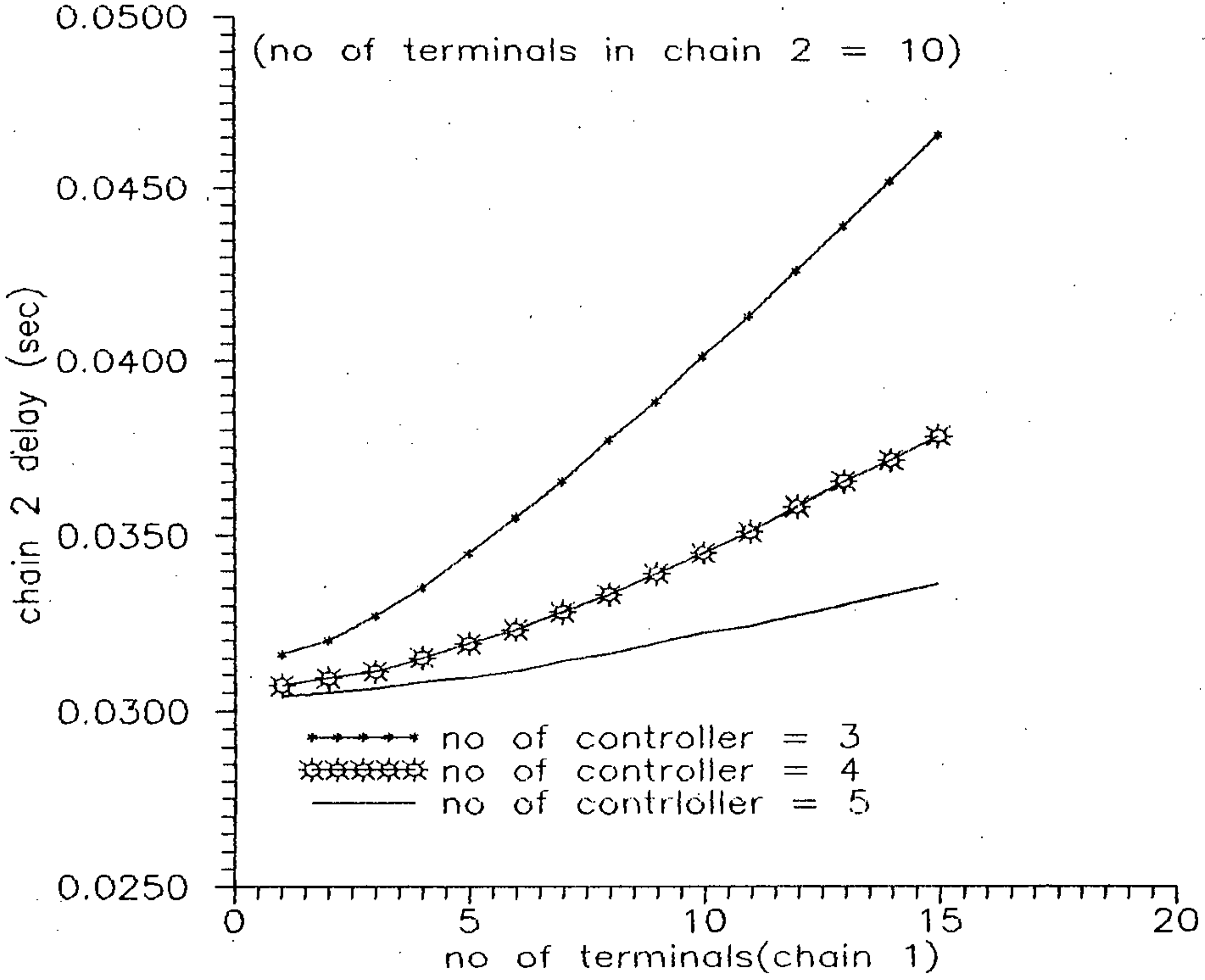


figure 4.8:chain 2 delay vs no of terminals in chain 1.

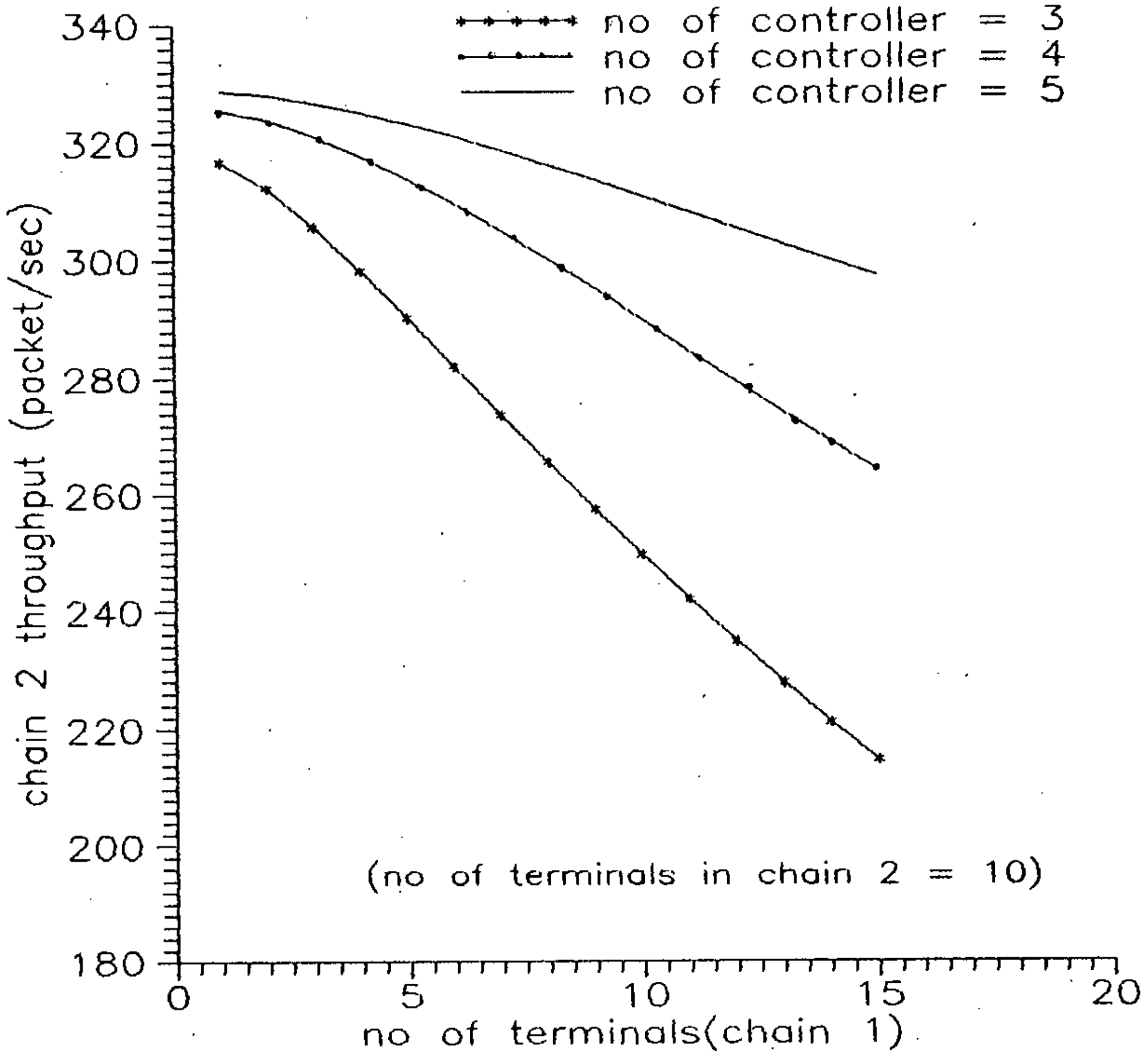


figure 4.9: chain 2 throughput vs choin 1 no of terminals.

CHAPTER 5

CONCLUSIONS

5.1 CONCLUSIONS:

Throughout the project-work analytical solution of node interface of computer network has been found by using different queuing models. To find the solution of single-chain queuing networks with multiple number of controllers and a number of processors, Buzen algorithm with load dependent behavior has been used. It has already been discussed in chapter 3 that Buzen algorithm can not be applied for the queuing network having three or more stages. So aggregation method has been used. Buzen algorithm is suitable for small number of population because if the number of population is large the space requirement and computational cost becomes very high.

The multi-chain queuing network of chapter 4 has been solved by using Mean Value Analysis. The method of solution of multi-chain queuing network using Mean Value Analysis is comparatively simpler than Buzen algorithm. Mean Value Analysis can be applied for a large number of population and a number of closed chains. Though there is R dimensional recursion for queuing network with R number of closed chains, the algorithm works very efficiently. For both single-chain and multi-chain queuing network it is observed that network delay decreases and throughput increases as the number of controller is increased. But there is a limit beyond which throughput can not be increased by increasing the number of controllers because in that case the protocol processor becomes saturated. So to increase the throughput beyond that limit parallel protocol processors must be used.

So it may be concluded that for high throughput and low delay multi-controller interface should be used. If the throughput is expected much higher then multi-processor and multi-controller interface may be used.

5.2 RECOMMENDATIONS:

It has already been discussed in chapter 1 that to integrate voice packet and data packet using broadband packet switch then a priority queue for voice packet must be used. The analysis of queuing network, in that case will be more complex. So the study may be enhanced for integrated voice and data packets.

The queuing network used so far for the analysis has maximum three stages and Mean Value Analysis may be used to solve the queuing network of any number of stages.

In the analysis of multi-chain queuing network, two chains has been used and number of recursion is two. It may be used for any number of chains and in that case number of recursion will be increased.

The performance indices found in the analytical solution may be checked by constructing the interface unit in the field. The throughput and network delay may be different with the measured value of the queuing model but the requirement of the network interface unit may be estimated from the study of the queuing models.

REFERENCES

REFERENCES

[1] Tanenbaum, A.S, "Computer Networks" Prentice Hall of India(pvt) Limited, Newdelhi- 110001, 1988.

[2] Stallings, William, "Data and Computer Communications" Macmillan Publishing Company, New York, 1988.

[3] Thilakam, K. and Jhunjhunwala, A., "Proposed high speed packet switch for broadband integrated networks" Computer Communications, Vol. 12, No. 6, December 1989.

[4] Ferrari, D.,Serazzi, G. and Zeigner, A., "Measurement and Tuning of Computer Systems" Prentice Hall Inc, Englewood cliffs, NJ07632 (1983).

[5] Reiser, M., Lavenberg, S. S., "Mean Value Analysis of closed multichain queuing networks" J. ACM, vol. 27, n. 2, pp. 313-322, April 1980.

[6] Wilmarth, N., "Two-chip set unlocks Ethernet gateway" Electronic Design, June 23, 1983.

[7] Fontine, J.A., "Controller and micro team up for smart Ethernet node" Computer Design, February 1984.

[8] Ohr, S., "Ethernet chips hold the lead in VLSI scramble for local networks, Electronic Design, June 23, 1983.

[9] Trevedi, K.S., "Probability and statistics with reliability, queuing, and computer science applications, Prentice-Hall Inc, Englewood cliffs, N.J.07632.

[10] Reiser, M., "A queuing network analysis of computer communication networks with window flow control" Transaction on Communications, Vol. COM-27, No. 8, August 1979.

APPENDIX

```
program mvalue(input,output);

(* This is the solution of a simple
queuing network with single protocol processor
and single controller*)

var
   n1:ARRAY[0..25] OF real;
   n2:ARRAY[0..25] OF real;
   t1:ARRAY[1..25] OF real;
   t2:ARRAY[1..25] OF real;
   lamda:ARRAY[1..25] OF real;
   u1:ARRAY[1..25] OF real;
   r:ARRAY[1..25] OF real;
   tow1,tow2,d:real;
   n,i,j,k:integer;
   pp:text;

(* variable definition

   n1 : mean queue length of protocol processor.

   n2 : mean queue length of controller.

   t1: delay of message in protocol processor.

   t2: delay of message in controller.

   r: network delay.

   tow1: average service time of protocol processor.

   tow2: average service time of controller.

   n: no of terminals.

   lamda: throughput rate of the network.*)

begin

     assign(pp,'out');
     rewrite(pp);

     writeln('Enter the no of terminals ');
     readln(n);

     tow1 := 0.002;
     tow2 := 0.003;
     d:= 0.005;      (* bus delay *)
     i:= 1;
```

```
  n1[0] := 0;
  n2[0] := 0;

  for i := 1 to n do
      begin
         t1[i] := tow1 * (1 + n1[i-1]);
         t2[i] := tow2 * (1 + n2[i-1]);
         r[i] := t1[i] + t2[i] + d;
         lamda[i] := i/r[i];
         n1[i] := lamda[i] * t1[i];
         n2[i] := lamda[i] * t2[i];
         u1[i] := lamda[i] * tow1;
         writeln(pp,i:2,'   ',u1[i]:8:4,'   ',r[i]:8:4,'    ',
         lamda[i]:8:4);
     end;
end.
```

```
program buzen(input,output);

(* solution of a queuing network
with several controllers using
flow equivalant aggretion.*)


var

 rn:ARRAY[1..25,1..25] OF real;
 rr:ARRAY[1..25,1..25] OF real;
 xn:ARRAY[1..25,1..25] OF real;
 nq:ARRAY[0..25,0..25] OF real;
 u:ARRAY[1..25,1..25] OF real;
 s:ARRAY[1..25] OF real;
 yys:ARRAY[0..25,0..25] OF real;
 a:ARRAY[0..25,0..25] OF real;
 aa:ARRAY[0..25,0..25] OF real;
 d,sum:real;
 i,n,lnt,lnc,nc:integer;
k,j: integer;
r: ARRAY[1..25] OF real;
x: ARRAY[1..25] OF real;
ue: ARRAY[1..25] OF real;
uc: ARRAY[1..25,1..25] OF real;
mc: ARRAY[1..25] OF real;
yy: ARRAY[0..25,0..25] OF real;
y: ARRAY[0..25] OF real;
q1: ARRAY[1..25] OF real;

mp:real;
qq,qq1,qq2:text;


(* variable definitions:


   rn:delay of different controllers of
      equivalant network.

   rr:total delay of equivalant network

   xn:throughput of equivalant network.

   nq:queue length of different controllers of equivalant network.

   u:utilization of different controllers of equivalant network.

   s: average service time of protocol processor and controllers.
```

```
  r: delay of the network.

  x: throughput of the network.

  ue: utilization of the protocol processor.

  uc: utilization of controllers

  mc: service rate of protocol processor and  controllers.

  yy: variable to find normalizing constant.

   y: variable used to find normalizing constant.

  q1:probability of a job from processor to controllers

  d: bus delay *)


begin


  assign(qq,'r.txt');
  rewrite(qq);
  assign(qq1,'r1');
  rewrite(qq1);

  writeln( 'Enter no of terminals  ');
  readln(lnt);
  writeln('Enter no of controllers  ');
  readln(lnc);

(* assigning service rate to different controllers*)

(* mc[1] = service rate of the protocol processor
   s[1] = service time of the protocol processor
   mc[i] where i = 2,..lnc+1 is the service rate of different controllers.*)

 (* Assigning service rate to different controllers*)

 mc[1] := 500;
 s[1] :=1/mc[1];
 mc[2] := 250;
 for i := 2 to lnc+1  do
    s[i] := 1/mc[2];

(* Initialization of queue length of each controllers*)

 for i := 2 to lnc+1 do
   begin
     nq[i,0] := 0;
     a[i,0] := 1;
     aa[i,0] := 1;
   end;

 d := 0.005;
 i := 2;
 n := 1;
 for i := 2 to lnc+1 do
 begin
    for n := 1 to lnt do
    begin
      rn[i,n] := s[i] * (1 + nq[i,n-1]);
      rr[i,n] := rn[i,n] + d;
      xn[i,n] := n/rr[i,n];
      a[i,n] := xn[i,n]/xn[i,1];
      aa[i,n] := aa[i,n-1] * a[i,n];
      u[i,n] := xn[i,n] * s[i];
      nq[i,n] := u[i,n] * (1 + nq[i,n-1]);
    end;

 end;


  q1[1] := 0.15;

(* q1[1] = probability of departure
   q1[i] = probability of a job from
         protocol processor to controller i
         where i = 2,..lnc+1 *)

  writeln(qq1);
  writeln(qq1);
  writeln(qq1, '  no of terminals = ',lnt:3);
  writeln(qq1);
  writeln(qq1,'  no of controllers = ',lnc:3);
  writeln(qq1);


(* assigning the value of probability and service rate
 of different controllers.*)

   for i := 2 to lnc+1 do
   q1[i] := (1-q1[1])/lnc;


(* finding the values of y'k to find
 normalizung constant*)

 y[1] := 1.0 ;
 for i := 2 to lnc + 1 do
```

```
   y[i] := (mc[1]/xn[i,1]) * q1[i];

(* finding the value of the normalizing constant *)

n := 1;
for n:= 0 to lnt  do
   yy[1,n] := 1;

 for i := 1 to lnc + 1 do
    yy[i,0] := 1;

for  i := 2 to lnc + 1 do
   begin
     for n := 1 to lnt   do
        begin
            sum:= 0;
           for j:= 0 to n do
              begin
                yys[i,j] := ((exp(j*ln(y[i])))/aa[i,j])*yy[i-1,n-j];
                sum := sum + yys[i,j];
               end;
            yy[i,n] := sum;
          end;
   end;


    writeln(qq1);
    writeln(qq1,' ','No of terminals','  ','utilization of the processor');
    writeln(qq1,' ','***************','  ','****************************');
    writeln(qq1);


 (* finding the utilization of the protocol processor *)

 for n := 1 to lnt  do
    begin
       ue[n] := (yy[lnc+1,n-1]/yy[lnc+1,n]);
       writeln(qq1,'     ',n:3,'                         ',ue[n]:8:4);
     end;


(* finding the utilizations of
  different controllers.*)

  for i := 2 to lnc + 1  do
     begin
       for n := 1 to lnt do
         begin
              uc[i,n] := ue[n] * y[i];
              write(qq);
              write(qq,'   ',uc[i,n]:8:4);
        end;
      writeln(qq);
   end;


(* Finding the throughput and network delay *)

writeln(qq1);
writeln(qq1);
writeln(qq1,'   no of terminals   throughput   network delay');
writeln(qq1,'   ***************   **********   *************');


for n := 1 to lnt do
     begin
       x[n] :=mc[1]*ue[n]*q1[1];
       r[n] := (n/x[n]);
       writeln(qq1,'       ',n:3,'
       ,x[n]:8:4,'                   ',r[n]:8:4);

     end;
 end.
```

```
program buzen1(input,output);


(* solution of multi-processor queuing network
with several controllers using
flow equivalant aggretion.*)


var

 rn:ARRAY[1..25,1..25] OF real;
 rr:ARRAY[1..25,1..25] OF real;
 xn:ARRAY[1..25,1..25] OF real;
 nq:ARRAY[0..25,0..25] OF real;
 u:ARRAY[1..25,1..25] OF real;
 s:ARRAY[1..25] OF real;
 yys:ARRAY[0..25,0..25] OF real;
 a:ARRAY[0..25,0..25] OF real;
 aa:ARRAY[0..25,0..25] OF real;
 r: ARRAY[1..25] OF real;
 x: ARRAY[1..25] OF real;
 ue: ARRAY[1..25] OF real;
 uc: ARRAY[1..25,1..25] OF real;
 mc: ARRAY[1..25] OF real;
 yy: ARRAY[0..25,0..25] OF real;
 y: ARRAY[1..25] OF real;
 q1: ARRAY[1..25] OF real;
 beta:ARRAY[0..25] OF real;
 d,sum:real;
 i,n,lnt,lnc:integer;
 k,j,ci: integer;
 qq,qq1,qq2:text;

(* variable definitions:

   rn:delay of different controllers of
      equivalant network.

   rr:total delay of each equivalant network

   xn:throughput of each equivalant network.

   nq:queue length of different controllers of equivalant network.

   u:utilization of different controllers in equivalant network.

   s:average service time of different controllers.

   r: average delay of the network.

   x: throughput of the network.
   ue: utilization of protocol processor.

   uc: utilization of controllers.

   mc: average service rate of protocol processor and  controllers.

   yy: variable to find normalizing constant.

    y: variable used to find normalizing constant.*)


function fact(m:integer):integer;

(* function fact calculate the factorial of a number *)

var
   res:integer;

   begin
      res := 1;
      i := 1;
      for i := 1 to m do
        begin
          res := res * i;
         end;
      fact := res;
    end;


 begin

   assign(qq,'r');
   rewrite(qq);
   assign(qq1,'r1');
   rewrite(qq1);

   writeln( 'Enter no of terminals   ');
   readln(lnt);
   writeln('Enter no of controllers  ');
   readln(lnc);

   writeln('enter no of processors');
   readln(ci);

  (* assigning service rate to different controllers*)

  (* mc[1] = average service rate of each processor
     s[1] = average service time of each processor *)


   (* Assigning service time to different controllers*)
 mc[1] := 500;
 s[1] :=1/mc[1];
 mc[2] := 200;
 for i := 2 to lnc+1  do
      s[i] := 1/mc[2];

(* Initialization of queue length of each controllers*)


 for i := 2 to lnc+1 do
 nq[i,0] := 0;

 for i := 1 to lnc + 1 do
  begin
    a[i,0] := 1;
    aa[i,0] := 1;
  end;

 d := 0.005;

(* d = bus delay *)

 for i := 2 to lnc+1 do
 begin
    for n := 1 to lnt do
    begin
      rn[i,n] := s[i] * (1 + nq[i,n-1]);
      rr[i,n] := rn[i,n] + d;
      xn[i,n] := n/rr[i,n];
      a[i,n] := xn[i,n]/xn[i,1];
      aa[i,n] := aa[i,n-1] * a[i,n];
      u[i,n] := xn[i,n] * s[i];
      nq[i,n] := u[i,n] * (1 + nq[i,n-1]);
    end;

 end;


  q1[1] := 0.25;

 (* q1[i] = probability of a job from processor to controller i
 where i = 2,...lnc+1
   q1[1] = probability of departure*)


  writeln(qq1);
  writeln(qq1);
  writeln(qq1, '  no of terminals = ',lnt:3);
  writeln(qq1);
  writeln(qq1,'  no of controllers = ',lnc:3);
  writeln(qq1);
 writeln(qq1,'  no of protocol processors = ',ci:3);
 writeln(qq1);

(* assigning the value of probability
 to different controllers.*)


 for i := 2 to lnc + 1 do
 q1[i] := (1-q1[1])/lnc;


(* finding the values of y'k to find
 normalizung constant*)

 y[1] := 1.0 ;

 for i := 2 to lnc + 1 do
    y[i] := (mc[1]/xn[i,1]) * q1[i];


 (* finding the value of beta[n]*)

 n:=1;
 beta[0] := 1;
 for n := 1 to lnt do
   begin
     if (n < ci) then
       begin
         beta[n] := fact(n);
       end
      else

         begin
            beta[n] := fact(ci) * exp((n-ci)*ln(ci));
         end;
   end;


 for n:= 1 to lnt  do
   yy[1,n] := 1/beta[n];

 for i := 1 to lnc + 1 do
   yy[i,0] := 1;


 (* finding the value of the normalizing constant *)


 for  i := 2 to lnc + 1 do
    begin
```

```
     for n := 1 to lnt   do
        begin
            sum:= 0;
           for j:= 0 to n do
              begin
                yys[i,j] := ((exp(j*ln(y[i])))/aa[i,j])*yy[i-1,n-j];
                sum := sum + yys[i,j];
              end;
           yy[i,n] := sum;
        end;
  end;


 writeln(qq1);
 writeln(qq1,'   ','No of terminals','    ','utilization of the processor');
 writeln(qq1,'   ','***************','    ','****************************');
 writeln(qq1);


 (* finding the utilization of the protocol processor*)

 n := 1;
 for n := 1 to lnt  do
    begin
       ue[n] := (yy[lnc+1,n-1]/yy[lnc+1,n]);
       writeln(qq1,'   ',n:3,'      ',ue[n]:8:4);
    end;


(* finding the utilizations of
  different controllers.*)

  n := 1;
  i := 2;

  for i := 2 to lnc + 1  do
     begin
       for n := 1 to lnt do
         begin
              uc[i,n] := ue[n] * y[i];
              write(qq);
              write(qq,uc[i,n]:8:4);
          end;
        writeln(qq);
    end;


(* Finding the throughput and network delay *)

 writeln(qq1);
 writeln(qq1);
writeln(qq1,' no of terminals   throughput    network delay');
writeln(qq1,' ***************   **********    *************');


 for n := 1 to lnt do
      begin
        x[n] :=mc[1]*ue[n]*q1[1];
        r[n] := (n/x[n]);
        writeln(qq1,'    ',n:3,'       ',x[n]:8:4,'     ',r[n]:8:4);
      end;
end.
```

```
program mchain(input,output);

(* Solution of multi-chain queing network*)

var t11:ARRAY[0..20,0..20] OF real;
    n11:ARRAY[0..20] OF real;
    lamda1:ARRAY[0..20,0..20] OF real;
    t12:ARRAY[0..20,0..20] OF real;
    n12:ARRAY[0..20] OF real;
    t21:ARRAY[0..20,0..20] OF real;
    n21:ARRAY[0..20] OF real;
    lamda2:ARRAY[0..20,0..20] OF real;
    t22:ARRAY[0..20,0..20] OF real;
    n22:ARRAY[0..20] OF real;
    t1:ARRAY[0..20,1..20] OF real;
    t2:ARRAY[0..20,1..20] OF real;
    tow11,tow21,tow12,tow22:real;
    w1,w2 :integer;
    lnc,i,n,j:integer;
    xx:text;
    pc,td1,td2:real;

 (* Definition of variables

 t11 : message delay of processor in chain 1.

 n11: queue lenth of processor in chain 1

 lamda1: throughput of chain 1

 t12: delay of controller for chain 1 message

 n12: queue length of controller for chain 1 message

 t21: message delay of processor in chain 2

 n21: queue length of processor in chain 2

 lamda2: throughput of chain 2

 t22: delay of controller for chain 2 message

 n22: queue length of controller for chain 2 message

 tow11: average service time of processor in chain 1

 tow21: average service time of processor in chain 2

 tow12 : average service time of controller for chain 1 message

 tow22: average service time of controller for chain 2 message
w1 : no of terminals in chain 1

w2 : no of terminal in chain 2

lnc: no of controllers

td1 : terminal delay of chain 1.

td2 : terminal delay of chain 2.*)


begin
    assign(xx,'mc');
    rewrite(xx);

    writeln('enter the value of the no of population of chain 1 :');
    readln(w1);
    writeln('enter the value of the number of population of chain 2:');
    readln(w2);
    writeln('enter the value of the number of controllers');
    readln(lnc);
    writeln(xx,'   no of controllers = ',lnc:2);
    writeln(xx,'  no of population in chain 1 =',w1:2);
    writeln(xx,'  no of population in chain 2 =',w2:2);
    writeln(xx);
    writeln(xx);
    writeln(xx,'           STATISTICS OF CHAIN 1');
    writeln(xx);
    writeln(xx,'w1',' ','w2','      ','res1','     ','thput1','     ','res2',
    '    ','thput2');
    writeln(xx);

    td1 := 0.006;

    td2 :=0.004;
    tow11 := 0.002;
    tow21 := 0.003;
    tow12 := 0.005;
    tow22 := 0.005;

    n11[0] := 0;
    n12[0] := 0;
    n22[0] := 0;
    n21[0] := 0;


    pc := 1/lnc;

    for i := 1 to w1 do
      begin
      for j := 1 to w2 do
         begin
            t11[i,j] := tow11 * (1 + n11[i-1]);
            t12[i,j] := tow12 * (1 + n12[i-1] + n22[j]);
          t1[i,j] := t11[i,j] + t12[i,j] + td1;
            lamda1[i,j] := i/t1[i,j];
            n11[i] := lamda1[i,j] * t11[i,j];
            n12[i] := lamda1[i,j] * t12[i,j] * pc;
            t21[i,j] := tow21 * (1 + n21[j-1]);
            t22[i,j] := tow22 * (1 + n12[i] + n22[j-1]);
            t2[i,j] := t21[i,j] + t22[i,j] + td2;
            lamda2[i,j] := j/t2[i,j];
            n21[j] := lamda2[i,j] * t21[i,j];
            n22[j] := lamda2[i,j] * t22[i,j] * pc;
            writeln(xx,' ',i:2,'  ',j:2,'   ',t1[i,j]:8:4,
            '   ',lamda1[i,j]:8:4,' ',t2[i,j]:8:4,' ',lamda2[i,j]:8:4);
          end;
          writeln(xx);
      end;
end.
```