New Jersey Institute of Technology

Digital Commons @ NJIT

Theses

Electronic Theses and Dissertations

Spring 5-31-1990

Design and analysis of movable boundary allocation protocol

Karun Sekhri

New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/theses

Part of the Electrical and Electronics Commons

Recommended Citation

Sekhri, Karun, "Design and analysis of movable boundary allocation protocol" (1990). *Theses*. 1337.
https://digitalcommons.njit.edu/theses/1337

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Theses by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

DESIGN AND ANALYSIS OF MOVABLE BOUNDARY ALLOCATION PROTOCOL

Karun Sekhri, MSEE, New Jersey Institute of Technology
Thesis advisor: Dr I. Wang

The increasing digital communications traffic will require very high speed networks. The use of high communication speed increases the ratio between end to end propagation delay and the packet transmission time. This increase causes rapid performance deterioration and restricts the utilization of the high system bandwidth in broadcast channel based systems. Using several parallel channels in place of a single channel improves this ratio. For a given system bandwidth the total system capacity is increased by bandwidth division and parallel communication. FTDMA protocols have been suggested for the parallel channel network and these protocols are suitable for different loads. In this thesis, the movable boundary allocation protocol has been suggested for the parallel communication architecture. This protocol is suitable for varying loads and yields a better throughput versus delay characteristics. The analysis demonstrates the potential for improvement in the system capacity and the average message delay when compared to conventional single channel system.

DESIGN AND ANALYSIS OF MOVABLE BOUNDARY ALLOCATION PROTOCOL

By

Karun Sekhri

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

Approval Sheet

Title of Thesis: **Design and Analysis of Movable Boundary Allocation Protocol**

Name of Candidate: Karun Sekhri
Master of Science in Electrical Engineering, 1990

Thesis & Abstract Approved by the Examining Committee:

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

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

Dr Meng - Chu Zhou — Date
Assistant Professor
Department of Electrical and Computer Engineering

New Jersey Institute of Technology

VITA

Name: Karun Sekhri

Degree and date to be conferred: MSEE, 1990

Secondary Education: DAV College, Chandigarh, India

Bachelor degree: Punjab Engineering College,
Chandigarh, India. 1984-1988

Major: Electrical Engineering

CONTENTS

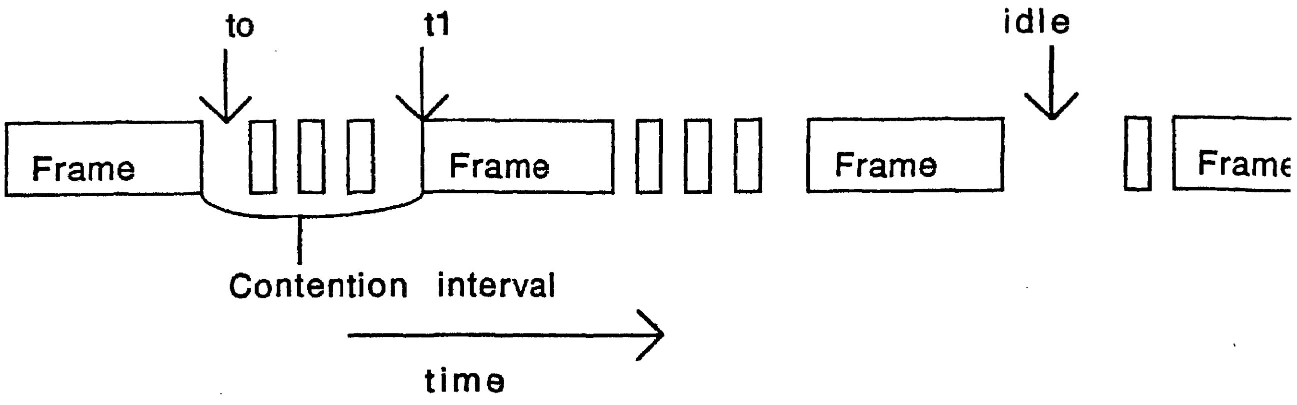
CHAPTER 1

Introduction

As we come across the increasing digital communications traffic such as packetized voice, internet traffic, graphics and video messaging, the need for very high networks will be indispensible. The bandwidth requirement for such may range from several hundred MHz to a few GHz. The fastest of the networks available today have speed of less than 100 MHz which will be very insufficient. The presently availablenetworks which are commonly used are based on CSMA/CD, Token bus,Token ring. The high speed networks available on fiber are FDDI, S/NET, FASTNET, EXPRESSNET and Datakit. The speed for these networks varies from 4 Mbps to 100 Mbps. There are problems associated with the networks if we increase the speed on these networks.

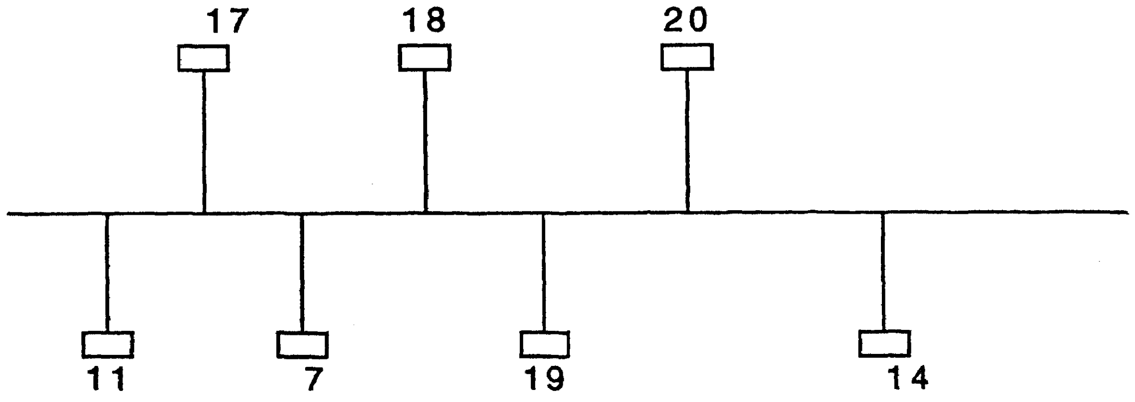
Multiple access communications systems require control information of some type to schedule stations seeking access to the medium. At low speeds, the channel capacity required by this control information is a small fraction of the overall system capacity. However at high speeds, the scheduling capacity required can swamp the capacity available for transmitting data. . .. . In the following section, some of commonly used protocols are discussed with their limitations.

1.1.1 CSMA/CD bus: Ethernet employs CSMA/CD in its media access sublayer. In the protocol, stations listen for a carrier. When a station has data to send, it first listens to the channel to see if anyone else is transmitting If the channel is sensed busy, the station waits until it becomes idle. When the station detects an idle channel, it transmits a frame. If a collision occurs, the station aborts transmission, waits for a random period of time and starts all over again. If two stations sense the channel idle and begin transmitting simultaneously they will both detect the collision almost immediately rather than finish transmitting their frames which are irretrievably garbled.



CSMA / CD frame transmission states

fig (1.0)



A token bus

fig 2

CSMA/CD uses the conceptual model of Fig. 1.0. At the point marked t0, a station has finished transmitting its frame. t1 - t0 is the contention interval. The length of contention period is an important factor in the calculation of efficiency. The minimum time to detect the collision is the time it takes the signal to propogate from one station to the other.

Let the time for a signal to propagate between the two farthest stations be τ. At t_0, one station begins transmitting. CSMA/CD uses the conceptual model of Fig. 1.0. At the point marked t_1, a station has finished transmitting its frame. At τ - ε, an instant before the signal arrives at the most distant station, that station begins transmitting. Of course, it detects the collision almost instantly and stops, but the little noise burst caused by collision does not get back to the original station until time 2τ - ε. In other words, in the worst case a station cannot be sure that it has seized the channel until it has transmitted for 2τ without hearing a collision. For this reason, the contention interval has a slot width of 2τ . If each station transmits during a contention slot with probability p, then the probability A, that some station acquires the medium (ether) during the slot is

$$A=Kp(1-p)^{K-1}$$

where K is the number of stations always ready to transmit on the network.

Probability that contention interval has exactly j slots in it is A $(1-A)^{j-1}$, so the mean number of slots per contention is given by

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1}=\frac{1}{A}$$

Since each slot has a duration of 2 τ , the mean contention interval is $w=2\times\frac{\tau}{a}$. If the mean frame takes P sec to transmit, when many stations have frames to send, then Channel efficiency =

$$\frac{P}{P+2\frac{\tau}{a}}$$

Typical values forτ are 5us for 1 Km long coaxial cable. There are four sources of inefficiency in the CSMA/CD protocol: collisions, sequencing or packet delay, synchronization delay and packet overhead. By far, the most significant factor is the loss due to collisions.

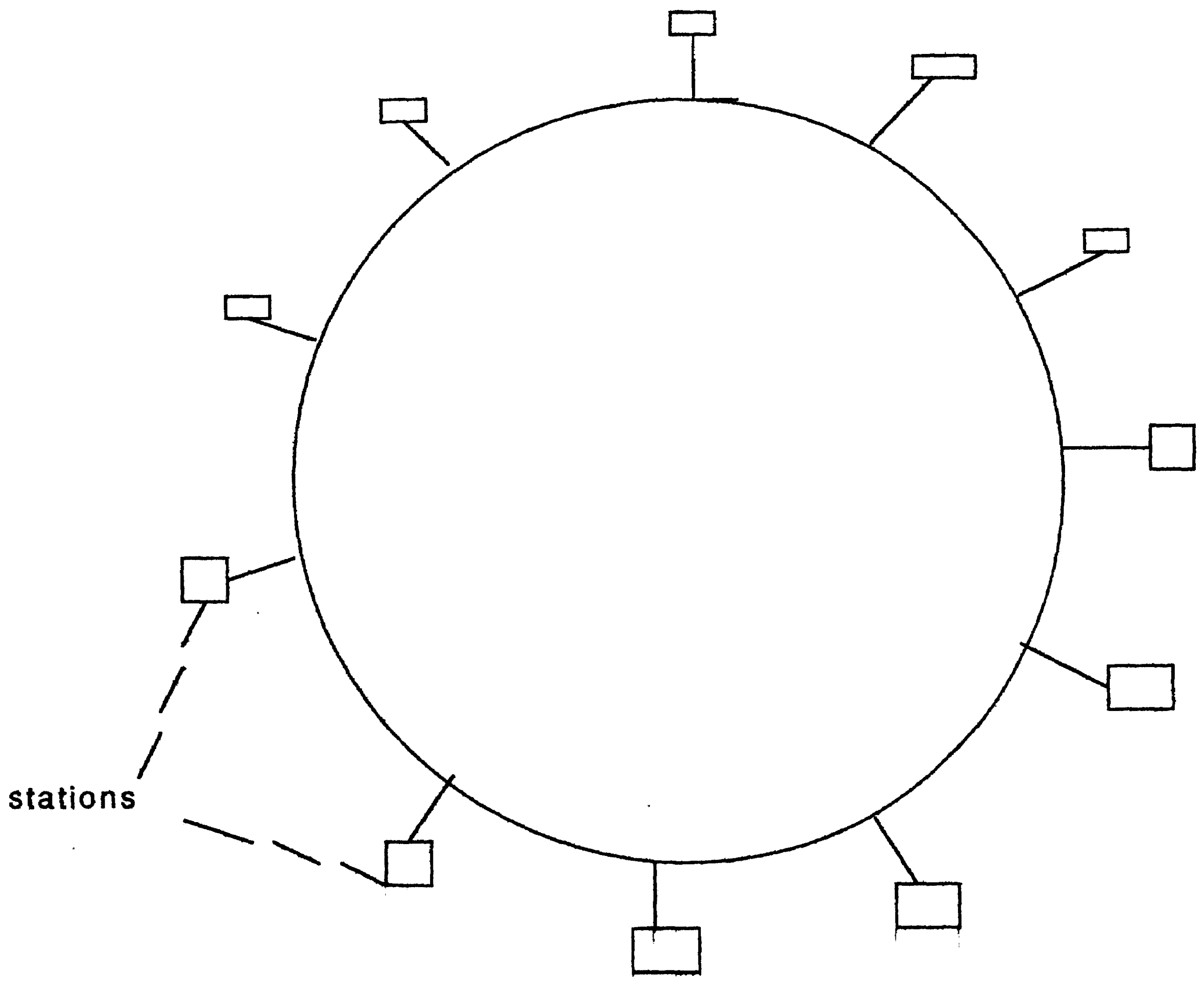
It has been shown that under the conditions of continuously queued sources and an ideal rescheduling algorithm, on the average, an interval of approximately 1.5T is required to reschedule the messages($T = 2\tau$ is the round trip propagation delay).
If we assume a high speed network with packet duration of 0.1T, the capacity of CSMA/CD networks is

$$C=\frac{0.1T}{T+1.5T}=0.04 erlangs$$

The minimum size for CSMA/CD to operate is T. Since packet duration is 0.1T, the packet will only be 10% full. The utilization of CSMA/CD is 0.04 erlangs which is very less and thus CSMA/CD cannot be operated at high speeds. If the signalling speed is increased, the capacity or utilization is furthur decreased.

1.1.2 Token Bus: Physically, the token bus is a linear or tree shaped cable into which the stations are attached. Logically the stations are organized into a ring with each station knowing the address of the station to its left and right. When the logical ring is initialized, the highest numbered station may send the first frame. After it is done, it passes permission to its immediate neighbour by sending the neighbour a special control frame called a token. The token propogates around the logical ring with only the token holder being permitted to transmit frames.

Since only one station at a time holds the token, collisions do not occur. An important point to realize is that the physical order in which the stations are connected to the cable is not important. Since the cable is inherently a broadcast medium, each station station receives each frame, discarding those not addressed to it. When a station passes the token, it sends a token frame specifically addressed to its logical neighbour in the ring.



A TOKEN RING NETWORK

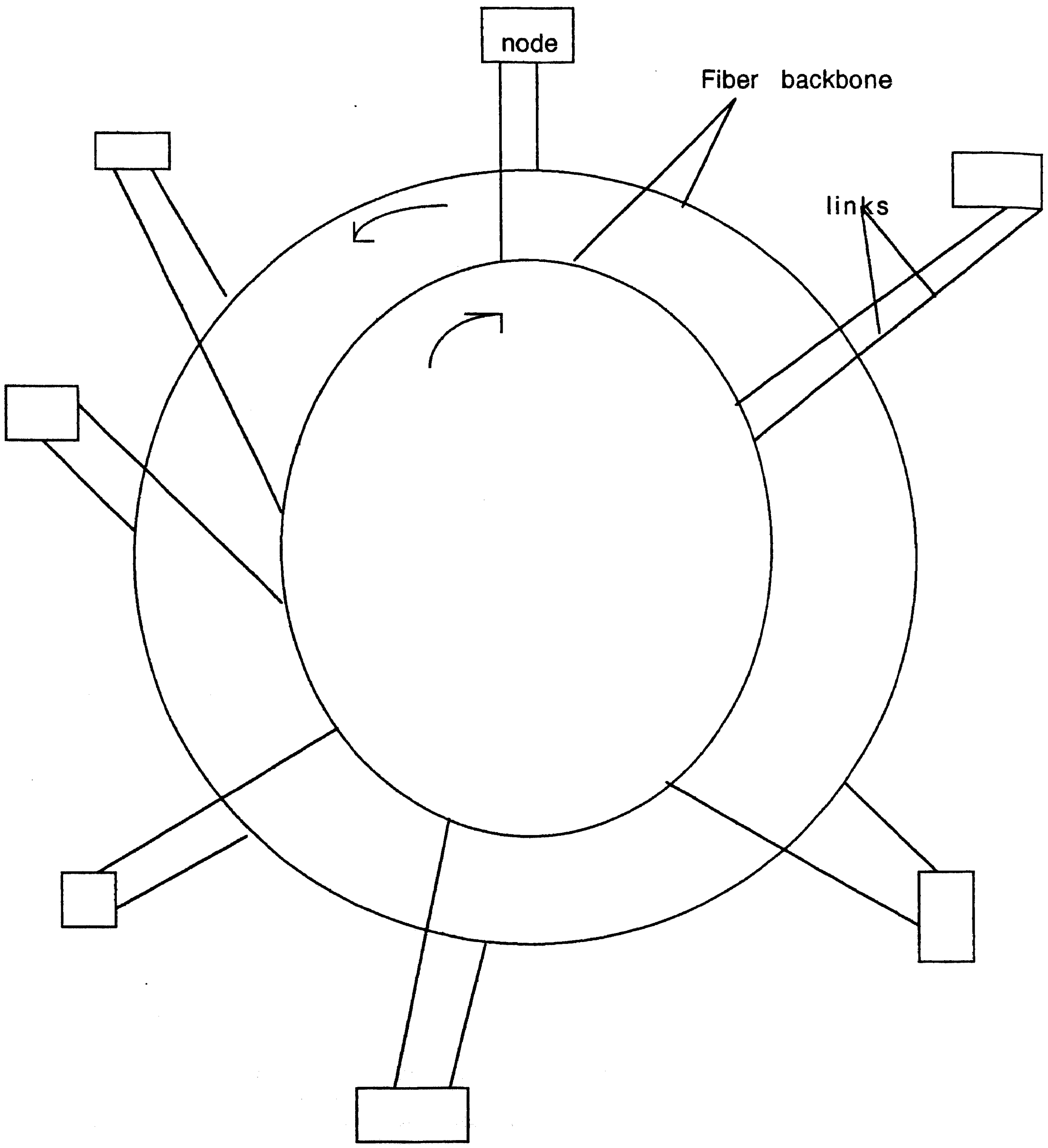
fig 3

The above architecture and the MAC sublayer protocol limits the data rate in token bus to 10 Mbps. Token bus are available with speeds of 1, 5 and 10 Mbps. Let us assume that a token list has been generated that randomly services the stations on the bus. There will be no inefficiency due to collisions but there will be sequencing delay associated with passing token to the next station on the list.

1.1.3 Token Ring: A ring consists of point to point links.The fig(3) shows a token ring network. A major issue in the design and analysis of any token ring is the physical length of a bit. If the data rate of the ring is R Mbps, a bit is emitted every 1/R usec. With a typical signal propogation speed about 200 m/usec, each bit occupies 200/R meters on the ring. This means, for example, that a 1 Mbps ring whose circumference is 1000 m can contain only 5 bits on it at once.

A ring really consists of a collection of ring interfaces connected by point to point lines. Each bit arriving at an interface is copied into a one bit buffer and then copied out on the ring again. In a token ring, a special bit pattern called the token circulates around the ring whenever all stations are idle. When a station wants to transmit a frame, it is required to seize the token and remove it from the ring before transmitting. An implication of the token ring design is that the ring itself must have a sufficient delay to contain a complete token to circulate when all stations are idle. The delay has two components; the 1-bit delay introduced by each station and the signal propagation delay. On a short ring, an artificial delay may have to be inserted into the ring to insure that a token can be contained on it.

When the traffic is light, the token will spend most of its time idly circulating around the ring. Occasionally a station will seize it, transmit a frame, and then output a new token. However when the traffic is heavy, so that there is a queue at each station, as soon as a station finishes its transmission and regenerates the token, the next station down stream will see and remove the token. In this manner, the permission to send rotates smoothly around the ring in round robin fashion. The



FDDI NETWORK

fig 4

network efficiency can approach 100 percent under the conditions of heavy load. But there are limitations like delay to increasing the capacity in a token ring.

1.2.0 High Speed Networks: Most of the high speed networks available today are fiber optic based. This is because the fiber has a high bandwidth, is thin and light-weight, is not affected by power surges and has excellent security. In the following section, we discuss some of the fiber optics based high speed networks.

1.2.1 FDDI: Fiber Distributed Data Interface:- This is a high performance fiber optic token ring LAN running at 100 Mbps over distances upto 200 Kms with upto 1000 stations connected. The FDDI design specification calls for no more than 1 error in 2.5×10^{10} bits. The FDDI cabling consists of two fiber rings, one transmitting clock-wise and the other transmitting counterclockwise. If either one breaks, the other can be used as a backup.

The basic FDDI protocols are closely modeled on the 802.5 protocols. However one difference between FDDI and 802.5 is that a station may not generate a new token until its frame has gone all the way around and back. In FDDI, with potentially 1000 stations and 200 km of fiber, the amount of time wasted for the frame to circumnavigate around the ring could be substantial. For this reason, it was decided to allow a station to put a new token back into the ring as soon as it is done with transmitting its frames. In a large ring, several frames may be on a ring at the same time.

1.2.2 Fibernet II : This is a fiber optic LAN compatible with Ethernet at the transceiver interface so that the stations could be plugged into using their existing station transceiver cable. The hard part about building any CSMA/CD network out of fiber optics is getting the collision detection to work. Methods like power sensing, pulse width, time delay and directional coupling are possible but these are quite tricky to implement. All these methods use passive stars that greatly weaken the signal because the incoming energy has to be divided over all the outgoing lines.

1.2.3 S/NET : This is another fiber optic network with an active star for switching. The goal of this is to provide fast switching. Each computer in the network has 20 Mbps fibers running to the switch, one for input and one for output. The fibers terminate in a BIB (Bus Interface Board). The CPUs each have an I/O device register that acts like a one-word window into BIB memory. When a word is written to that device register, the interface board in the CPU transmits the bits serially over the fiber to the BIB, where they are reassembled as a word in BIB memory. When the whole frame to be transmitted has been copied to BIB memory, the CPU writes a command to another I/O device register to cause the switch to copy the frame to the memory of destination BIB and interrupt the destination CPU.

1.2.4 FASTNET and EXPRESSNET : Fastnet uses two linear unidirectional buses, as shown in Fig (5). Each station taps into both buses and can send and receive on either one.

When a station wants to send a frame to a higher numbered station it transmits on bus A; when it wants to send to a lowered station, it transmits on bus B. Stations 1 and N play special roles in this network. A transmission cycle is started when station 1 begins transmitting a sequence of fixed-sized slots on bus B. These slots provide the clocking for the bus. Other stations synchronize their transmissions to them as they propagate by. The slots can be thought of as a train of empty flatcars onto which data can be loaded.

When a station wanting to transmit to a higher numbered station detects the start of the train of bus A, it waits until the first empty slot passes by. The station then sets a bit in the first byte of the slot marking it as busy, and places the source and destination addresses and the data in the empty slot. If the data do not fit in a single slot, several consecutive slots may be allocated. When the downstream to whom the frame is addressed sees the frame, it just copies it to its memory, leaving the slots on the bus, still marked as busy. An analogous mechanism is used on bus B.

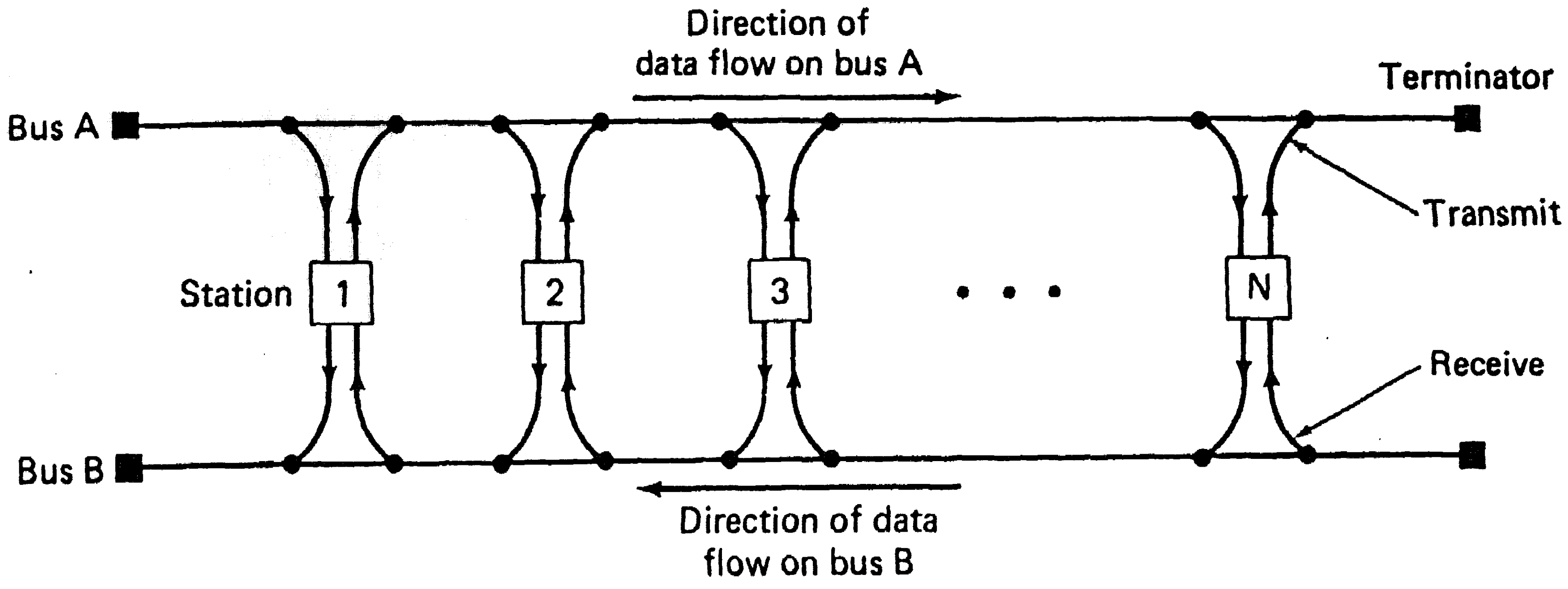

Direction of data flow on bus A
Terminator
Bus A
Transmit
Station
1
2
3
• • •
N
Receive
Bus B
Direction of data flow on bus B

Fig (5)

One of the major problem associated with fiber optics networks is taps. For adding a new node to the network, the backbone has to be tapped and it may cause loss of optical signal to a considerable db level. Moreover the fiber optic hardware is expensive and a lot more improvement has to made in the interfaces.

The parallel channel network architecture has been proposed which can use the existing interfaces and provide greater efficiency. This architecture employs a new set of protocols for its media access control sub layer. It is for this architecture that the movable boundary allocation protocol has been proposed. The following chapters discuss it in detail.

CHAPTER 2

Architecture and Modelling of the high speed network

As already mentioned in Chapter 1, in communication systems based on a single channel, the increase in channel bandwidth can only be partially utilized. In these systems, as explained, as the bandwidth is increased, the packet transmission time becomes small relative to the time required for the packet to propogate across the network and to the node processing time. Due to these factors and due to unavailability of suitable high speed interfaces, introduction of high speed channels into these systems is accompanied by only insignificant increase in the capacity (speed) of these networks.

2.1 Network Architecture: The single high speed channel is divided into a number of subchannels of lower speeds. Since the speed is less on lower speed channels, the packet transmission time increases on these subchannels. However the end to end propogation delay remains the same. Thus the packet transmission time T_p increases relative to the propogation delay T_c. End to end propogation + node processing time is a measure of control activity or time. If Tp is the packet transmission time and Tc is the end to end propogation delay.

$$\eta = \frac{T_p}{(T_p + T_c)}$$

Since Tc is reduced, η is increased and this leads to a better utilization and higher overall bandwidth for the network. The channel is divided into sub channels by frequency dividing the larger channel. If the single channel has a bandwidth of say 1 GHz, we can divide the channel into say n subchannels each of frequency 1000/n MHz. It is possible to provide frequency bands using modulation techniques. Each of

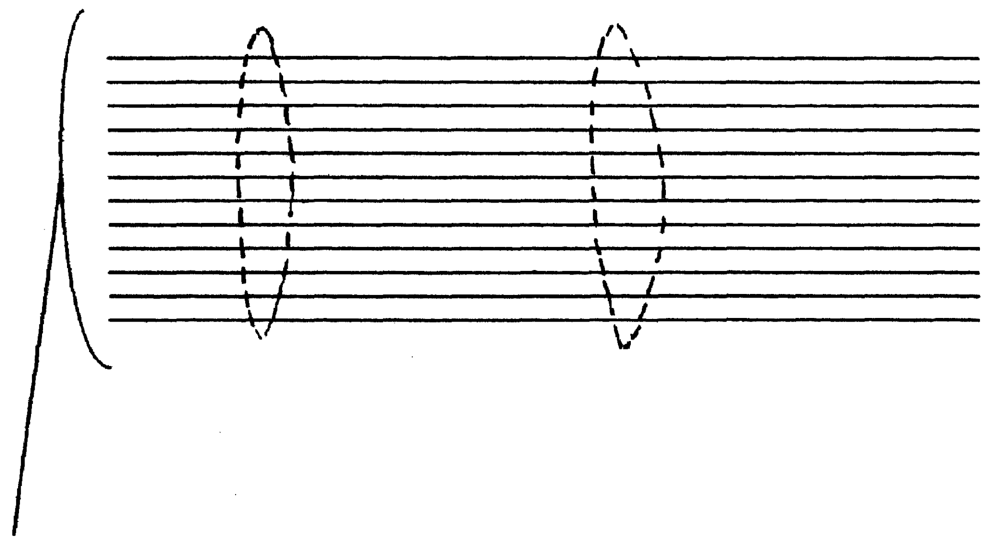


Fig 6

these bands have different frequency range but same bandwidth. It is possible to use physically disjoint channels but these lead to an increase in the number of taps.

Now the question that arises is that how many sub channels should be there. That an optimal network design exists is easy to see. If the no. of sub channels is very large, then inspite of improving packet transmission time to the end to end propogation delay, most channels will be unused at low to medium loads. Thus the remaining sub channels will work only at fraction of the original speed. If the number of sub channels is small, the network will behave more like singlechannel network and the advantages of the parallel network diminish. But even at lower loads, the probability of channels being unused is very small and utilization will be higher. Thus there has to be a compromise between these two factors:

a) the ratio between the packet transmission time to propogation delay

b) effective utilization of channels at low to medium loads.

Another design issue is the functional design of the node's channel interface. The nodes should be connected to all the subchannels in order to preserve the advantages of the broadcast channel communication i.e. more than one transmission can be directed to a node on different subchannels. Thus we must determine the number of subchannels on which each node is able to receive simultaneously. If the channel interface is such that node can receive on all channels simultaneously, the interface becomes very expensive and cannot be afforded to use in practice. Thus the balance for the interface reception design should be between minimum number of functionality duplication (i.e. minimum number of channel on which the reception is possible) and the network performance. Currently some of the available networks like hyperchannel use parallel communication channels. Multiple-channel adapters and frequency agile modems are available. Analysis of replicated channel ALOHA systems [5] , [6] and CSMA systems [9] have shown that significant performance improvement and a number of implementation advantages can be obtained by the use of multiple channel architectures. These approaches have considered the capacity increase due to additional bandwidth, available by increasing the number of channels simultaneously. However, assuming a morerealistic bounded reception capability, it has been demonstrated [10] that additional improvement obtained by channel multiplication can only be partially utilized.

2.2 Network Model:

The network consists of N nodes and B is the total capacity. We divide the channel bandwidth B bits/sec in subchannels i.e. each node can transmit on any of the b subchannels simultaneously. A node j can receive packets simultaneously on Kj sub channels simultaneously. $1 < b < Kj$. In the model we assume that the packets are of constant length in bits. Let every packet has x bits in it. Then To, the packet transmission time is given as

$$To = \frac{x}{B}$$

Let a be the normalized propogation delay with respect to To. IF z is the propogation delay, then

$$a = \frac{z}{To}$$

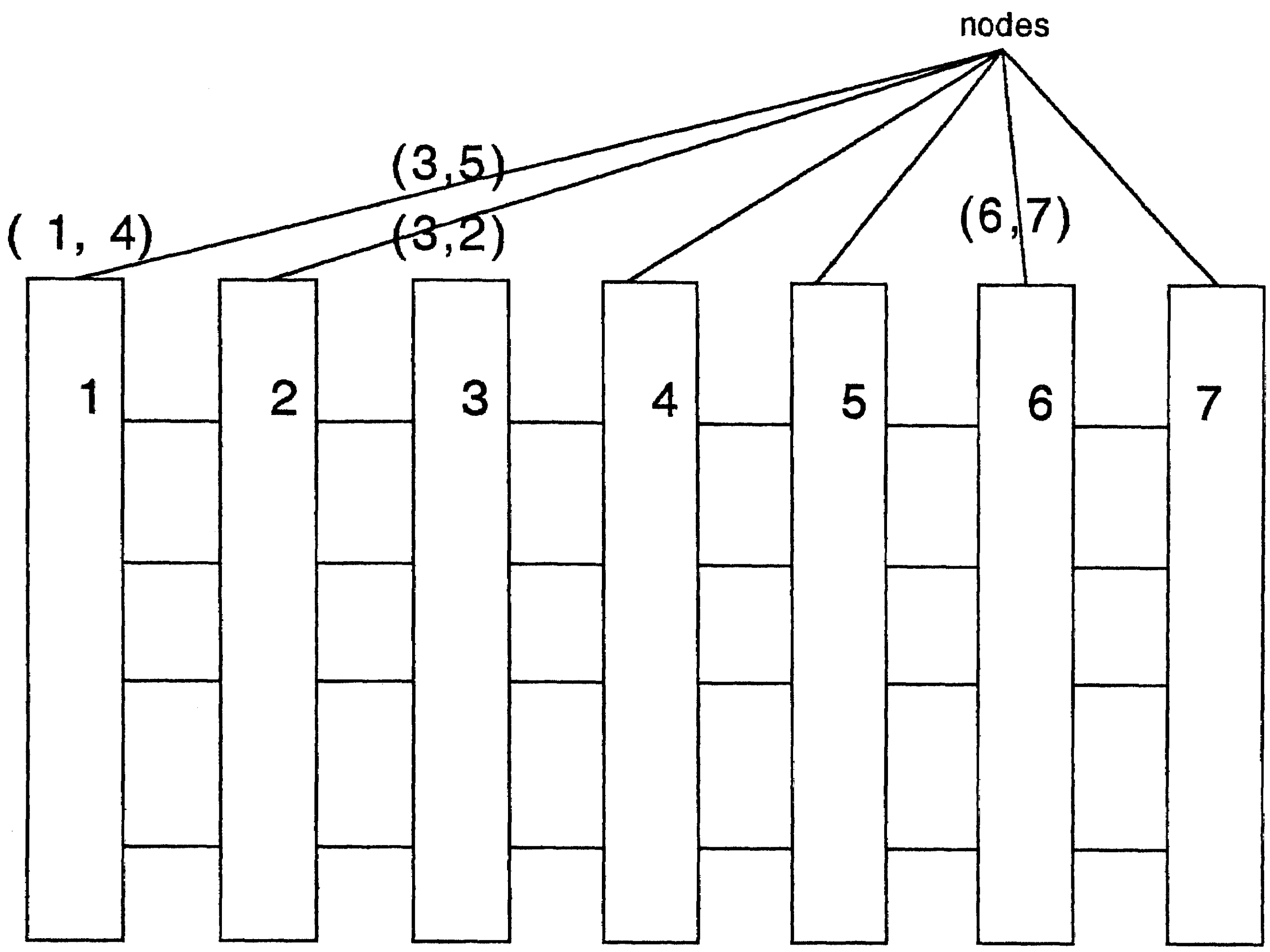
Let T_b be the time required for the packet transmission on the subchannel. The packet transmission time normalized with respect to To given by

$$b = \frac{T_b}{To}$$

We define the slot duration s of the system normalized with respect to To as

$$s = (b + a).$$

In the slot duration, the factor a represents the time for control activity. The control activity includes time for deciding how many permissions will be given in the slot, on which channel the packets are to be transmitted and other network information. We assume that a maximum period of whole propogation delay a is allowed to drain the packet from the system before next transmission.



The figure shows a 7 node network with the single channel sub divided into into 4 sub channels. (X,Y) represents a virtual pair which implies that X has a packet to send to Y. . There can be at most 7 X 6 = 42 such virtual pairs for this network.

To make the point clear, let us assume that the bandwidth is say 1 GHz. Let b be 100. The packet say consists of 1000 bits. Then the packet transmission time on the undivided channel is

$$To = \frac{1000}{10^9} \text{ sec} = 1\ \mu \text{ sec}$$

Now $b = \frac{T_b}{To}$

$$=> T_b = 100\ \mu \text{ sec}$$

Thus the transmission time on a sub channel is increased 100 times. However since there are 100 sub channels, at the most 100 packets can be transmitted in a slot. Thus 100 packets will require 100 usec to transmit on the parallel network architecture. However the packet transmission time is increased on a single sub channel and this largely improves the propogation delay to the packet transmission time ratio and thus leads to a much better utilization of the network.

To model the packet generation process, we assume that each node has one arrival buffer. Packet arrival rate is assumed possion. Let the mean arrival rate for an idle node i be λ_i packets/To or λ'_i packets per slot time where λ'_i = λ_i (b+a).

z is the interarrival time for poisson process. For poisson statistics, it turns out that z is an exponentially distributed random variable i.e. its probability distribution function Fz(x) is given by

$F_z(x) = 1 - e^{\lambda'_i}$.

Therefore the node packet generation is a Bernoulli process with rate $1 - e^{\lambda'_i}$ per slot.

The probability that a packet at a node i is destined for node j is q_{ij}. We formulate a closed queuing model in which generation of packets occur only when the node's transmission buffer is empty. This model has been shown to be more appropriate for communications activity of the interconnection nodes and of very high speed networks.

For traffic modelling, we assume that every node i has N conceptual buffers where N-1 transmission buffers are dedicated each to a different destination j, $1 < j < N$, $j != i$ and one buffer is dedicated to the reception of messages. Each transmission buffer will represent a virtual user (i,j) $1 < i, j < N$ or service point in ISO terminology. In this way, for the interconnection between each pair of nodes, peer protocols can be modelled independently.

Network Throughput:- The throughput of the network was given by Imrich Chalamtac and Ora Ganz [1] and given by the expression

$$S=\frac{1}{(b+a)}.(N-\sum_{i}\pi)(1-e^{\lambda_i}) \qquad (2.2)$$

where π represents the probability that at the beginning of a slot there are i packets in the system.

Now let us evaluate the effect of channel partitioning on the system bandwidth utilization. We first compute the maximum throughput S_{max}(b) defined as the system throughput obtained for a given b under heavy load. For heavy load, $\lambda \rightarrow \infty$ and therefore

$$1-e^{\lambda_i}=1$$

For maximum throughput, assume that no destination conflicts and channel collisions occur. For each transmission of duration b, we occupy the channel for a slot duration b + a. Thus the maximum system throughput is given by

$$\text{Smax(b)} = b/b + a$$

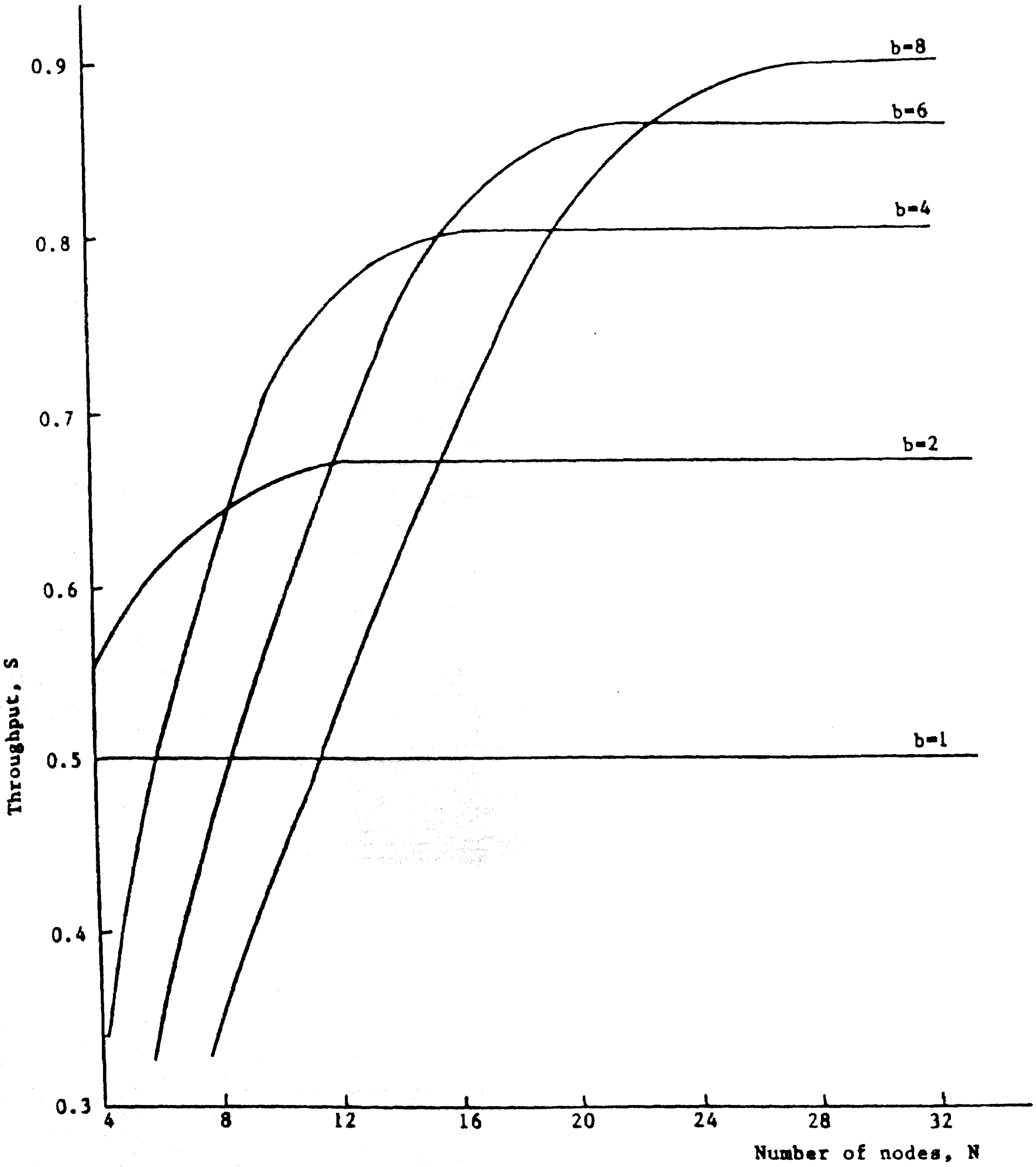
$$=\frac{1}{1+\frac{a}{b}} \qquad \text{---(2.3)}$$

We define the system capacity C as the max(b) Smax(b) obtained over all possible values of channel partitioning factor b. Clearly, the capacity is obtained when the no. of sub channels equals the no. of packets in the system at the begining of each slot since if the no. of packets is more, the collisions will occur and if the no. is less than b, the full capacity will not be utilized. Since in the considered model, we choose a constant value of b, the maximum throughput will be obtained when also the no. of packets at the begining of each slot remains the same. Given the closed queuing model in which packet arrival occurs only to the empty arrival buffers and the preceding observations,(i.e. no. of packets at the slot begining is constant) the capacity is obtained when the number of active nodes equals the number of blocked nodes.

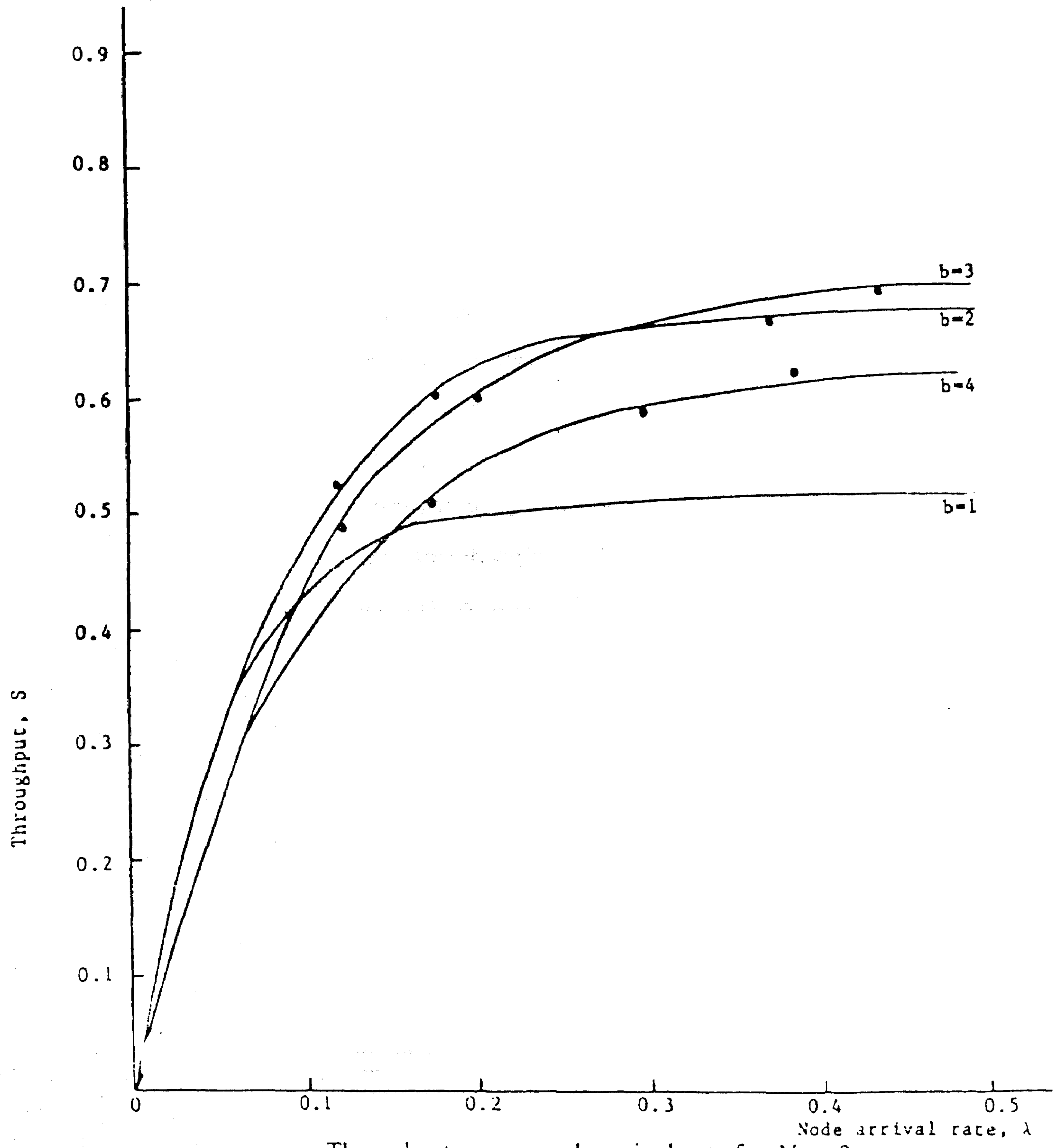
In other words, the number of nodes having a packet to transmit be equal to the nodes already transmitting. This number equals to N / 2 i.e. the number of partitioned sub channels is equal to N - half the total number of nodes in the network. The substitution of b=N/2 (optimized) into the above equation 2.3 produces the system capacity $C = 1 / (1 + a/N/2)$.

By partitioning the bandwidth b>1, the improvement in capacity can be obtained for every N as seen from the fig(2.1). The capacity improves as N increases since larger N' allow the use of a higher partitioning factor b = N/2. This leads to a larger number of subchannels which in turn increases the Tb = bTo, the packet transmission time on the sub channel. Thus the ratio between packet transmission time to the propogation delay increases leading to a better utilization of the total system bandwidth.

At lower loads $p = 1 - e^{-\lambda} < 1$, and from eq (2.1) for throughput it is apparent that the throughput will decrease. All of the N/2 subchannels will not be fully utilized and some will be vacant. Also due to the bandwidth division, these few utilized sub channels work at large packet transmission times and are slower in speed. The above two factors combined together lead to a higher packet delay and lower system throughput. Thus, for lower loads, the best performance is obtained for a partitioning factor b which is strictly less than N/2 so that none or lesser no. of channels are wasted. The result for this case were derived using eq(2.1) for various values of λ. Fig (2.2) shows throughput S as a function of the arrival rate λ packets/sec for N = 8 and a=1, 5. As already mentioned, from the figure it is clear that for different traffic loads, different values of b provide the best performance in terms of throughput. It can be



Throughput versus number of nodes for $p = 1$ and $a = 1$.



Throughput versus node arrival rate for $N = 8$.

furthermore observed that for corresponding values of λ, when the normalized propogation delay a increases from 1 to 5, additional channel partitioning i.e.increasing the number of sub channels leads to a better system throughput. Undoubtedly the overall is more in the case a=1. The preceding observation is consistent with the intuition suggesting that from the consideration of improving the a/b ratio, higher partitioning factor. The fig (2.3) shows the system throughput as a function of the number of nodes for different partitioning factors with a= 1 for very high loads. We observe that throughput is an increasing function of the number of nodes N. This is due to ability to utilize a higher number of sub channels as N increases thus improving the a/b ratio.

The probability of having i packets i>k destined to a single node is a nonlinearly decreasing function of k where k is the maximum number of packets which can be received by a node. It has ben shown that for example, in the given system of N=8, the probability of finding more than three packets with the same destination in one slot is negligible. In fact, for sufficiently high loads, the throughput behaviour approaches the system capacity given by $C = 1/\ 1 + a/(N/2)$ already for $k > 3$. The ability to receive several packets simultaneously also leads to a reduced average packet delay. Again, it can be shown that high throughput values larger k's (k=b) can, due to virtual elimination of destination conflicts significantly reduce the expected packet delay.

2.3 FTDMA protocols:- These protocols are suitable for the network architecture described above. In these protocols, the channel access is governed by a fixed cycle. In each cycle, source/destination oriented permissions are granted, specifying for each node, or virtual user, the channel number and the slots in which to transmit. The protocols in this class differ in the number of permissions granted in each slot and in their channel allocation policy - random or deterministic. The number of permissions per slot and the channel allocation dictate the channel access associated penalities. For these protocols, collision is defined as simultaneous transmission by two or more nodes on a single channel.

We refer a destination conflict as the simultaneous arrival of two or more successful packets on the same destination node. In FTDMA protocols, the loss of bandwidth due to collisions and destination conflicts can be traded against the probability of finding unused channels (idle slots) in a non empty system. When the system is being fully utilized,the probability of finding unused channel is small. But the delay at the nodes increases i.e. users have the right to transmit. Due to the existence of above mentioned tradeoff, it is possible to define FTDMA protocols ranging from fully controlled access (collision/conflict free) to uncontrolled access of the channels.

To formally define the FTDMA protocols, we let T be the size of allocation cycle in slots. Let U(t) be an N*N allocation matrix. Each entry Uij(t) of the matrix will represent the permission right granted to user (i,j) in slot t, t= 1, 2, 3 ...... T 1<i, j<N. i represents the source node and j represents the destination node. Four types of FTDMA protocols have already been suggested for the network architecture. They are

(1) Source/destination allocation

(2) Source allocation

(3) Destination allocation

(4) Allocation free protocols

In the **source/destination allocation protocols,** disjoint sources and destinations are chosen. Since there are b channels, b source-destination pairs are chosen. The pair (i,j) are assigned the channel number. The allocation matrix U(t), 1 < t < T for source destination allocation protocol obeys the following conditions:

a) Sum i Wij(t) < 1 for destination conflict free transmission. b) Sum i Sum j Wij(t) = b for restricting the total no. of permissions in a slot to a number of channels. c) Sumj Wij(t) < 1 for single transmission per source node where Wij is a matrix whose elements are 1 for {U(t)} > 0. By allocating permissions to virtual users, both source and destination nodes are uniquely determined in each transmission.

Thus there are no collisions in this protocol.

Destination allocation protocol:- In this protocol, the total number of permissions in a slot is increased to N. Thus N disjoint pairs are chosen among a population of large no. of disjoint pairs without the specification of channel number. The N virtual users choose randomly the b available channels and in this way, the destination gets allocated but channel collisions occur because two packets may try to occupy the same channel. The allocation matrix in this case is a binary matrix U(t) where a nonzero entry Uij(t) obeys the following conditions:

a) SumiUij(t) < 1 for destination conflict free transmission.

b) SumiSumj Uij(t) = N restricting total number of permissions in a slot to number of destinations.

c) SumjUij(t) < 1 for single transmission per node.

Source allocation protocol: In the source allocation protocol, permissions are granted to b source nodes. Thus in this protocol, b out of b*(N-1) users transmit without collision. In source allocation matrix, the value Uij(t) in a non zero row i is permitted to transmit a message. The total number of non zero entries in U(t) equals b*(N-1) and is greater than in source/destination allocation and destination. The probability of finding an unused channel is therefore smaller, making the protocol attractive for low to medium loads, especially when b < N.

Allocation free protocols:

In this protocol, the time division allocation cycle is collapsed into a single slot and any node with a message ready for transmission can transmit randomly on a channel. In each slot N * (N-1) permissions are granted for transmission on b channels. At the beginning of each slot, any node with a message for transmission is allowed to transmit the message on a randomly chosen channel. The allocation matrix is thus simply represented by a binary matrix U(t) with Uij(t)=1 for all i, j.

Problems with the above protocols:

In source/destination allocation protocols, only b permissions can be given to disjoint virtual pairs. The maximum number of disjoint pairs in the system can be N. At high loads, there is a good probability that sufficient number of disjoint pairs is there. These pairs can be given to only b channels. Thus the number of slots in a cycle increase. Also at low to medium loads, there may be not sufficient number of disjoint pairs and some of the channels will not be utilized.

In destination allocation protocol, the channel collisions will increase significantly in the case $b \ll N$. For $b \ll N$, the number of disjoint pairs may be much more and these disjoint virtual users (i,j) will contend for b channels and collisions result. The collisions will also increase for $b = N/2$ and high loads. In allocation free protocols, any node may be allowed to transmit on a channel randomly. Since there are at most $N(N-1)$ virtual users contending for b channels, the chances of collisions on the channels are very high.

To solve the above problems to some extent, we propose the Movable boundary allocation protocol.

CHAPTER 3

Movable Boundary Allocation Protocol

3.1 MBA PROTOCOL: The movable boundary protocol results from the nature of FTDMA protocols mentioned in Chapter 2. It is possible to formulate these protocols because in the proposed parallel channel architecture, the channel collisions and destination conflicts are a function of system load and configuration. For low to medium loads, the source allocation protocol is prefered and for high loads, the source/destination allocation protocol offers the best performance.

In view of the above considerations, in movable boundary protocol.

a) The disjoint pairs of virtual users n are identified and are given the allocation for n subchannels.

b) The control for the remaining channels ,if any, is given to the remaining nodes. Since each node can transmit to N-1 different nodes, the contention takes place among N-1 buffers at these nodes.

c) If the number of disjoint pairs is more or equal to b, the number of subchannels, the protocol behaves like source/destination allocation protocol. If the number of disjoint pairs is none, the protocol is same as source allocation protocol.

d) The boundary is movable between the source/destination allocation and source allocation characteristics. Variable number of channels depending on the load (disjoint pairs) can be assigned for each of the above characteristics.

3.2 Analysis of the MBA protocol:

To evaluate the performance of movable boundary allocation, we make use of following definitions:

$S_{i,j}$ - the throughput of user (i,j) defined as the number of successful messages of user (i,j) per message transmission time To.

S - the system throughput defined as the total number of successful messages of user (i,j) per message transmission time To.

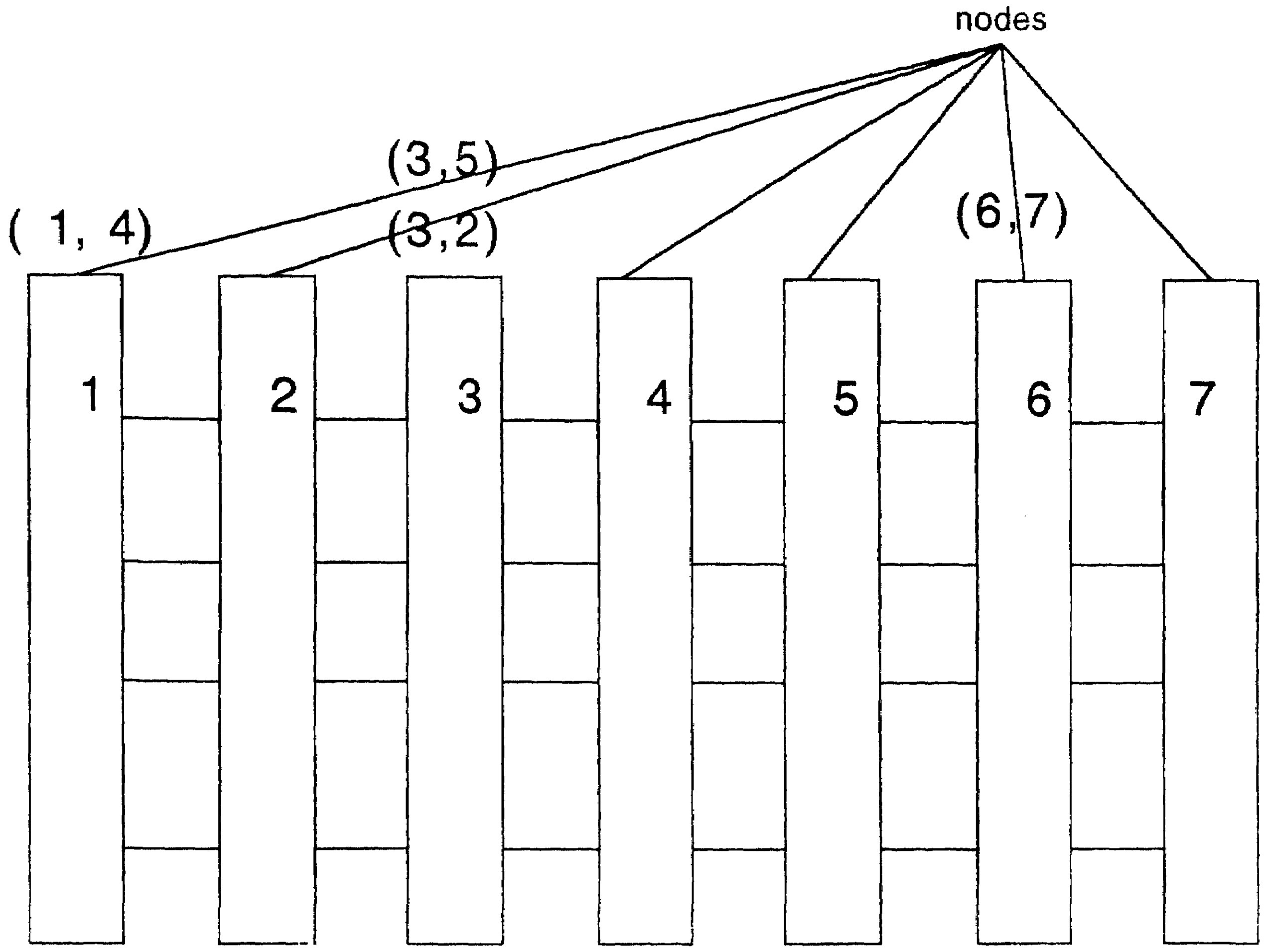
$D_{i,j}$ - the average message delay of user (i,j) defined as the average time between the arrival of a message at (i,j) and the begining of its transmission, normalized to To.

D - the average message delay in the system. The system load G is given by

$$G = \sum_i \sum_j \lambda'_{ij}$$

Under the previous assumptions in chapter 2, we can model the system by a discrete time Markov chain, obtained by observing the state of the system at the beginning of each slot. How much time the system remains in a particular state is arbitrary. Hence the Markov chain is not pure but an embedded Markov chain. At the instants of state transition, the system behaves like an ordinary Markov chain. The embedded Markov chain can be modelled as a two dimensional process [I(t), Xo(t)] where I(t) is the number of customers present at time t and Xo(t) is the service time already received by the customer in service at time t.

To significantly reduce the complexity of analysis, we build approximate models by reducing the amount of information recorded in each state while preserving the Markovian property. The exact description of the source or the destination of all packets in the system for any combinations of arrival rates and any level of channel interface multiplication leads however to a number of states which is exponential in the number of nodes N and equal to N^N. Due to the existence of channel collisions and destination conflicts, the state of the system must record the source/destination distribution of all messages in the system at the embedded points. The embedded points are just at the beginning of each slot. Therefore, in our approximate model, the complexity of solution is transferred from need to correctly model the system behaviour in the face of only partial state information. For the protocol analysis model, we shall assume that the arrival rates of all the users are equal,thus $\lambda = \lambda_{ij}$ and $\lambda' = \lambda'_{ij}$



The figure shows a 7 node network with the single channel sub divided into into 4 sub channels. (X,Y) represents a virtual pair which implies that X has a packet to send to Y. . There can be at most 7 X 6 = 42 such virtual pairs for this network.

In the steady state, the expected number of arrivals must equal the expected number of departures. Now the throughput S was defined as the total number of successful messages per To. Since the total number of successful messages is the expected number of departures, in the steady state, the throughput S must be equal to the expected number of arrivals. Let CL denote the node cycle time defined as the average time elapsing between two consecutive arrivals of a message at a node. Since the arrival rate is assumed poisson, the mean interarrival time is $\frac{1}{\lambda}'$. D is the delay in the system and it is the time between the arrival of a packet and its transmission from the system. Thus the length of cycle CL equals D + $\frac{1}{\lambda}'$. The expected number of arrivals to the whole system per second becomes N'.1/(D + 1/λ') where N' denotes the total number of virtual users in the system and is given by N'=N .(N-1). From this obervation we obtain that

$$\frac{N'}{D+\frac{1}{\lambda}'}=S \tag{3.1}$$

Thus the average waiting time D

$$D=\frac{N'}{S}-\frac{1}{\lambda}' \tag{3.2}$$

Since poisson distribution is assumed, the message arrival forms a bernoulli process with rate (distribution) 1 - $e^{-\lambda'}$, which is the probability of message generation during the time interval T. Without the loss of generality, we let T = 1. Let π_i denote the steady state probability that at the beginning of a slot; there are i messages in the network. From the definition of discrete random variable, the mean value of random variable i is

$$\sum_{i=0}^{N'} i\pi_i \tag{3.3}$$

i.e this is the expected number of messages in the network. Since the closed queuing system is assumed, the message arrival can occur to N'-$\sum_{i=0}^{N'}$ i π_i buffers. The probability of message generated as stated is 1-$e^{-\lambda'}$. Thus the throughput of the network which is equal to the number of arrivals is

$$S=\left[N'-\sum_{i=0}^{N'} i\,\pi_i\right](1-e^{-\lambda'}) \tag{3.4}$$

To compute D and S, we derive the steady state probabilities $\Pi = (\pi_0, \pi_1, \pi_2, \ldots\ldots \pi_{N'})$ which can be obtained from the solution of following equations.

$$\Pi=\Pi P \tag{3.5a}$$

$$\sum_{i=o}^{N'}\pi_i=1 \tag{3.5b}$$

The transition probability p_{ij} is defined as the probability that at the beginning of a slot, the system is in state j given that the system was in state i at the beginning of the previous slot. p_{ij} is given by

$$p_{ij}=\sum_{s=0}^{min(i,b)} Suc(i,s).B_{j-i+s,i}(1-e^{-\lambda'}) \tag{3.6}$$

Suc(i,s) --

This is the probability that s packets reach the destination successfully in a slot time given there were i packets in the system at the beginning of a slot.

$B_{j-i+s,i}$(r) --

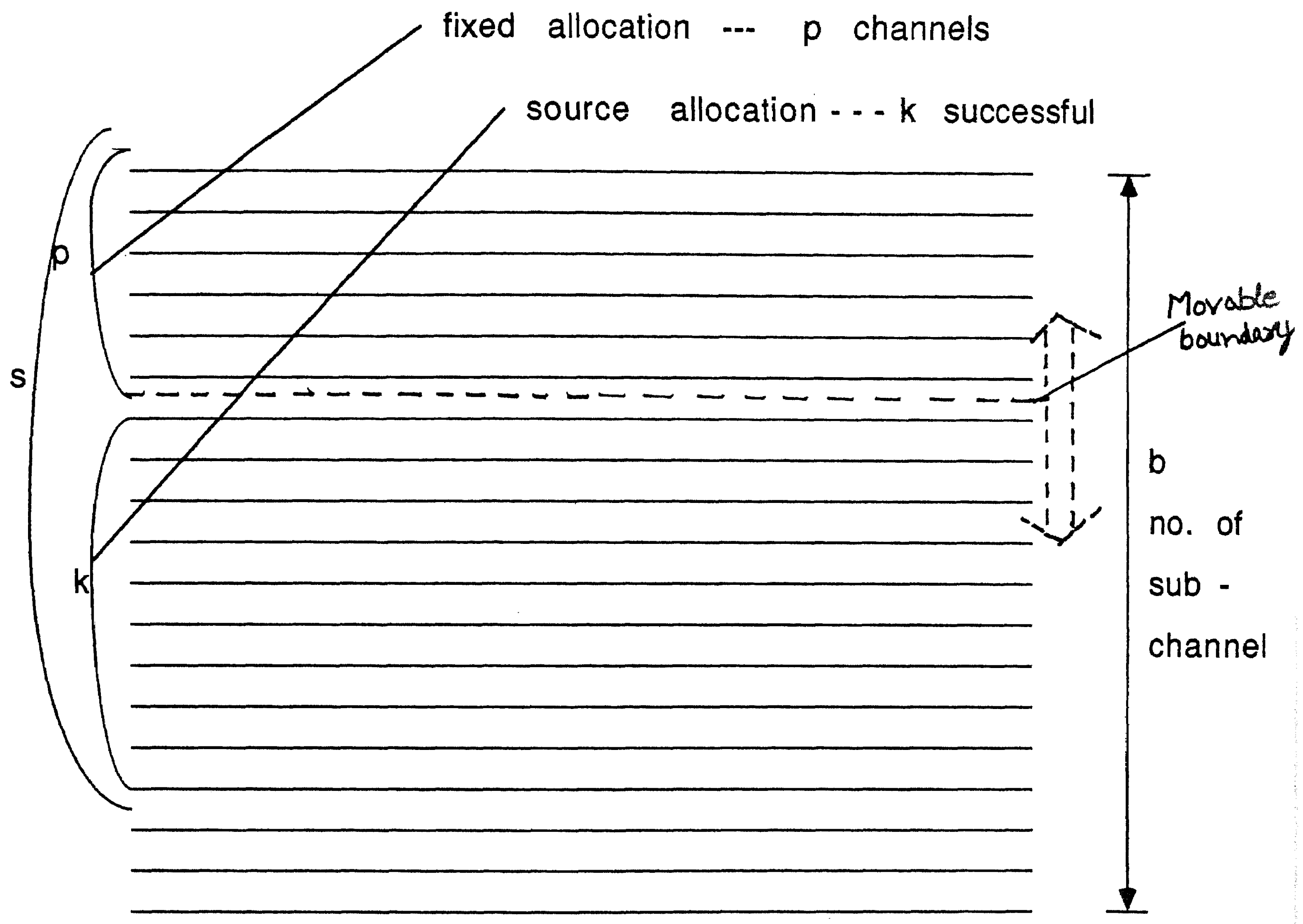
This accounts for the probability that given i packets at the beginning of a slot in a system, j packets at the end of the slot and s packets transmitted successfully during the slot, j-i+s new messages originate from the binomial process with parameter r. Here r is given by 1 - $e^{-\lambda'}$.

$B_{ij}(r)$ has been derived in [4] and can be represented by the following expression :

$$B_{ij}(r) = \binom{N'-j}{i}.r^{i}.(1-r)^{N'-i-j} \tag{3.7}$$

Suc(i,s) depends on the movable boundary allocation protocol For analysis,the protocol can be modelled in the following steps.

1) For p disjoint virtual users, there are p disjoint pairs (i,j). Thus p nodes corresponding to the i in the virtual users are given permission. p varies between


fixed allocation --- p channels
source allocation - - - k successful
p
s
k
Movable boundary
b
no. of
sub -
channel

$s = p + k$

s is the total number of messages which succeeed on s out of b channels.

0 and b where b is the number of channels.

2) p channels are allocated to disjoint pairs.The p virtual users are transmitted collision free with probability c(p,p) = 1.

3) The remaining b-p channels are assigned randomly to the remaining N-p nodes. Out of the N-p nodes, b-p nodes acquire the control of the remaining b-p channels, a single channel being allocated to a single node.

4) Each of the b-p nodes can transmit to N-1 destination nodes. Thus messages can be selected from a population of (b-p).(N-1) virtual users.

5) s is the total number of packets which succeed.

As mentioned in the network architecture, we define a collision as simultaneous transmission by two or more nodes on a single channel. We define a destination conflict as simultaneous arrival of two or more successful messages to the same destination node leading to successful reception of only one of the arriving messages. As pointed out in the model, every node i has N buffers where N-1 transmission buffers are dedicated each to a different destination j, 1<j<N, j=1 and one of the buffers is dedicated to the reception of messages.

We shall represent the messages by balls, the virtual users and the nodes by urns. Using the urn analogy, we can model the distribution of messages in a symmetric system as the distribution of n indistinguishable balls into m distinguishable urns with at the most v balls per urn. The total number of possible virtual users is

$$N' = N.(N-1)$$

There are i messages in the system.

To obtain the transition probability, the following probabilities are formulated.

A(i,p)- The probability of finding p disjoint pairs. This can be modelled as p permissions to the virtual users.

C(p,p)- The probability that out of p transmitted messages, p are successful. In our model, we assume it to be 1.

g(l,i-p)-the probability that out of i-p messages after the fixed allocation, l are transmitted collision free, given b-p permissions distributed among N-p nodes.

d(k,l)- The probability that k messages are received collision free given that l messages were transmitted.

A(i,p)-

To evaluate this probability, we use the urn model. Our network architecture is based on a closed queuing system i.e. packets or messages arrive only to idle buffers. Thus in our case, i balls are to be distributed among N.(N-1) urns with at the most one ball per urn under the condition that p out of N specific urns(nodes) contain exactly one ball and the remaining N-p are empty. Since occupancy is restricted to 1, we can use the Fermi-dirac model in this case. According to Fermi-dirac model, i balls can be distributed in

$$\begin{bmatrix} N.(N-1) \\ i \end{bmatrix} ways$$

and this gives us the sample space. We cannot use Bose Einstein statistics because of occupancy restriction. The number of ways in which p permissions can be given to N nodes is

$$\begin{bmatrix} N \\ p \end{bmatrix}$$

The number of ways in which the remaining i-p balls can be distributed among the remaining N(N-1) - N users is

$$\begin{bmatrix} N.(N-1)-N \\ i-p \end{bmatrix}$$

Thus the probability A(i,p) of permitting p disjoint pairs to transmit is

$$A(i,p) = \frac{\begin{bmatrix} N \\ p \end{bmatrix} . \begin{bmatrix} N.(N-1)-N \\ i-p \end{bmatrix}}{\begin{bmatrix} N.(N-1) \\ i \end{bmatrix}} \tag{3.9}$$

C(p,p)-

This probability is 1 since in the case of disjoint virtual users, both the source and destination and the channel number are known and there are no collisions.

g(l,i-p)-

In this case the occupancy restriction becomes N-1 equalling the number of users at each node. g(l,i-p) is the probability that i-p balls are distributed among N-p urns with at the N-1 balls per urn under the condition that exactly l urns from b-p specific urns contain at least one ball and the remaining b-l urns are empty. For the nmv urn model, the sample space is the total number of ways in which n balls can be distributed among m urns with at the most v balls per urn. Let R(n,m,v) denote this sample space. We define another term r(n,m,v) to calculate g(l,i-p).

r(n,m,v)- The number of ways of distributing n balls into m urns with at the most v balls per urn under the condition that exactly l urns out of b-p specific urns contain at least one ball and the rest b-p-l urns are empty.

To evaluate R(n,m,v), we use the method of generating functions. The generating function of the nmv model is

$$G(x,m,v)=(1+x+x^2+x^3+......+x^v)^m$$

with R(n,m,v) given by the coefficient of x^n in the above equation. G(x,m,v) can be written as br

$$G(n,m,v)=(1-x^{v+1})^m(1-x)^{-m}$$
$$=(1-x^{v+1})^m\sum_{j=0}^{\infty}\begin{bmatrix} m+j-1 \\ m-1 \end{bmatrix} x^j$$

Thus R(n,m,v) is given by

$$R(n,m,v)=\sum_{t=0}^{m}(-1)^t.\begin{bmatrix} m \\ r \end{bmatrix}\begin{bmatrix} m+n-r(v+1)-1 \\ m-1 \end{bmatrix} \qquad (3.10)$$

Let r(n,m,v) represent the number of ways of distributing n balls into m urns with at the most v balls per urn under the condition that exactly l urns out of b-p specific urns contain at least one ball and the remaining b-p-l urns are empty.

$$g(l,i-p)=\begin{bmatrix} b \\ l \end{bmatrix}.r(i-p,N-p,\frac{N-1)}{R(i-p,N-p,N-1)} \qquad (3.11)$$

To calculate r(n,m,v), we define K(n,m,v) to be the number of ways in which n balls can be distributed into m urns with at the most v balls per urn under the condition that l specific urns have at least one ball. To obtain K(n,m,v) we denote by a_r the property that r specific urns out of l given urns are empty. Let M(a_r denote the arrangements of the given nmv model satisfying property a_r.

$$M(a_r)=R(n,m-r,v)$$

Thus K(n,m,v) is given by

$$K(n,m,v)=\sum_{r=1}^{l}(-1)^{r-1}\begin{bmatrix} l \\ r \end{bmatrix} R(n,m-r,v) \tag{3.12}$$

Therefore r(n, m, v)can be analytically formulated as

$$r(n,\ m,\ v)\ =\ R(i\text{-}p,\ m\text{-}b\text{+}p\text{+}l,\ v)\ -\ K(i\text{-}p,\ m\text{-}b\text{+}l,\ v)$$

where R(n, m-b+p+l, v) represents the total number of arrangements in nmv model under the condition that b-p-l urns are empty. Substituting the eq 3.13 into eq 3.11 for m = N-p , we have

$$g(l,i-p)=\begin{bmatrix} b-p \\ l \end{bmatrix} \frac{\left[R(i-p,N-b+l,N-1)-\sum_{r=1}^{l}(-1)^{r-1}\begin{bmatrix} l \\ r \end{bmatrix} R(i-p,N-b+l-r,N-1)\right]}{R(i-p,N-p,N-1)} \tag{3.15}$$

Using eq 3.10, the above can be written as

$$g(l,i-p)=\begin{bmatrix} b-p \\ l \end{bmatrix} \left[\sum_{t=0}^{N-(b)+l}(-1)^t \begin{bmatrix} N-(b)+l \\ t \end{bmatrix} \begin{bmatrix} N-(b)+l+i-p-t(N)-1 \\ N-b+l-1 \end{bmatrix}\right.$$

$$\left.-\sum_{r=1}^{l}(-1)^{r-1}\begin{bmatrix} l \\ r \end{bmatrix} \sum_{t=0}^{N-b-r+l}(-1)^t \begin{bmatrix} N-b-r+l \\ t \end{bmatrix} \begin{bmatrix} N-b+l-r+i-p-t(N)-1 \\ N-b+l-r-1 \end{bmatrix}\right]$$

$$\times\frac{1}{\sum_{t=0}^{N-p}(-1)^t \begin{bmatrix} N-p \\ t \end{bmatrix} \begin{bmatrix} N-p+i-p-t(N)-1 \\ N-p-1 \end{bmatrix}} \tag{3.16}$$

d(k,l)-

To calculate d(k,l), we interpret the urns as the destination nodes. Thus in this case, l nodes which already have been assigned permission are taken to be balls. The p nodes which have already been given permission to transmit can receive. Thus N nodes in the network can receive. Therefore, in this case, l balls are to be distributed among N urns with at the most N-1 balls per urn under the condition that exactly k urns contain exactly i balls. d(k,l) is the same as the probability derived for source allocation protocol by I Chalamtac and O Ganz. [7]

The following were defined to calculate d(k,l)

$\eta_{i,k}$ - the probability that exactly k urns have exactly i balls given n balls are distributed among m urns with at the most v balls per urn.

$$\eta_{i,k}(n,m,v)=\frac{R(n,m,v\mid i,k)}{R(n,m,v)} \tag{3.17}$$

R(n, m, v | i, k)- the total number of arrangements in the nmv model under the condition that exactly k urns contain exactly i balls.

$$d(k,l)=\frac{\begin{bmatrix}N\\N-k\end{bmatrix}\sum_{r=0}^{N-(N-k)}(-1)^{N-(N-k)-r}\times\sum_{r_2=0}^{l+p(N-r)}(-1)^{r_2}\begin{bmatrix}l+p(N-r)\\r_2\end{bmatrix}\begin{bmatrix}l+p(N-r)-r(N-1)-1\\l+p(N-r)-1\end{bmatrix}}{\sum_{r=0}^{N}(-1)^r\begin{bmatrix}N\\r\end{bmatrix}\begin{bmatrix}N-rN+l-1\\N-1\end{bmatrix}} \tag{3.17}$$

From the fig 3.2 it is clear that s is the total number of messages which succeed.

$$s = p + k$$

$$\text{or } k = s-p.$$

Substituting the value of k in the above expression, we get

$$d(s-p,l)=\frac{\begin{bmatrix}N\\N-s+p\end{bmatrix}\sum_{r=0}^{s-p}(-1)^{s-p-r}\cdot\sum_{r_2=0}^{l+p(N-r)}(-1)^{r_2}\begin{bmatrix}l+p(N-r)\\r_2\end{bmatrix}\begin{bmatrix}l+p(N-r)-r(N-1)-1\\l+p(N-r)-1\end{bmatrix}}{\sum_{r=0}^{N}(-1)^r\begin{bmatrix}N\\r\end{bmatrix}\begin{bmatrix}N-rN+l-1\\N-1\end{bmatrix}} \tag{3.18}$$

The probability that a packet will be successfully transmitted in the parallel channel network is Suc(i,s). Suc(i,s) depends upon both the disjoint pair allocation probability

A(i,p) and the probabilities g(l,i-p) and d(s-p,l). For the movable boundary allocation protocol, the probability of success is formulated as

$$Suc(i,s) = \sum_{p=0}^{\min(i,b)} \left[\sum_{l=s-p}^{b-p} g(l,i-p).d(s-p,l) \right] A(i,p) \tag{3.19}$$

Substituting the expression for Suc(i,s) into eq(3.19) for p_{ij}, we get

$$p_{ij} = \sum_{s=0}^{\min(i,b)} \left[\sum_{p=0}^{\min(i,b)} \left[\sum_{l=s-p}^{b-p} g(l,i-p) \times d(s-p,l) \right] A(i,p) \right] B_{j-i+s,i}(1-e^{-lambda}). \tag{3.20}$$

Substituting the value of g(l,i-p) , d(s-p,l) , A(i,p) and $B_{j-i+s,i}$ (1-$e^{-\lambda'}$) from eqn 3.16, 3.18 and 3.9 into eq 3.20,we get

$$p_{ij} = \sum_{s=0}^{(\min} i,b) \left[\sum_{p=0}^{b} \left(\sum_{l=s-p}^{b-p} \binom{b-p}{l} \left[\sum_{t=0}^{N-(b)+l} (-1)^t \binom{N-(b)+l}{t} \binom{N-(b)+l+i-p-N(t)-1}{N-b+l-1} \right. \right. \right.$$

$$\left. \frac{-\sum_{r=1}^{l} (-1)^{r-1} \binom{l}{r} \sum_{t=0}^{N-b+l-r} (-1)^t \binom{N-b+l-r}{t} \binom{N-b+l-r+i-p-t(N)-1}{N-r-b+l-1}}{\sum_{t=0}^{N-p} (-1)^t \binom{N-p}{t} \binom{N-p+i-p-t(N)-1}{N-p-1}} \right]$$

$$\times \left[\frac{\binom{N}{N-s+p} \sum_{r=0}^{s-p} (-1)^{s-p-r} . \sum from r_2 = \overset{l+p(N-r)}{0} (-1)^{r_2} \binom{l+p(N-r)}{r_2} \binom{l+p(N-r)-r(N-1)-1}{l+p(N-r)-1}}{\sum_{r=0}^{N} (-1)^r \binom{N}{r} \binom{N-rN+l-1}{N-1}} \right] \Bigg]$$

$$\left. \times \frac{\binom{N}{p} \binom{N(N-1)-N}{i-p}}{\binom{N.(N-1)}{i}} \right]$$

$$\times \frac{(N'-i)!}{(N'-j-k)!(j+k-i)!} . (1-e^{-lambda})^{j-i+k} (e^{-lambda})^{N'-j-k} \tag{3.21}$$

As already stated, p_{ij} is the transition probability i.e. it is probability of the change of state of the system where the state of the system is measured by the number of customers at the slot beginning. i is the number of customers in the system. Since there can be at the most N' virtual pairs, i can be a maximum of N'. The eq(3.21) gives a transition probability matrix P for various states (i,j) and this matrix can have a maximum dimension of N'* N'. The transition probability matrix P can be substituted into the equations (3.5) to find the steady state probability vector $\Pi = \{\pi_0, \pi_1, \pi_2, \ldots\ldots\pi_{N'}\}$. The steady state probabilities are substituted in the equation (3.4) and (3.2) to derive a relation between throughput and delay.

The above equations were solved on SPARC 470 using C language. For N=8, N'=8 * 7 =56. Thus the transition probability matrix in this case is a 56 × 56 matrix. Also, the number of variables to be evaluated is 56. The 57th equation is $\sum_{i=0}^{N'} i\pi = 1$. However one of the 56 equations is always redundant according to []. Therefore in 56 linear equations are to solved for finding 56 variables. As N increases, the number of equations increase as $O(n^2)$. For a 25 node network, the number of variables is 600 and so is the number of equations. This requires a lot of memory and thus an efficient algorithm is used to solve the equations. The elements the matrix p_{ij} are written to a file called matx.data and the program lu.c takes in the the input from this file. The steady state probabilities π_i's are taken in by equation (3.4) to get the throughput.

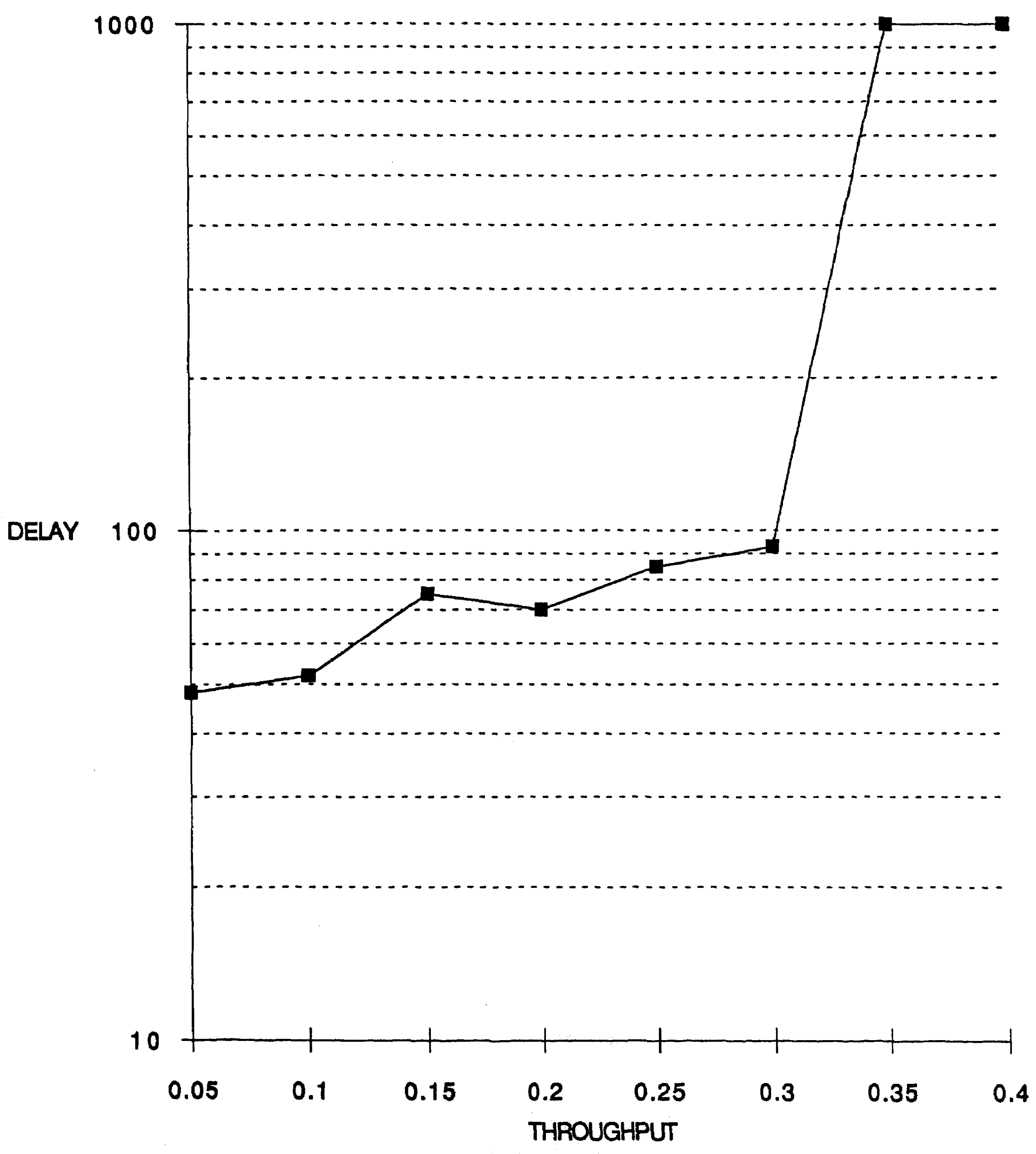
```
/* Program to find the transition probability matrix. This program puts the value

#include<stdio.h>
#include<math.h>
main()
{
double pwr(), comb();
double fact();
double sum=0.0,sum1=0.0,sum2=0.0,sum3=0.0,sum4=0.0,sum5=0.0,sum6=0.0,sum7=0.0,
sum8=0.0,sum9=0.0,sum10=0.0,sum11=0.0;
double result,result1,result2,result3,result4,result5,prob1,prob;
int t, b=4,n=8,n1=56,s,l,p,r,r1,r2,r3,t1,t2,i,j;
FILE *test;
for(i=0; i<=56; ++i){  /* i loop begin */
   for(j=0; j<=56; ++j) {  /* j loop begin */
      for(s=0; s<=b; ++s) {    /* s loop begin */
         for(p=0; p<=b; ++p) {    /* p loop begin */
            for(l=s-p; l<= b-p; ++l) { /* l loop begin */
               for (t=0; t<=n-b+l;++t)
                  sum += pwr(-1.0,t)*comb(n-b+l,t)
                         *comb(n-b+l+i-p- n*t -1,n-b+l-1);
               for(r3=1; r3<=l ; ++r3) {
                  for(t1=0; t1<=n-b+l-r3; ++t1)
                     sum2=sum2+pwr(-1.0,t1)*comb(n-b+l-r3,t1)
                            *comb(n-b+l-r3+i-p+(t1*n)-1,n-r3-b+l-1);
                  sum1+= pwr(-1.0,r3-1)*comb(l,r3)*sum2;
               }
               sum4= comb(b-p, l)*(sum-sum1);
               for (t2=0; t2<=n-p ; ++t2)
                  sum5 += pwr(-1.0,t2) * comb(n-p,t2)
                          * comb(n-(2*p)+i-(n*t2)-1,n-p-1);
               result1 = sum4 / sum5;
               /* d(k,l) =>  d(s-p,l)  */
               for(r=0; r <= s-p; ++r) {
                  for (r2=0; r2<= l+p*(n-r); ++r2)
                     sum6 += pwr(-1.0,r2)* comb(l+(p*(n-r)),r2)
                             * comb(l+(p*(n-r2))-(r2*(n-1))-1,l+p*(n-r2)-1);
                  sum7 += pwr(-1.0,s-p-r) * sum6;
               }
               sum8 += comb(n,n-s+p) * sum7;
               for(r1=0; r1<=n; ++r1)
                  sum9 += pwr(-1.0,r1) * comb(n, r1) * comb(n-(r1*n)+l-1,n-1);
               result2 = sum8/sum9; /* d(k,l) is result2  */
               result3 = result1*result2;
            } /*  l  loop  ends here  */
            result4 = result3*
                   (comb(n, p) * comb(n*(n-1)-n , i-p) / comb(n*(n-1),i));
         }    /*   p loop  ends  here  */
         /* result4  is  suc(i,s)  */
         prob1= (fact(n1 - i) * pwr(0.5,j-i+s) * pwr(0.5,n1-j+s))
                /(fact(n1-j-s)*fact(j+s-i));
         prob = result4 * prob1;
      }    /* s  loop  end  */
       fprintf(test,"%f\n",prob);
   }    /* j loop  end  */
   fprintf(test,"%f\n",prob);
}    /* i loop  end */
}

double fact(q)
int q;
{
         double result = 1.0;
         if (q==0)
                  result = 1;
         while (q != 1)
{
        result = result * q;
        q=q-1;
}
        return(result);
/*}else
                result = q * fact(q-1);
        return(result);*/
}

double comb (q1,q2)
int q1,  q2;
{
        double fact();
        double result;
        if (q1<q2)
        return(0);
        if (q1<0)
        return(0);
        if (q2<0)
        return(0);
        result = fact(q1) / (fact(q1-q2)* fact(q2));
        return(result);
}

double pwr(x,y)
double x; int y;
{
double prod = 1.0;
int i;

for(i=1; i<=abs(y); ++i)
   prod = prod * x;
   if(y<0)
    return(1/prod);
    if (y>0)
    return(prod);
}
```

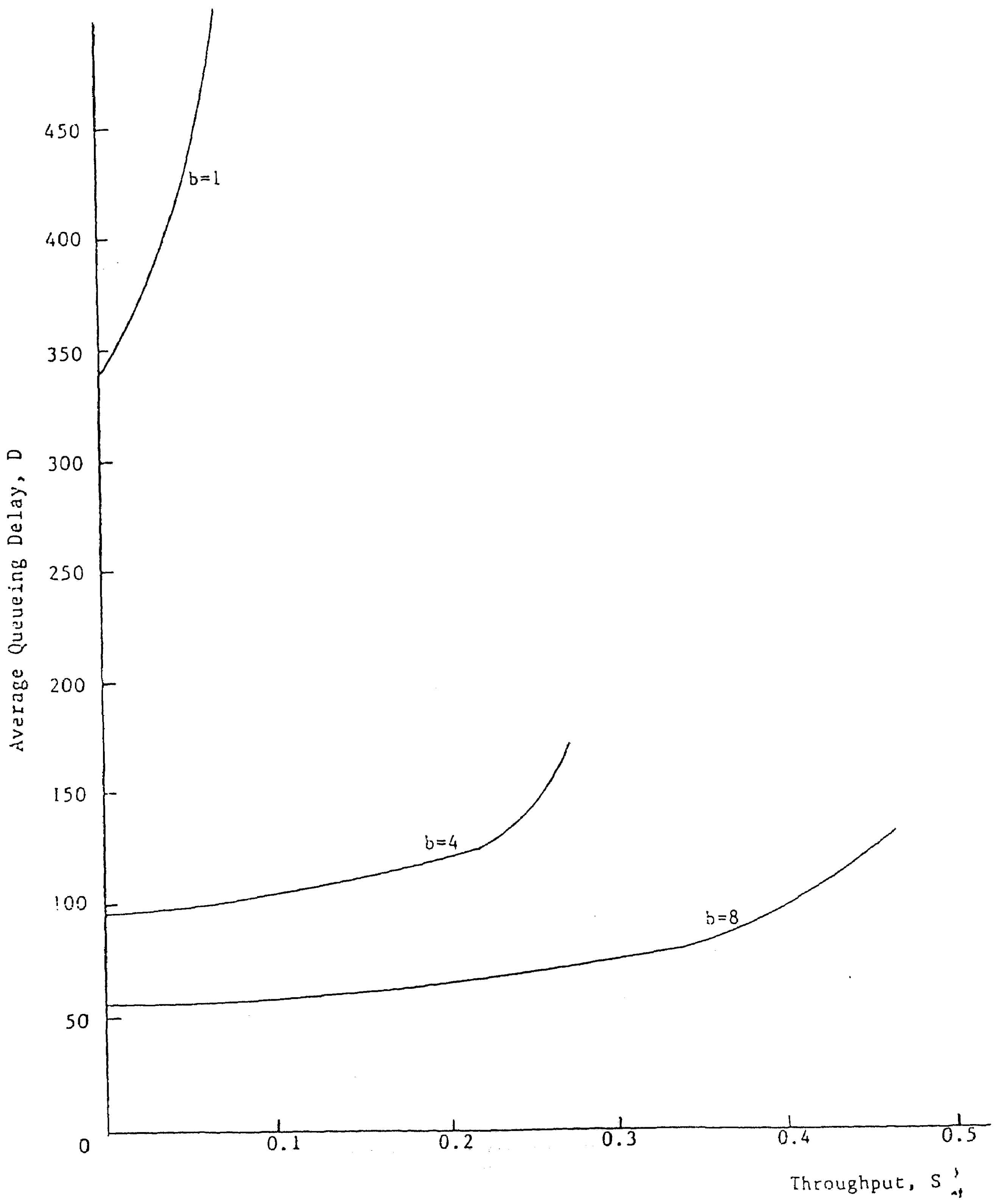
```
/* program to solve the 56 linear equations generated by the transition probabi

#include <stdio.h>
#include <floatingpoint.h>

#define      MAXLEN 100
#define MAXVEC     10

static double a[MAXVEC][MAXVEC];
static double b[MAXVEC];

main(argc, argv)
int argc;
char *argv[];
{
      FILE *fpin, *fpout;
      char line[MAXLEN], num[MAXLEN], *lp, *getword();
      int k, m, i, j, vecsize;
      double sum;
      void solve_for_x(), error();

      if (argc < 2)
             error("Usage: LU infile [outfile]\n", "");
      else if ((fpin = fopen(*++argv, "r")) == NULL)
             error("LU: cannot open %s\n", *argv);
      else if (argc == 2)
             fpout = stdout;
      else if ((fpout = fopen(*++argv, "w")) == NULL)
             error("LU: cannot open %s\n", *argv);

/*****************************************************************
 *
                       *
 *    An Example of Input File Format: [ A Matrix is 4 x 4 ]      *
 *
                       *
                   *           4            <--           Matrix
*
 *    10.0   7.0   8.0   7.0   4.0  <-- Row 1            *
 *     7.0   5.0   6.0   5.0   3.0  <-- Row 2            *
 *     8.0   6.0  10.0   9.0   3.0  <-- Row 3            *
 *     7.0   5.0   9.0  10.0   1.0  <-- Row 4            *
 *
                       *
 *      ^           ^           ^           ^           ^
                       *
 *      |           |           |           |           |
                       *
 *
                       *
 *    Col 1 Col 2 Col 3 Col 4 b Vector                         *
 *
                       *
 ****************************************************************/

      /* Get Matrix Dimension */
      if (fgets(line, MAXLEN, fpin) == NULL)
            error("LU: matrix dimension not specified in %s\n", argv[1]);
      else if ((vecsize = atoi(line)) <= 0)
            error("LU: dimension must be > 0", "");

      /* Read Data from file */
      for (i = 0; i < vecsize; i++) {
            if (fgets(line, MAXLEN, fpin) == NULL)
                 error("LU: too few rows in %s\n", argv[1]);
            for (j = 0, lp = line; j < vecsize; j++) {
                if ((lp = getword(lp, num)) == NULL)
                        error("LU: too few columns in %s\n", argv[1]);
                a[i][j] = (double) atof(num);
            }
            if ((lp = getword(lp, num)) == NULL)
                error("LU: too few columns in %s\n", argv[1]);
            b[i] = (double) atof(num);
    }

    for (i = 0; i < vecsize; i++) {
            for (j = 0; j < vecsize; j++)
                fprintf(fpout, " %12.6g", a[i][j]);
            fprintf(fpout, "\n");
    }
    /* Do In-Place LU Decomposition */
    k = 0;
    while (1) {
            for (j = k+1; j < vecsize; j++) {
                for (m = 0, sum = 0.0; m < k; m++)
                        sum += a[k][m] * a[m][j];
                a[k][j] = (a[k][j] - sum) / a[k][k];
            }
            if (++k >= vecsize)
                break;
            for (i = k; i < vecsize; i++) {
                for (m = 0, sum = 0.0; m < k; m++)
                        sum += a[i][m] * a[m][k];
                a[i][k] -= sum;
            }
    }

    /* Print out L Matrix */
    fprintf(fpout, "L =\n");
    for (i = 0; i < vecsize; i++) {
            for (j = 0; j <= i; j++)
                fprintf(fpout, " %12.6g", a[i][j]);
            for (j = i+1; j < vecsize; j++)
                fprintf(fpout, " %12.6g", 0.0);
            fprintf(fpout, "\n");
    }

    /* Print out U Matrix */
    fprintf(fpout, "\nU =\n");
    for (i = 0; i < vecsize; i++) {
            for (j = 0; j < i; j++)
                fprintf(fpout, " %12.6g", 0.0);
            fprintf(fpout, " %12.6g", 1.0);
            for (j = i+1; j < vecsize; j++)
                fprintf(fpout, " %12.6g", a[i][j]);
            fprintf(fpout, "\n");
    }

    /* Find A Inverse */
/*      for (i = 0; i < vecsize; i++)
            solve_for_x(I[i], vecsize); */
    /* Now Transpose of A Inverse is stored in I */

    /* Print out A Inverse Matrix */
/*      fprintf(fpout, "\nA Inverse =\n");
    for (i = 0; i < vecsize; i++) {
            for (j = 0; j < vecsize; j++)
                fprintf(fpout, " %12.6g", I[j][i]);
            fprintf(fpout, "\n");
    }
*/
    solve_for_x(b, vecsize);
```

```
	/* Print out x vector */
	fprintf(fpout, "\nx =\n");
	for (i = 0; i < vecsize; i++)
		fprintf(fpout, " %12.6g\n", b[i]);
}

/*
 *	Find x from LUx = v
 *	Return x in vec
 */
void solve_for_x(v, vecsize)
double *v;
int vecsize;
{
	int k, m;
	double sum;

	/* Do Forward Substitution to find y from Ly = v */
	v[0] = v[0]/a[0][0];
	for (k = 1; k < vecsize; k++) {
		for (m = 0, sum = 0.0; m < k; m++)
			sum += a[k][m] * v[m];
		v[k] = (v[k] - sum) / a[k][k];
	}

	/* Do Backward Substitution to find x from Ux = y */
	for (k = vecsize-2; k >= 0; k--) {
		for (m = k+1, sum = 0.0; m < vecsize; m++)
			sum += a[k][m] * v[m];
		v[k] -= sum;
	}
}

char *getword(s, w)
char *s, *w;
/*	Fetch a word from a string */
{
	/* First skip over leading white spaces */
	while (*s == ' ' || *s == '\t')
		s++;
	if (*s == '\0' || *s == '\n')
		return(NULL);
	while (*s != ' ' && *s != '\t' && *s != '\n' && *s != '\0')
		*w++ = *s++;
	*w = '\0';
	return(s);
}

void error(s1, s2)
char *s1, *s2;
{
	fprintf(stderr, s1, s2);
	exit(1);
}
```

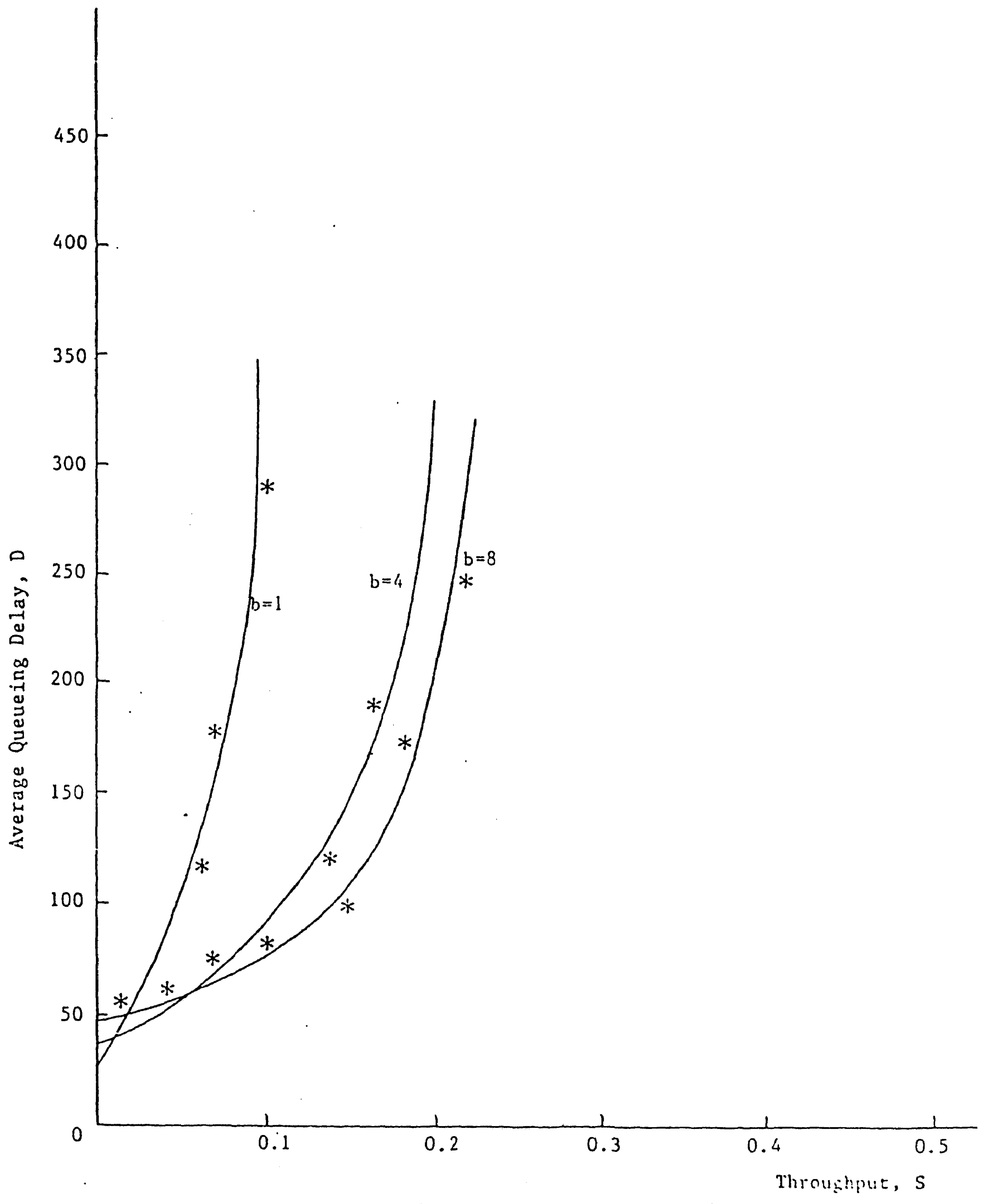
Chart4

THROUGHPUT vs. DELAY for MBA PROTOCOL

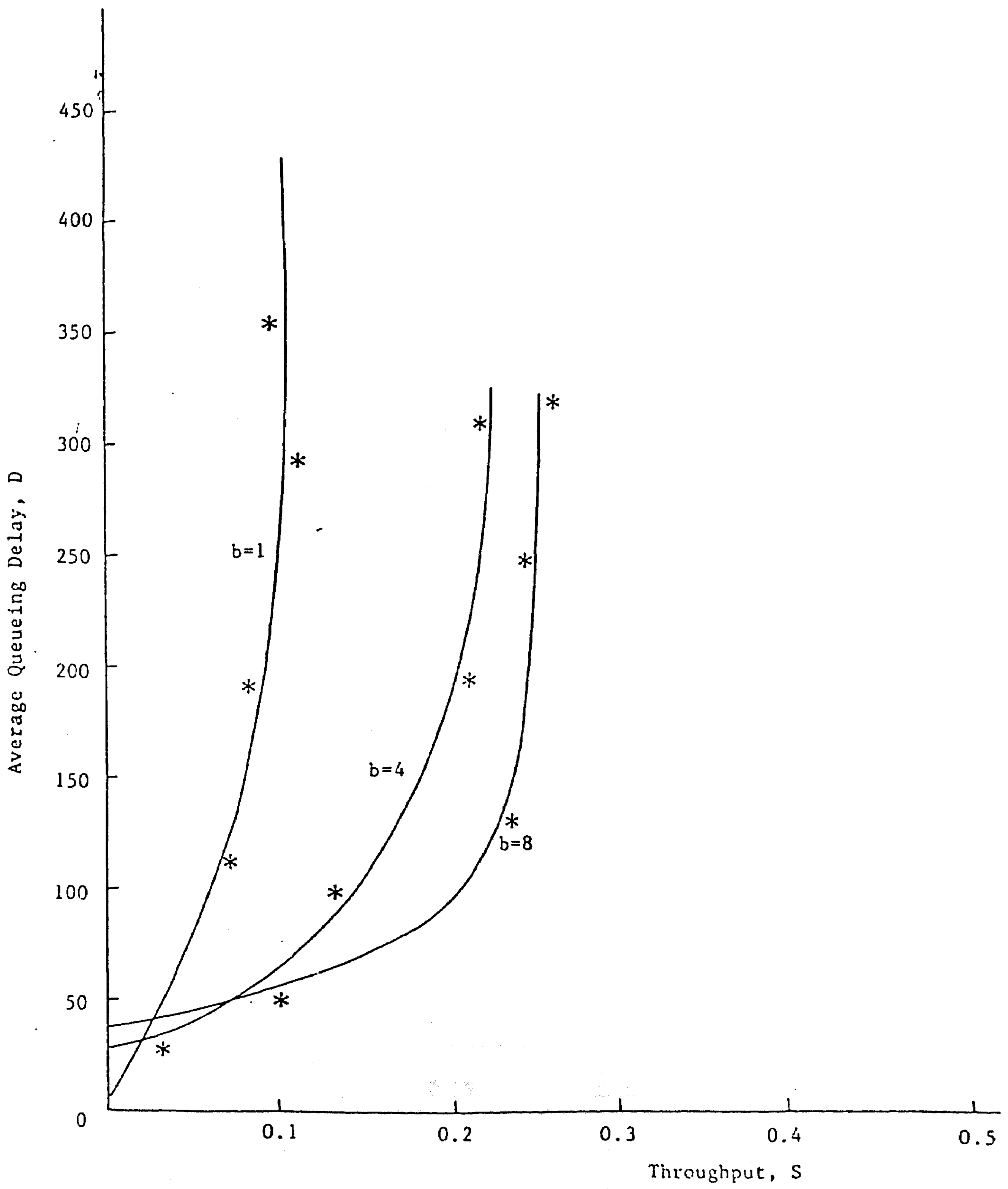




Average queueing delay versus throughput for source/destination allocation protocol ($N = 8$, $p = 10$).

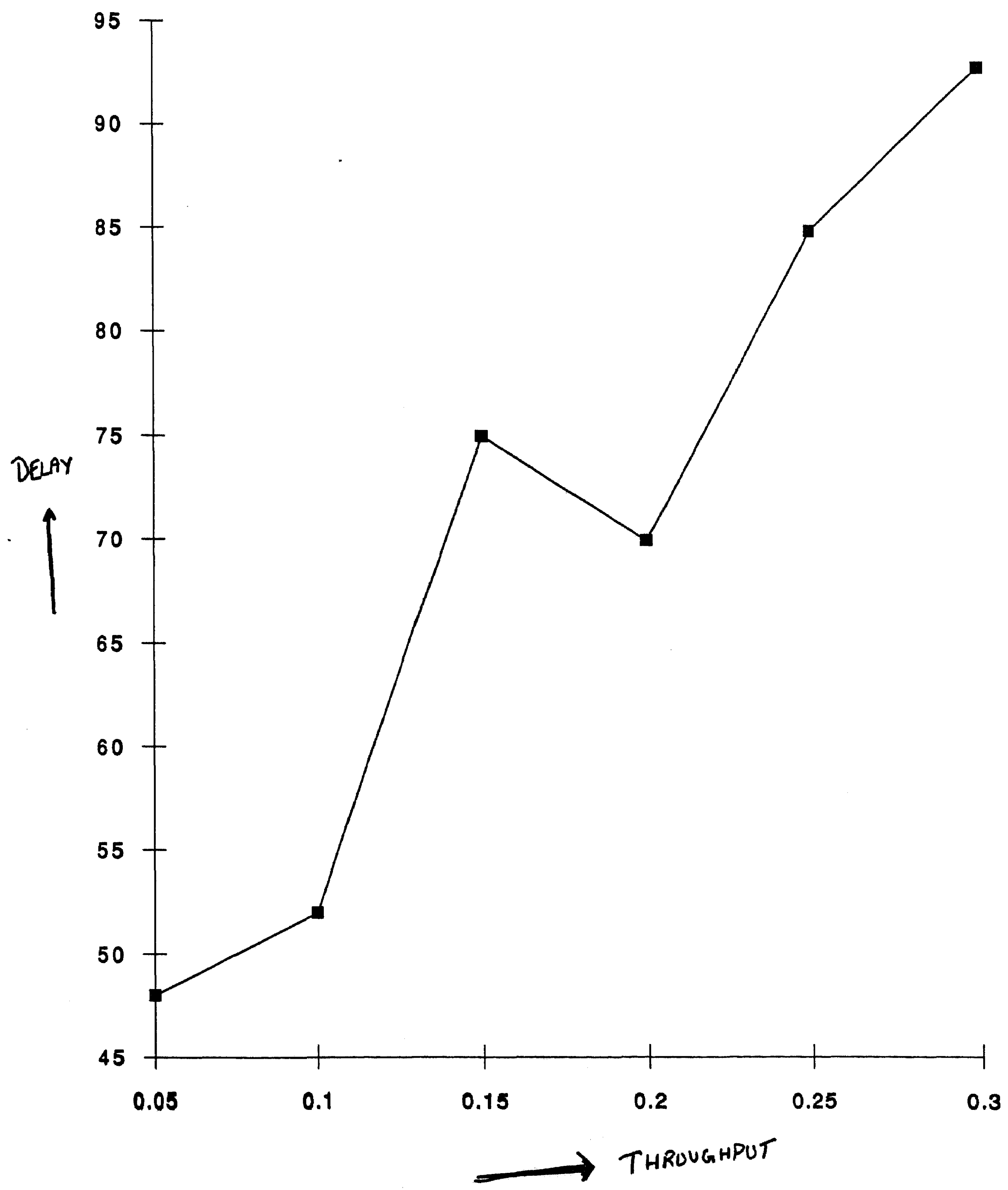


Average queueing delay versus throughput for destination allocation protocol ($N = 8$, $p = 10$).



Average queueing delay versus throughput for source allocation protocol (N = 8, p = 10).

Chart3 [MBA PROTOCOL]



CHAPTER 4

CONCLUSIONS

The movable boundary allocation protocol was designed and analysed for the parallel channel network architecture. A graph was plotted for the throughput and delay and this was compared to the throughput and delay characteristics for the other protocols. The MBA protocol produced slightly better results than the source allocation protocol.