MEDIUM ACCESS DELAY EVALUATION FOR DISTRIBUTED QUEUEING DUAL BUS (DQDB) MAC PROTOCOL

by

Don H. Pham

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

Dr. F.J. Ricci, Chairman

Dr. J.A. Knight

Dr. D.J. Schaefer

May, 1992

Blacksburg, Virginia

MEDIUM ACCESS DELAY EVALUATION FOR DISTRIBUTED QUEUEING DUAL BUS (DQDB) MAC PROTOCOL

by

Don H. Pham

Committee Chairman: Dr. F.J. Ricci

Electrical Engineering

(ABSTRACT)

Distributed Queuing Dual Bus (DQDB) is a media access control (MAC) technique, which is being considered by the IEEE 802.6 for the metropolitan area network (MAN). The DQDB medium access technique has many promising advantages over other access methods. However it has one drawback, which is its unfairness in terms of node-dependent medium access delay.

In this paper a mathematic model is formulated to describe this detrimental behavior of DQDB. The access control method is first modelled as a M/G/1 queueing system with a single priority level, then it is remodelled as a non-preemptive priority system with three priority levels. By employing these models, the approximate medium access delay analysis of a DQDB network is investigated for a metropolitan area network containing 50 stations with a channel bandwidth of 150 Mbps. Numerical results are then presented to illustrate the network unfairness performance under various traffic intensities and under different priority levels. The results have been obtained for non-isochronous (asynchronous) traffic.

Acknowledgement

I would like to extend my deepest gratitude to *Dzung* and *Lester Steinfeld* for their support, their proofreading and helpful comments for revision of this paper, and especially for their constant encouragement. Special thank to my niece, *Dagania*, who has not disturbed my sleeping in the early mornings, which I most needed to work on my computer programs through the nights. Last but not least, special thanks to all *Professors* and *Teachers*, who had shaped my educational background and enriched my knowledge in the field of engineering that made it possible for me to carry out my research and complete this paper.

— § —

Table of Contents

— § —

List of Figures

— § —

List of Abbreviations

DQDB - Distributed Queueing Dual Bus

ACF - Access control field

MAC - Medium access control

REQ - Request signal

REQ_CNT - Request counter

CD_CNT - Countdown counter

QA - Queued Arbitrated

PA - Pre-Arbitrated

LAN - Local area network

MAN - Metropolitan area network

WAN - Wide area network

PBX - Private branch exchange

B-ISDN - Broadband integrated service digital network

PMD - Physical medium dependent

QPSX - Queued packet synchronous exchange

CSMA/CD - Carrier sense multiple access with collision detection

FIFO - First in first out

— § —

1 Introduction

In recent years, the number of high-speed local area networks (LANs) in industry and commerce has increased rapidly. The stations or mainframe computers in these LANs perform highly sophisticated programs. Thus, the LANs are required to handle an enormous amount of data within the networks; and at the same time users demand a close-to-real-time application.

The number of these LANs certainly will continue to grow in the future, as will the areas of coverage by the LANs. Therefore, an interest in the interconnection of these LANs and the LANs with other private branch exchange (PBX) networks has risen in the telecommunication industries.

This kind of interconnection can only be efficiently done with the deployment of high-speed metropolitan area networks (MANs). By this method the interconnection networks can keep up with high-speed demand. Usually data rates of 50 up to 150 Mbps or beyond are required. Obviously, gateways and bridges are required to provide the appropriate corresponded speed and the necessary flow control.

A high-speed network can be categorized as a network having a propagation delay

not negligible when comparing to the transmission delay. This attribute can usually be found in modern high-bandwidth or long-haul communication networks using packet switching.

Many existing protocols, such as Token Passing Bus or Ethernet, waste too much channel bandwidth due to their access controls. The intrinsic characteristic of these medium access control (MAC) protocol limits the network transmission speed. These protocols are obviously inadequate and inefficient for high-speed MANs.

A new medium access control protocol, Distributed Queueing Dual Bus (DQDB), has emerged as a leading technology for the high-speed MANs. The DQDB medium access protocol is based on the queued packet synchronous exchange (QPSX) MAN protocol developing by Telecom Australia [1], which was designed to provide both isochronous and non-isochronous (asynchronous) service such as voice, data and video on the same high-bandwidth channel.

The DQDB medium access protocol employs the concept of a distributed queueing service discipline. Consequently the network can reach one hundred percent of channel efficiency. Another advantage of the DQDB MAC protocol is that a node can enter and exit a DQDB network without any complex reconfiguration of the network.

Due to those favorable advantages, the DQDB standards are being developed with strong support from the telecommunication industries. They are currently being standardized in the IEEE working group 802.6 [8]. DQDB is also being considered by the ANSI T1S1.1 committee as an element in the B-ISDN User Network Interface standard.

Currently there are only a few models for the IEEE 802.6 DQDB MAC protocol; and these models can only simulate a DQDB network with a single priority level within the non-isochronous traffic. Therefore, the interested information on the medium access delay of a DQDB network with multiple priority levels is unavailable for quantitative evaluation. Hence, the ultimate goal of this thesis is to develop a working model for the DQDB MAC protocol capable of simulating multiple priority levels service. The model is then used in a simulation of a DQDB network with three priority levels within the non-isochronous traffic, and the experimental results along with the medium access delays are reported. The model is also used to simulate a DQDB network with one priority level to confirm the claim of access delay unfairness associated with DQDB networks.

This paper is divided into seven (7) chapters. Chapter 1: the introduction is presented. Chapter 2: the physical organization of the DQDB network is described. Chapter 3: the DQDB MAC protocol is presented. Chapter 4: notations and general assumptions concerning the models are stated. Chapter 5: the analytical models are

described; the M/G/1 queueing model and the non-preemptive priority queueing service model are analyzed with the usual hypothesis: symmetric Poisson traffic and infinite buffer capacity. Chapter 6: The application of the models to a MAN of 50 nodes with a channel capacity of 150 Mbps, and the results are presented. Chapter 7: the conclusion remarks and suggestion for further study are presented.

— § —

2 Network Overview

The DQDB network is based on a fiber-optic network operating on two 150 Mbps unidirectional buses with signals propagating in opposite directions (see Figure 1). Both buses are operated simultaneously, thus making the bandwidth of the network twice that of a single bus. Access units (AUs) or nodes[1] are situated along the buses. The nodes are connected to both buses via read taps and unidirectional OR-write taps to provide each node with a direct path to every other node and a full duplex communication. The read taps are on the upstream[2] of the OR-write taps so that all the information, which is read by a node, will not be altered by the node's own writing (see Figure 2).

Each node implements the transfer of messages in many discrete sections, which are called packets. A packet consists of: a destination address field, a source address field, a number of data bytes, following by an error check sequence field. Each node is passive, but has an ability to read all data packets and all the information packets pertaining to the MAC as they pass by. Logically, a node only copies into the reassembly buffers the data packets for which the node is the destination. Each node is

---

[1] In this paper, access unit, station and node are used interchangeably when referring to the place where data packets are sent and received.

[2] Throughout this paper, the term "upstream" refers to the direction against the flow of information, and "downstream" refers to the direction of the flow of information.

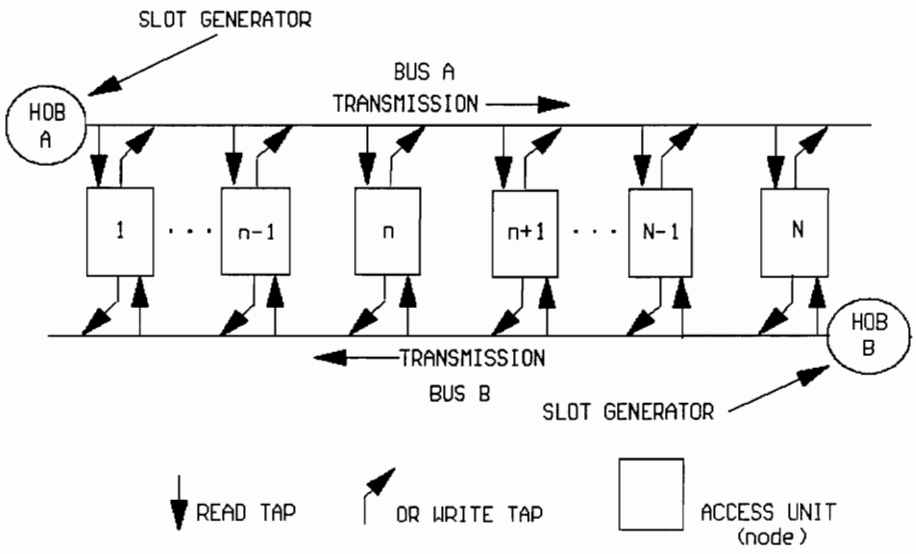

Figure 1 DQDB network configuration.

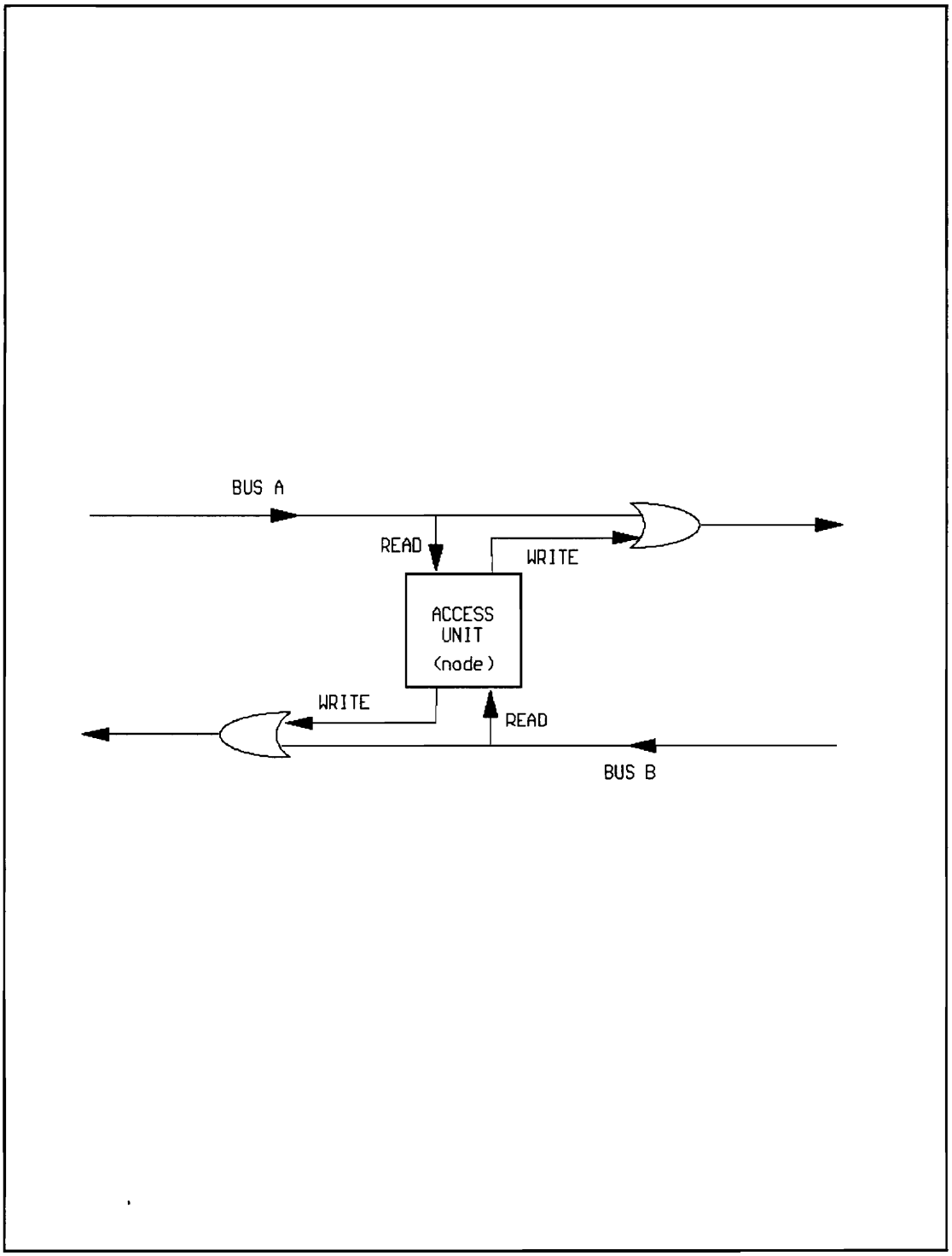

Figure 2 Media input/output interface of a DQDB node

capable of reassembling packets of different messages simultaneously, since a node can receive packets from other different nodes in no particular order at the same time.

Each node can be considered as a bridge connecting a LAN with other LANs to form a MAN, as in Figure 3. For hierarchical layer, these nodes can be gateways[3] interconnecting MANs over a wide area network (WAN) [1]. The bridges make no modification on the content of packets nor do they add on any information to the address fields.

Each node can only communicate with the downstream nodes on any one bus. Therefore the node desiring to transmit signals must know the locations of the receiving nodes with respect to its own position within the network, so that signals can then be transmitted on the appropriate bus.

At the head of each bus is a slot generator. It periodically generates a train with a fixed number of slots. The size of each slot is 69 bytes[4] as shown in Figure 4: one (1) byte for the access control field (ACF), four (4) bytes for the segment header, and 64 bytes for the segment payload. The slot structure is approved by the IEEE 802.6 and

---

[3] Bridges operate at the logical link or medium access control layer; in contrast, gateways use information at the network layer and operate at layer 3 of the OSI.

[4] In the earlier version of the draft standard, the IEEE 802.6 committee set a slot to be 53 bytes. According to the updated draft standard, a slot is now 69 bytes.

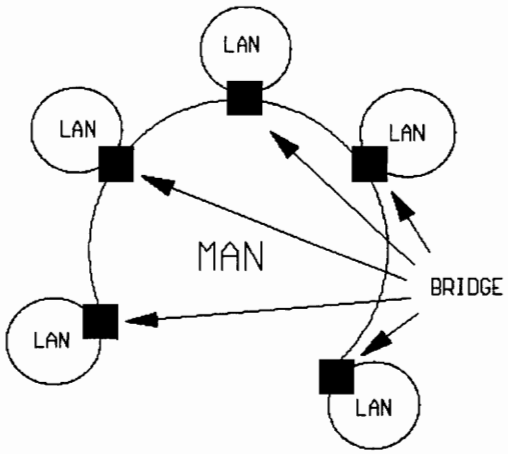

Figure 3 Interconnection of LANs.

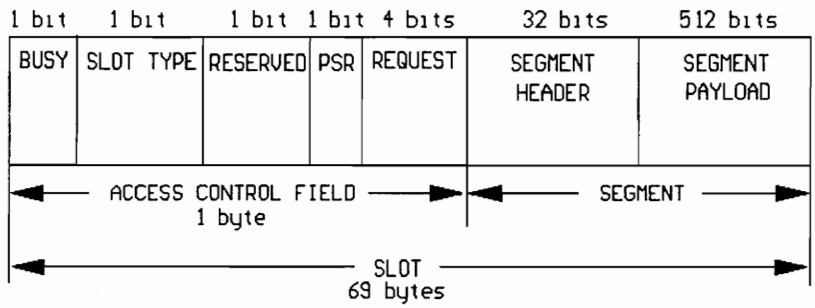

Figure 4 Slot structure diagram.

by the T1S1.1 to switch packets between MANs and broadband-ISDNs (B-ISDNs) [1][3]. Both slot generators generate slots at the same rate, but they operate independently.

Each slot contains control information in its ACF as follows:

- A busy indicator, which indicates whether the slot is used, otherwise the slot is an empty slot.

- A slot-type indicator, which is used to designate the slot as a Pre-Arbitrated (PA) slot for isochronous service, or as a Queued Arbitrated (QA) slot for non-isochronous (asynchronous) service.

- A request-to-send indicator, which is used by a node to send a request (REQ) to all upstream nodes indicating that there is an information packet of a certain priority level queueing for an empty slot, so that the information packet can be sent without interference to a downstream node.

- A PSR indicator, which indicates whether the slot can be erased for reuse -- this option is still under further study by the IEEE 802.6 committee and will not be considered in the medium access delay analysis. At present a slot can only be

used once to send a data packet or a REQ.

DQDB provides real time voice services through isochronous traffic. Therefore it uses a fixed frame structure of equal sized slots. The frame interval is allocated to be 125 microseconds to meet the telephony standard. The frame length is set by the need to digitize voice once per 125 microseconds.

A DQDB network can provide isochronous and also non-isochronous communication. For isochronous communication, such as real-time telephone voice or video service, a fixed part of each frame is designated as PA slots for isochronous service. The rest of the slots are used for non-isochronous communication. The PA slots are pre-assigned at the slot generator and to be shared only by the selected nodes. The allocation of channel capacity for isochronous service depends only on the number of nodes subscribing for this type of service. If there is no subscriber for isochronous service, the total bandwidth is dedicated for non-isochronous traffic.

For non-isochronous communication, a distributed queueing service discipline is used as the access scheme. Data packets are queued in the node's buffers and are sent via the QA slots. QA slots can be used by all nodes for sending REQs and data packets.

DQDB provides three priority levels[5] of service within the non-isochronous traffic [7]. The higher priority packets are served before the lower priority packets.

Isochronous traffic is independent of non-isochronous traffic. It does not affect the performance of the non-isochronous traffic except by sharing the bandwidth. Thus isochronous traffic increases the load of the network so increasing the unfairness in the medium access for non-isochronous packets.

The outstanding features of the distributed queueing access protocol are: (i) there is no specific upper limit on the number of nodes affiliated to the DQDB network; (ii) there is no maximum length of the media; (iii) no maximum speed to confine the MAC performance of the network [1].

These are the advantages over the Token Ring network, where the number of nodes in the ring network topology affects the average packet access delay time; and over the Ethernet using carrier sense multiple access with collision detection (CSMA/CD), where the data packets' collision possibilities, which hinder the packet access time, depend chiefly on the current traffic intensity, number of nodes in the network, and the physical length of the media. On the contrary, for a DQDB network, the average packet

---

[5] As of May 1990, IEEE 802.6 committee has decided that DQDB will have three priority levels instead of four.

access delay time depends only on the current amount of network traffic intensity.

However, the DQDB access protocol is not without liability due to the distributed queueing protocol. A node's medium access delay strongly depends on the physical order of the node within the network. This is the unfairness associated with a DQDB network.

The distance between nodes does not affect the medium access delay, but only affects the propagation delay as a part of the transmission delay. Transmission delay is referred to as the total time a packet spends in the network, which is a sum of segmentation delay, medium access delay, propagation delay, receiver access delay, and reassemble delay.

The interest addressed in this paper is the node's medium access delay time per packet due to the distributed queueing. Therefore, only non-isochronous traffic using QA slots is considered in the analysis. The exact derivation of the average access delay time of the DQDB distributed queueing is a very intricate problem, therefore only bounds and approximates for the case of a single priority level are currently known. However, in any practical communication network, multiple priority service must be available to meet the urgency of some messages. For this reason, a general knowledge of the access delay time of a non-preemptive priority service network is of interest, so that a conclusion

about the DQDB network can be drawn.

The case of a network with a single priority level is analyzed first and its results are compared with other known results to ensure that these results are dimensionally correct. Next, a network with three priority levels is examined to see the effect of non-preemptive priority service on a DQDB network.

The models for the DQDB MAC protocol will be examined through the use of probability modeling. An approximated mathematical model will be formulated to (i) illustrate the distributed queueing service discipline of DQDB; (ii) confirm the claimed unfairness of DQDB network; and (iii) demonstrate any new feature of unfairness of a DQDB network.

— § —

3 DQDB MAC Protocol

The description of the MAC protocol is based on a network consisting of *N* stations, labeled in order with the integers from **1** to **N** (see figure 5). The transmission of bus **A** is from left to right; the transmission of bus **B** is the reverse.

The non-isochronous traffic accessing to the buses is controlled by the distributed queueing MAC protocol [8]. To facilitate managing of the distributed queueing service discipline, every node is comprised of two outfits, one for each bus.

Each outfit consists of: one countdown counter (CD_CNT); and for each priority level a request counter (REQ_CNT_j), where *j* denotes the priority level, and a buffer housing the data packets of priority level *j*. For example, for three priority levels there will be three REQ_CNTs and three separate buffers. $j=\{1,2,3\}$, where the smaller index number indicates the higher priority level.

As usual, arriving messages are segmented into packets. Each packet contains the exact amount of information, that can be written on the segment payload of a slot. Stuff bits would be used to fill in if the packet contains less information. These packets along with their headers and ACFs are placed in the appropriate buffers according to the

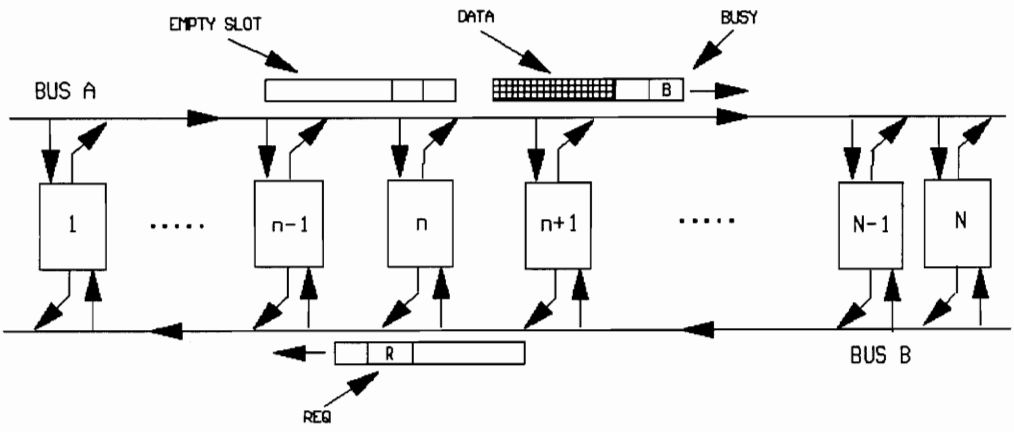

Figure 5 Network layout.

priority level of the message.

If node **n** desires to communicate with nodes **n+1** through **N**, it must utilize bus **B** to send the requests (REQs), and using bus **A** to send the data packets. In the other direction, to communicate with nodes **1** through **n-1**, node **n** must utilize bus **A** to send the REQs, and using bus **B** to send data packets.

A node can only be in one of two modes [7] idle or active, as shown in the state machine (see Figure 6). A node is in the IDLE state when there is no packet queueing for service. In the IDLE state, when a REQ_i of priority level *i* is read from the bus **B**, the node's REQ_CNTs of priority level *i* and lower increase by one. When an empty slot passes by on bus **A**, all the REQ_CNTs, regardless of the priority level, decrease by one. Obviously a REQ_CNT remains the same if its value is zero. The node in the IDLE state is applicable to both buses by interchanging the letter **A** and **B**.

Medium access on bus **A** is identical with the medium access on bus **B**. Therefore only the medium access on bus **A** will be described. This implies that the process of sending a packet form node **n** to node **m**, where $\mathbf{m} > \mathbf{n}$, will be described.

A node is in the ACTIVE state when one or more packets are queueing for service. Packets of a higher priority level are served before packets of a lower priority

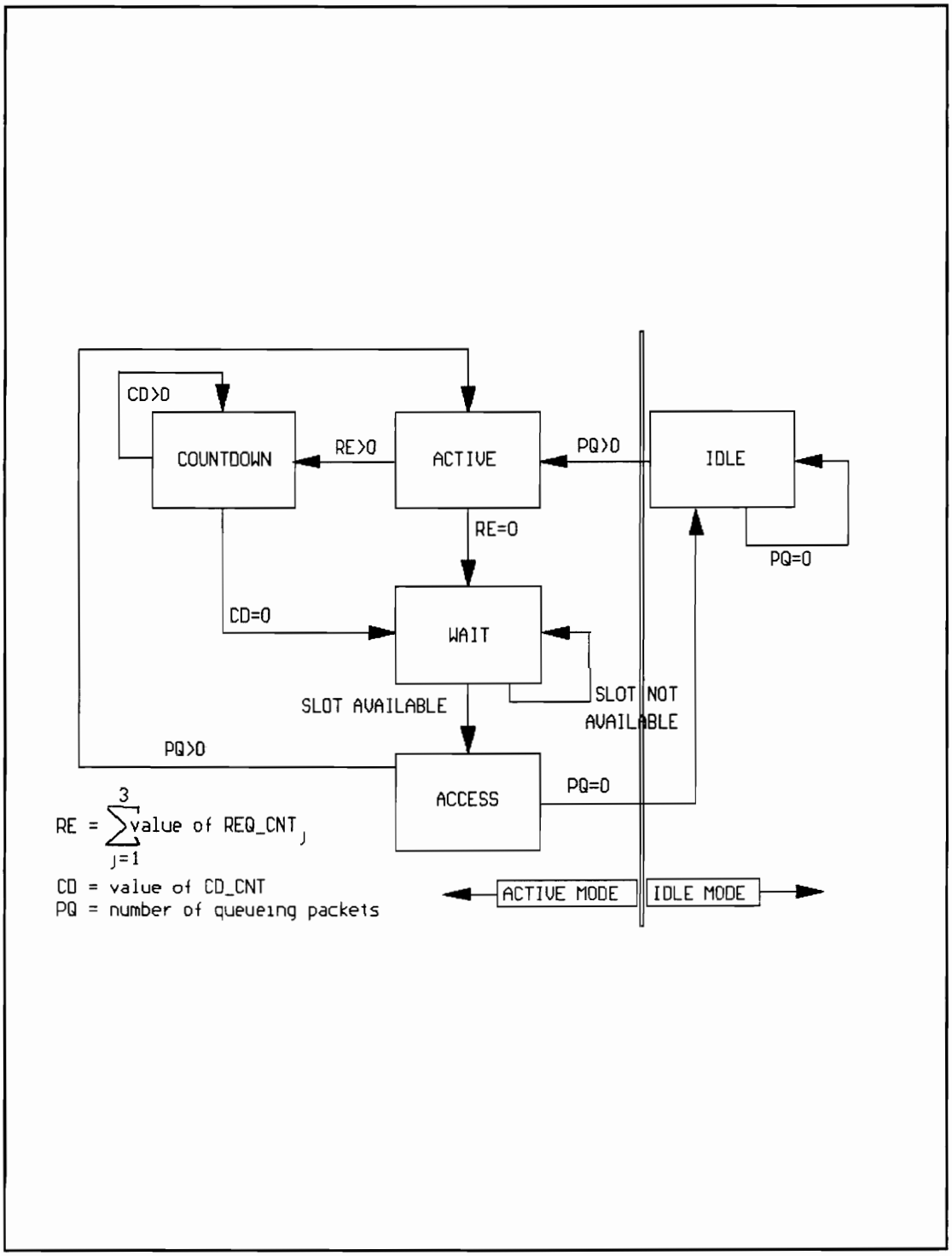

Figure 6 DQDB MAC protocol state diagram.

level. This process follows the non-preemptive priority queueing service.

When an information packet of priority *j* arrives at the head of the queue, referred to as the serving position in this paper, the node checks the value of the REQ_CNT_j. If its value is zero, the node enters the WAIT state and sends the information packet on the first empty slot arriving on bus **A**. Otherwise, the node enters the COUNTDOWN state and has to send a REQ_j (where *j* denotes priority level of the message) via the first free request indicator on a slot arriving on bus **B** to acknowledge the upstream nodes. Simultaneously the node transfers the value of the REQ_CNT_j to the CD_CNT. During the wait for a free slot to arrive on bus **B**, the REQ_CNTs behave as in the IDLE state.

When the REQ_j has been sent, the CD_CNT decreases by one for every empty slot passing by on bus **A**. Meanwhile, all the REQ_CNTs resume their incremental function only. When a REQ_i passes by on bus **B**, REQ_CNTs of priority level *i* and lower increase by one. The REQ_CNTs proceed with the incremental task only while the value of the CD_CNT is not zero. The node must allow as many empty slots to pass as the value of the CD_CNT. Only when the value of the CD_CNT equals zero, then the node can send the packet via the next empty slot arriving on bus **A**. The process returns to the ACTIVE state recommencing for the next queueing packet. If there are no more packets to send, the node returns to the IDLE mode.

Each node can send only one REQ at a time. It cannot send another REQ until the transmission of the previous data packet is satisfied. In this way, the number of REQs in the REQ_CNT will not be added to the previous number of REQs. This algorithm is ensured to provide a first-in-first-out (FIFO) access mechanism.

Nodes continuously monitor the buses. By counting the REQs and the empty slots that pass by, a node always knows how many empty slots are reserved by the downstream nodes. From this information, a node can determine its position in the distributed queue knowing exactly when it has a right to access the bus. Consequently, no slot is wasted when there is at least one node having a packet to be transmitted. As a result, the channel bandwidth can be utilized one hundred percent efficiently, unaffected by the size and the rate of the network [1].

According to the DQDB MAC protocol, each node administers its own distributed queueing discipline. Therefore, the reliability of the buses is not effected by a node failure. Failed nodes will not prevent all other nodes from using the network. Moreover, a reconfiguration of the network is not necessary when a new node enters the network. A node can be connected to the network just by tapping into the media.

— § —

4 Model Assumptions and Definitions

Certain assumptions, which were applied to the models, are necessary for simplicity and tractability. However, the model is kept within a neighborhood of resembling a real network, so that the modeled network still retains those essential characteristics.

4.1 Notations

The following is a list of notations and their definitions, which are used in the derivation of the probability models to describe the behavior of the distributed queueing MAC protocol.

n - the referent node or tagged node.

N - total number of nodes connected to the network.

$\frac{1}{\mu}$ - constant length of packets, [bits/packets].

$\frac{1}{\eta}$ - length of slots, [bits/slot].

C - channel capacity of each bus, [Mbits/second].

t - transmission time of a packet over the channel, [seconds/packet].

$$t=\frac{1}{\mu C}$$

τ - duration of a slot, [seconds/slot].

$$\tau=\frac{1}{\eta C}$$

ν_{nmj} - average arrival rate of packets with priority level j entering the network having source node **n** and destination node **m**, [packets/second].

λ_{nm} - average arrival rate of packets entering the network having source node **n** and destination node **m**, [packets/second].

Λ_{nj} - average arrival rate of packets per second with source node **n** for priority level j, [packets/second].

V_{nj} - sum of the average arrival rate of packets per second with source node **n+1** through source node **N-1** for priority level j, [packets/second].

ρ_{nj} - share of bandwidth of priority level j at node **n**, [dimensionless].

β_{nj} - total share of bandwidth of node **1** through node **n-1** for priority level j, [dimensionless].

Γ_A - traffic intensity of bus **A**, the percentage of channel bandwidth under utilization, [dimensionless].

Q_j - average queueing delay for a packet of priority level j, [slots/packet].

$\overline{MAC_j}$ - average MAC delay for a packet of priority level j, [slots/packet].

$\overline{Y}_j$ - the average processing time for a packet of priority level j, [slots/packet].

σ^2_{Yj} - the variance of the processing time for a packet of priority level j, [slots/packet]^2.

T_j - average medium access delay for a packet of priority level j, [slots/packet].

4.2 Assumptions and Definitions

The following conditions are assumed in determining the distributed queueing service model:

1. A node is as likely to send a packet to another node as to every other node.
2. Poisson distribution for the packet arrival rate for each node.
3. Constant packet length, which can be written in the space of a slot less its ACF.
4. Unlimited buffer capacity for each priority level at each node to supply waiting room for the queueing packets.
5. There is always at least a packet in the buffer for service.
6. There is always at least a REQ in the REQ_CNTs.

Since this paper is restricted to the analysis of the packet medium access delay, the evaluation begins after the segmentation of messages into packets[6]. The evaluation is initiated once a packet enters a buffer, and ended when the packet is written on a slot. The medium access delay of a packet is a result of three sources of delay: *queueing delay*, *request access delay* and *data access delay*. Figure 7 shows the physical relationship between the different delays.

§ Queueing delay, Q, is the time a packet spends in the buffer waiting for transmission, which is the interval of time from the moment a packet enters the buffer to the moment it approaches the serving position.

§ Request access delay, r, is the amount of time from the moment a packet arrives to the serving position to the moment a REQ for this packet is successfully sent.

§ Data access delay, d, is the amount of time from the moment a REQ for the queueing packet is sent to the moment the data of this packet is actually written on an empty slot.

---

[6] All messages coming into the network are assumed to be equal in length; hence all nodes experience the same segmentation delay. The segmentation delay does not affect the unfairness of DQDB network.

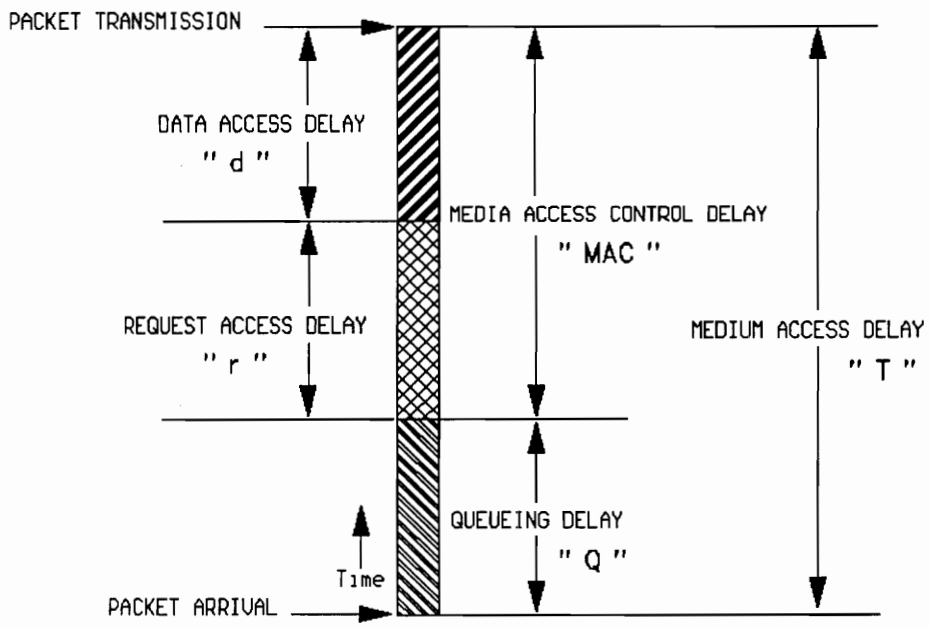

Figure 7 Delay time diagram.

The average MAC delay, $\overline{MAC}$, values the average amount of time a packet is delayed due to the MAC protocol. Therefore the average MAC delay is defined to be the sum of the average request access delay and the average data access delay. Consequently, the sum of the average queueing delay, Q, and the average MAC delay represents the average packet medium access delay, T.

Since there is no actual data available to evaluate the network, and since it is preferred to analyze the medium access delay as a function of the traffic intensity, a square matrix NxN representing the overall traffic condition of the network (see Figure 8) is used to determine the average packet arrival rate, λ_{nm}, for a given traffic intensity Γ_A.

For clarity, only the access technique for bus **A** is analyzed; access techniques for bus **B** and bus **A** are reciprocal. The entries of the traffic matrix are the average arrival rates of packets which has a source node **n** and a destination node **m**. The columns represent the source nodes and the rows represent the destination nodes. There are no entries in the lower triangle because bus **A** is unidirectional. Based on this matrix and the defined notations above, the average arrival rates along with other relations are derived.

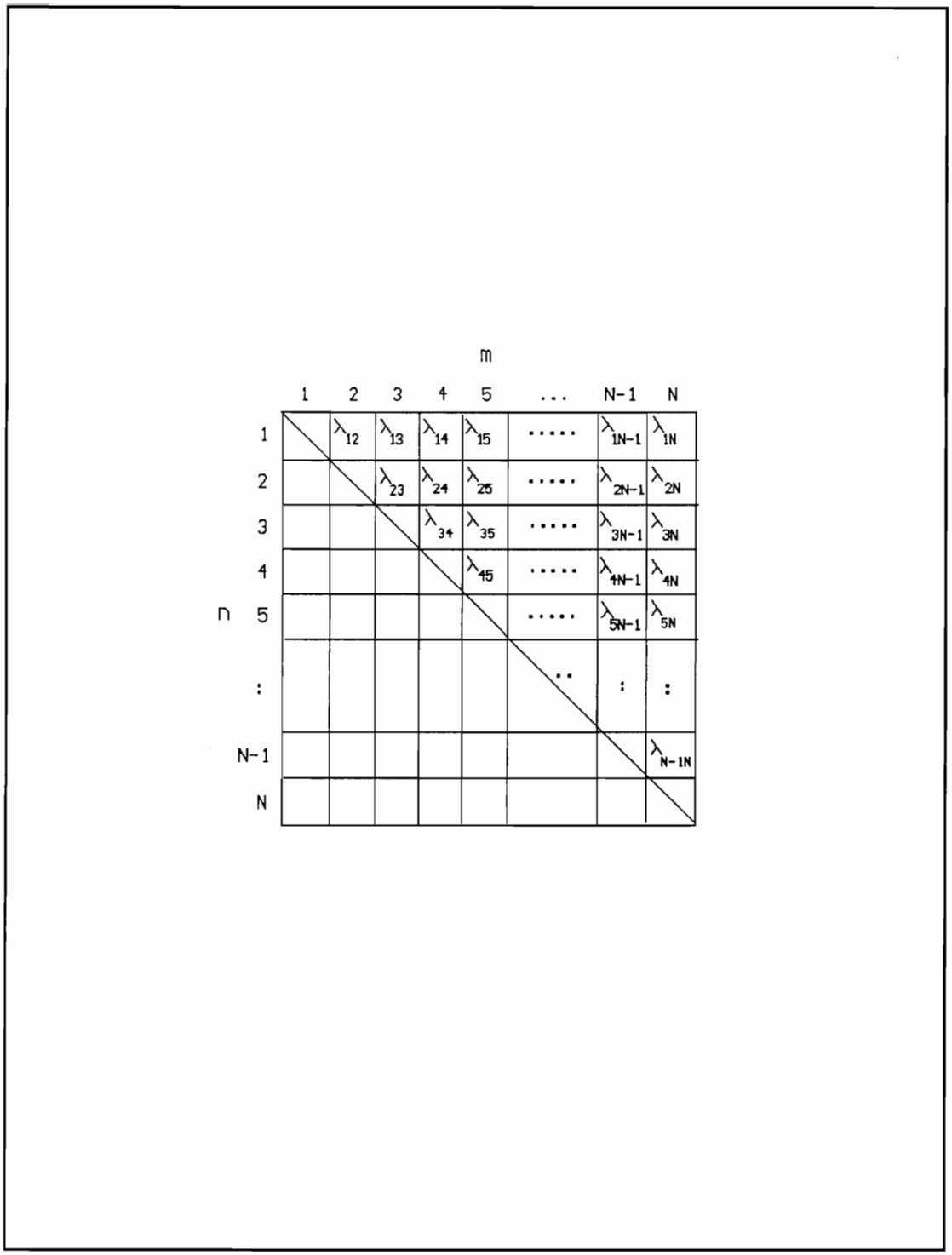

Figure 8 Traffic matrix of bus **A**.

The average arrival rate of packets due to all three priority levels is

$$\lambda_{nm}=\sum_{j=1}^{3}\nu_{nmj} \tag{1}$$

and the total arrival rate of packets, entering the network having source node **n** and having priority level j, is

$$\Lambda_{nj}=\sum_{m=n+1}^{N}\nu_{nmj} \tag{2}$$

Therefore, the total arrival rate of the packets from the external, regardless of priority level, having source node **n** is

$$\Lambda_{n}=\sum_{j=1}^{3}\Lambda_{nj} \tag{3}$$

or it can also be written as

$$\Lambda_{n}=\sum_{m=n+1}^{N}\lambda_{nm} \tag{4}$$

The share of bandwidth is defined as a ratio of the average arrival rate of packets (in bits per second) entering the network over the total channel bandwidth of bus **A**

$$\rho_{n}=\frac{\Lambda_{n}}{\mu C} \tag{5}$$

The summation of all ρ_n in the network is the network load or the traffic intensity, Γ_A

$$\Gamma_A=\sum_{n=1}^{N-1}\rho_n \tag{6}$$

According to an assumption made earlier -- a node is equally likely to communicate with every other node -- the average arrival rates of packets, regardless of the source nodes and destination nodes, are the same,

$$\lambda_{nm}=\lambda \tag{7}$$

From this assumption, Eq. (4) can be rewritten as

$$\Lambda_n=(N-n)\lambda \tag{8}$$

This is just the summation of the columns for each row of the traffic matrix. Using Eqs. (5), (6), and (8), the traffic intensity of bus **A** can be written as

$$\Gamma_A=[\sum_{n=1}^{N-1} t(N-n)]\lambda \tag{9}$$

and the average arrival rate of packets, λ_{nm}, can be easily shown as

$$\lambda=\frac{\Gamma_A}{\sum_{n=1}^{N-1} t(N-n)} \tag{10}$$

As can be seen for any one bus, node **1** (the node closer to the slot generator) has a higher share of bandwidth, while the last node, **N**, on a medium does not have a share of bandwidth. When both directions are evaluated, all nodes have the same share of bandwidth.

— § —

5 DQDB Models Analysis

Probability theories and queueing analyses are used in the evaluation of the performance of the DQDB MAC protocol. The qualitative evaluations of the interaction within the network by each node are done through the stochastic concepts. The M/G/1 and the non-preemptive priority service theories are applied to the analysis of distributed queueing service discipline of the network. For clarity, only the medium access delay on bus **A** will be analyzed.

5.1 DQDB MAC protocol as M/G/1 model

The DQDB network is first modelled with a single priority level. In other words all information packets belong to a same priority level; the DQDB MAC protocol for each node can be depicted as in Figure 9. This is viewed as a single buffer, with assumed infinite capacity, receiving fixed-length packets from an external traffic source. The traffic source is characterized by its average arrival rate , λ_{nm}. Through the control of a REQ_CNT and a CD_CNT, the packets are removed from the buffer by a high-speed medium operating at a channel bit rate of *C* Mbits/second. This is regarded as a single-server for each direction of the network. A packet can only be served when it reaches the serving position (head of the queue).

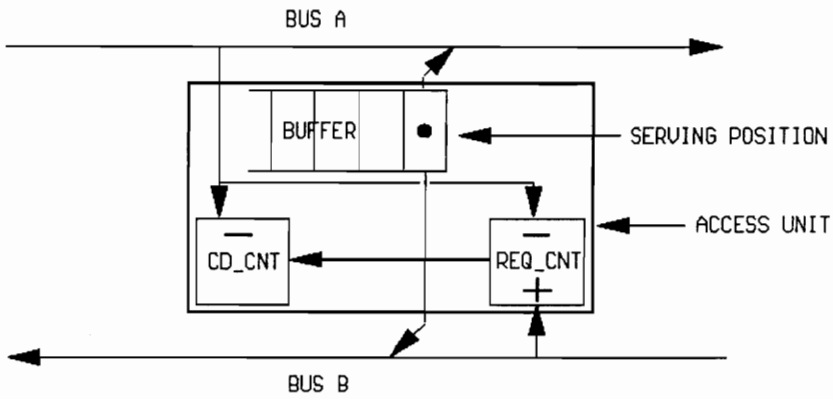

Figure 9 M/G/1 model of a DQDB node.

The DQDB MAC protocol can be thought of as a queueing model having a discrete-time single-server queue with a Poisson arrival process, a general service process, and a service discipline carried out in a FIFO basis. This queueing model is referred to as a M/G/1 model [9][10].

The approximation procedure is used in this paper for the analysis of the M/G/1 queueing model. The arrival process of the network is characterized to be a Markovian input process, which is equivalent to a Poisson distribution. This assumption corresponds very well to the traffic of an actual network. It therefore poses no problem to the modelling. The service process is specified to be general, consequently a probability description of the process must be derived for the medium access delay analysis.

Before proceeding with the formulation of a mathematical description of the general service process, the sources of delay which hinder the flow of packets to the medium as the result of forming a queue must be determined. Figure 7 shows that the *request access delay*, r, and the *data access delay*, d, are responsible for the progress of the packets in the buffer. Therefore the sum of the request access delay and the data access delay represents the *processing time*, Y, for the queue(s). According to the DQDB MAC protocol, a node can send another REQ as soon as the previous queued data packet is transferred to the medium. The propagation delay therefore cannot be a factor in the determination of the processing time. In fact, the processing time equals to the MAC

delay, and so can be written as

$$\overline{Y}=\overline{MAC} \tag{11}$$

Accordingly the moments of the request access delay and the data access delay are derived in order to obtain the mean and the variance of the processing time. The mean and the variance are then used to compute the queueing delay. Finally the sum of the average queueing delay and the average processing time yields the interested average packet medium access delay. They are related mathematically by

$$\overline{T}=\overline{Q}+\overline{Y} \tag{12}$$

All three of the variables in Eq. (12) are random variables, and $\overline{Q}$ depends on $\overline{Y}$.

The request access delay at node **n** arises due to the fact that passing slots on bus **B** were used by the upstream nodes, node **1** through node **n-1**, for sending requests so that node **n** has to wait for an unused request indictor on a slot in order to send a REQ. Once a request indicator is used by a node, it cannot be reused by any other downstream node. Logically the probability description of the request access delay must take the share of the bandwidth used by the upstream nodes into account.

Because time in the medium is divided into discrete time-slots, it would be more convenient and appropriate to represent the request access delay and the data access delay as discrete probability distributions. In effect, the request access delay is theoretically

characterized as a geometric distribution.

$$Pr\{r\}=(1-\beta_n)\beta_n^{r-1} \tag{13}$$

where β_n is

$$\beta_n=\sum_{k=1}^{n-1}\rho_k \tag{14}$$

r, which symbolizes the request access delay, is the integer-valued random variable representing the time interval between the moment the packet arrives at the serving position to the moment its REQ is sent. The time interval is taken to have a unit of a slot.

As stated earlier, the average request access delay and the average data access delay are needed to calculate the average queueing delay and subsequently the average medium access delay. The probability generating function, frequently referred to as the geometric transform G( ), is employed to obtain the moments of the probability distribution Pr{r} (see Appendix A). The first moment or the average is defined as

$$E(r)=\frac{dG(Pr\{r\})}{dz}\Big|_{z=1} \tag{15}$$

After solving for the geometric transform, the first moment emerges as

$$E(r)=\sum_{r=1}^{\infty} r\cdot Pr\{r\}\cdot z^{r-1}\Big|_{z=1} \tag{16}$$

To get the variance of the probability distribution, the second moment has to be

obtained. As usual, the variance is defined as

$$\sigma^2_r = E(r^2) - [E(r)]^2 \tag{17}$$

But when the second derivative is applied to the geometric transform of the geometric distribution the result is

$$\frac{d^2 G(Pr\{r\})}{dz^2}\Big|_{z=1} = E(r^2) - E(r) \tag{18}$$

Due to the exceptional composition of the geometric transformation the expression for the variance of the geometric distribution must be written differently as

$$\sigma^2_r = \frac{d^2 G(Pr\{r\})}{dz^2}\Big|_{z=1} + E(r) - [E(r)]^2 \tag{19}$$

Because of the complexity nature of the network, a geometric probability distribution is fabricated to symbolize the tentative state of the network.

$$Pr\{c\} = (1 - \Gamma_A \Gamma_B)(\Gamma_A \Gamma_B)^{c-1} \tag{20}$$

When $\Gamma_A \to 1$ and $\Gamma_B \to 1$, E(c) increases without bound. Thus, the system becomes unstable, as expected. The dimensionless value *c* represents the state of the network. This distribution is only a function of the traffic intensities of both media. The first moment and the variance of Pr{c} are expressed, respectively, as follows

$$E(c) = \frac{dG(Pr\{c\})}{dz}\Big|_{z=1} \tag{21}$$

and

$$\sigma^2_c = \frac{d^2 G(Pr\{c\})}{dz^2}\Big|_{z=1} + E(c) - [E(c)]^2 \tag{22}$$

Thus, the modified request access delay, $\hat{r}$, can be designated as the previous defined request access delay multiplies with the value representing the state of the network.

$$\hat{r} = r \cdot c \tag{23}$$

It follows that

$$E(\hat{r}) = E(r) \cdot E(c) \tag{24}$$

and

$$\sigma^2_{\hat{r}} = E(r^2) \cdot E(c^2) - [E(r) \cdot E(c)]^2 \tag{25}$$

Data access delay results from the fact that a node has to let as many empty slots pass as the value of the REQ_CNT at the instant the REQ is sent. In other words, as many empty slots as the value of the CD_CNT.

The value in the CD_CNT is the result of two sources. The first is the accumulation of a number of requests from downstream nodes during the interval of data access delay of the previous packet. It should be noted that the REQ_CNT only performs its incremental task during this interval. The number of requests can be related to the total bandwidth used by the downstream nodes. The bandwidth used by the downstream nodes is

$$\frac{1}{\mu C} V_n = \frac{1}{\mu C} \sum_{k=n+1}^{N-1} \Lambda_k \tag{26}$$

The second is the remainder of the number of requests from downstream nodes minus the number of passed empty slots during the interval of request access delay of the packet. During this interval the REQ_CNT performs its task as in the IDLE state. The main emphasis for the number of request analysis is that the number of requests can be related to the ratio of the bandwidth of upstream nodes to the difference of the maximum traffic intensity and the current traffic intensity

$$\frac{\beta_n}{1-\Gamma_A} \tag{27}$$

The data access delay can be best characterized by a conditional Poisson distribution.

$$Pr\{d|r\}=\left(\frac{r\frac{V_n}{\mu C}\beta_n}{1-\Gamma_A}\right)^d \cdot \left[\frac{\exp\left(\frac{-r\frac{V_n}{\mu C}\beta_n}{1-\Gamma_A}\right)}{d!}\right] \tag{28}$$

where d, which symbolizes the data access delay, is an integer-valued random variable representing the data access delay in unit of slots.

The Poisson distribution can be seen as a probability function for a certain number of time-slots which have passed by the referent node. This is before the data packet from the referent node can be written on an empty slot in a particular amount of time r. It can be noted that when the mean E(r) is small, E(d) is small, the variance, σ^2_d, is also small, and vice versa.

The joint distribution can be written as the product of a conditional and marginal distribution

$$Pr\{d,r\}=Pr\{d\,|\,r\}\cdot Pr\{r\} \tag{29}$$

Thus the marginal distribution for d is

$$Pr\{d\}=\sum_{r} Pr\{d\,|\,r\}\cdot Pr\{r\} \tag{30}$$

It then follows that

$$Pr\{d\}=\sum_{r}\left(\frac{r\frac{V_n}{\mu C}\beta_n}{1-\Gamma_A}\right)^d\cdot\left[\frac{\exp\left(\frac{-r\frac{V_n}{\mu C}\beta_n}{1-\Gamma_A}\right)}{d!}\right]\cdot Pr\{r\} \tag{31}$$

To evaluate the first moment and variance of the random variable, d, the derivatives of the geometric transform of the probability distribution, G(Pr{d}) is used. The first moment is

$$E(d)=\frac{dG(Pr\{d\})}{dz}\Big|_{z=1} \tag{32}$$

and the variance is

$$\sigma^2{}_d=\frac{d^2G(Pr\{d\})}{dz^2}\Big|_{z=1}+E(d)-[E(d)]^2 \tag{33}$$

Since $\hat{r}$ and d are random variables, the mean and the variance of the processing time are used to evaluate the average access delay. It can be seen that the processing time is the sum of the request access delay and the data access delay (see Figure 7).

$$Y=\hat{r}+d \tag{34}$$

The related equations which describe the processing time can be written as follows:

Using the summation rule of random variables [11], the mean of the processing time can be written as

$$\bar{Y}=E(\hat{r})+E(d) \tag{35}$$

and its variance is represented as

$$\sigma^2{}_Y=\sigma^2{}_{\hat{r}}+\sigma^2{}_d \tag{36}$$

By applying the mean and the variance of the processing time to the Pollaczek-Khinchine mean-value formula [10], which includes both the packet queueing delay and the processing time, the expression for the average packet medium access delay of a M/G/1 model is written as

$$T=\frac{2\Lambda_n\bar{Y}+\Lambda^2{}_n\sigma^2{}_Y-\Lambda^2{}_n\bar{Y}^2}{2(\Lambda_n-\Lambda^2{}_n\bar{Y})} \tag{37}$$

The average packet medium access delay therefore is given as a function of the average arrival rate of packets, Λ_n; the mean , $\bar{Y}$; and the variance, $\sigma^2{}_Y$, of the processing time. The access delay is given in slots/packet. Λ_n, $\bar{Y}$ and $\sigma^2{}_Y$ are related indirectly to the traffic intensity.

5.2 DQDB MAC protocol as Non-preemptive priority service

Due to the extreme urgency or rapid response time requirements for certain types of information, such as control messages pertaining to the network or higher layers, a network is required to provide multiple priority services. For this reason, the DQDB network is remodelled as a non-preemptive priority queueing service network, which can provide services for three levels of priority. In these multiple priority levels, all nodes are assumed to be permitted to send packets which belong to all priority levels.

In analyzing the DQDB network with multiple priority levels, each node is assumed to operate the distributed queueing service discipline in a non-preemptive priority manner. This means that higher priority packets are not permitted to interrupt the service of lower priority packets. This assumption is reasonable and can be related to the functioning of a real network. The buffer for each priority level is expected to operate its queue in a FIFO basis having infinite capacity.

The basic operation of a DQDB node with three priority levels can be described by a performance model as shown in Figure 10. Each DQDB node maintains the distributed queueing discipline by using the CD_CNT and three REQ_CNTs. Data packets queue in proper buffers according to their priority levels; progress toward the service position is based on the non-preemptive system. The packet at the service

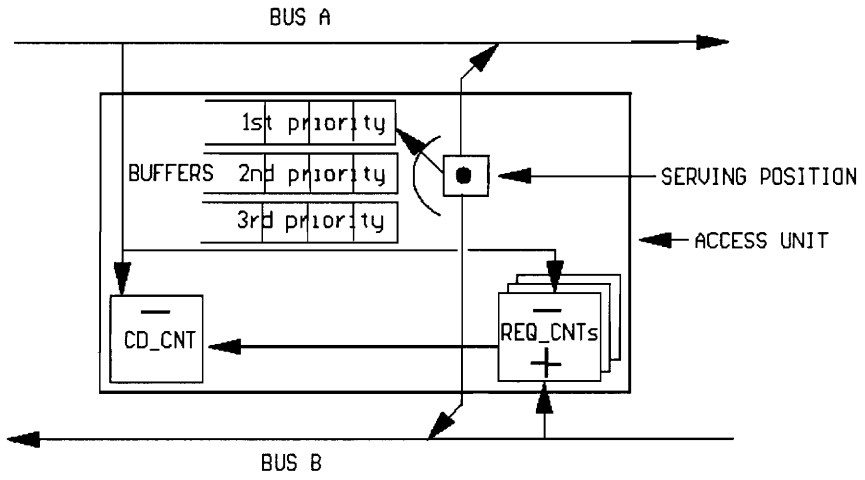

Figure 10 Non-preemptive priority queueing service model.

position (head of the queue) is removed by a single server, which is the high-bandwidth bus **A**.

For simplicity and clarity, the average arrival rate for each priority level is assigned to be a percentage of the overall arrival rate of a node. The percentage is assumed to be the same for every node. It follows that, for the first priority level

$$\Lambda_{n1}=a\cdot\Lambda_n \tag{38}$$

for the second priority level

$$\Lambda_{n2}=b\cdot\Lambda_n \tag{39}$$

and for the third priority level

$$\Lambda_{n3}=c\cdot\Lambda_n \tag{40}$$

where $a<b<c$, and the sum of a,b, and c equals unity.

As in the M/G/1 model, the sources of delay for the non-preemptive priority service are also the request access delay and the data access delay. When a packet, regardless of priority, arrives at the service position, the node will send a REQ on the first available request indicator on a slot. Clearly the request access delay is independent of the priority level. The probability distribution Pr{r} describing the request access delay, which was derived in section 5.1, can be applied to the non-preemptive priority model as well; with one exception that the bandwidth of all priority levels must be taken into account. Then Pr{r} is given as

$$Pr\{r\} = (1 - \sum_{j=1}^{3} \beta_{nj})(\sum_{j=1}^{3} \beta_{nj})^{r-1} \tag{41}$$

As mention earlier, when the state of the network is taken into account, the modified request access delay, $\hat{r}$, is written as

$$\hat{r} = r \cdot c \tag{42}$$

It follows that

$$E(\hat{r}) = E(r) \cdot E(c) \tag{43}$$

and the second moment is

$$E(\hat{r}^2) = E(r^2) \cdot E(c^2) \tag{44}$$

The data access delay, on the other hand, is deeply dependent on the priority level of the packet being served. The servicing of higher priority packets takes precedent. Therefore, lower priority packets queueing at all nodes in the network have to give up the empty slots for the higher priority packets, even though the lower priority packets were queueing before the higher priority packets.

The non-preemptive service discipline is strictly maintained by the REQ_CNTs. It should be noted that when a REQ_i passes by a node, only the REQ_CNTs with priority i and lower increase by one. As the result, the higher priority packets are served before the lower priority packets within the node environment (in the queueing buffers) and also

within the network environment. In effect, Pr{d} must be rewritten to include the priority-dependent factor of the network environment for each priority level. It follows that

$$[Pr\{d\}]_j=\sum_r \left(\frac{r\frac{\Phi_j}{\mu C}B_j}{1-\Gamma_A}\right)^d \cdot \left[\frac{\exp\left(\frac{-r\frac{\Phi_j}{\mu C}B_j}{1-\Gamma_A}\right)}{d!}\right] \cdot Pr\{r\} \tag{45}$$

where

$$\Phi_j=\sum_{i=1}^{j} V_{ni} \tag{46}$$

and

$$B_j=\sum_{i=1}^{j} \beta_{ni} \tag{47}$$

The corresponding mean and the second moment of the probability distribution, $E(d)_j$ and $E(d^2)_j$, respectively, can be shown as

$$E(d)_j=\frac{dG([Pr\{d\}]_j)}{dz}\Big|_{z=1} \tag{48}$$

and

$$E(d^2)_j=\frac{d^2G([Pr\{d\}]_j)}{dz^2}\Big|_{z=1}+E(d)_j \tag{49}$$

The processing time of each queue equals to the request access delay and the data access delay

$$Y_j=\hat{r}+d_j \tag{50}$$

Then the first moment and the second moment of the processing time for each priority level, $\overline{Y}_j$ and $\overline{Y^2}_j$, respectively, can be calculated as follow

$$\overline{Y}_j=E(\hat{r})+E(d)_j \tag{51}$$

and

$$\overline{Y^2}_j=E(\hat{r}^2)+E(d^2)_j \tag{52}$$

where $E(\hat{r}^2)$ is the second moments of the modified request access delay.

$$E(\hat{r}^2)=E(r^2)\cdot E(c^2) \tag{53}$$

Within the node environment, the multiple priority packets are served in a non-preemptive priority fashion. By applying the mean and the second moment of the processing time, the average packet queueing delay for each priority level [9][10] can be obtained as

$$\overline{Q}_j=\frac{E(D_d)}{(1-\sum_{i=1}^{j}\rho_i)(1-\sum_{i=1}^{j-1}\rho_i)} \tag{54}$$

where

$$E(D_d)=\sum_{j=1}^{3}\frac{\tau\Lambda_{nj}\overline{Y^2}_j}{2} \tag{55}$$

and

$$\rho_i=\tau\Lambda_{ni}\overline{Y}_i \tag{56}$$

The τ is included in Eq. (55) and (56) to make the results dimensionally correct; because

the mean of the processing time is given in slots/packet and the second moment is given in slots^2/packet^2. The average queueing delay analysis is presented in Appendix B. Finally, the average packet medium access delay for each priority level can be obtained as follow

$$T_j=\overline{Q}_j+\overline{Y}_j \tag{57}$$

The average packet medium access delay is the sum of the average queueing delay and the average processing time.

The average medium access delays is given in slots/packet, and the result must be multiplied with τ to convert to seconds/packet.

— § —

6 Performance Results

In this chapter, the mathematic models -- M/G/1 model and non-preemptive priority model -- are evaluated for a single priority level network and a three priority level network respectively. The models were applied to a metropolitan area communication network with the following parameters:

- A network consisting of 50 nodes equally positioned along two opposite unidirectional buses.
- No subscribers requested isochronous service. Thus, the total bandwidth is dedicated to non-isochronous traffic.
- Bandwidth of each bus is 150 Mbps.
- Slot size is 69 bytes.
- Access technique is DQDB MAC protocol.

The experiment network was developed using C programming language. The computer program source code is laid out in Appendix C. The data were output to a file and then plotted.

6.1 M/G/1 -- Single priority level

Some of the recommended performance measures have been suggested by [5] for a communication system. These measures are the MAC delay and the medium access delay. The analysis of MAC delay to evaluate the fairness of a network is the key performance measure [5]. The equation for the average of $\overline{MAC}$ delay can be expressed as

$$<\overline{MAC}> = \frac{1}{N-1}\sum_{n=1}^{N-1} \overline{MAC_n} \tag{58}$$

where $\overline{MAC_n}$ is the average MAC delay of node **n**. The fairness of a network is evaluated based on the difference between $\overline{MAC_n}$ and $<\overline{MAC}>$. If the differences for all nodes are not so large when the network is operated at a certain traffic intensity, then the network can be considered fair. The above equation is also applicable to the medium access delay.

The results obtained from the application of the M/G/1 model to a MAN with the stated specifications are used to evaluate the fairness of the DQDB network (see Figures 11 and 12).

As shown in Figure 11, it can be seen that a DQDB network is not a very fair

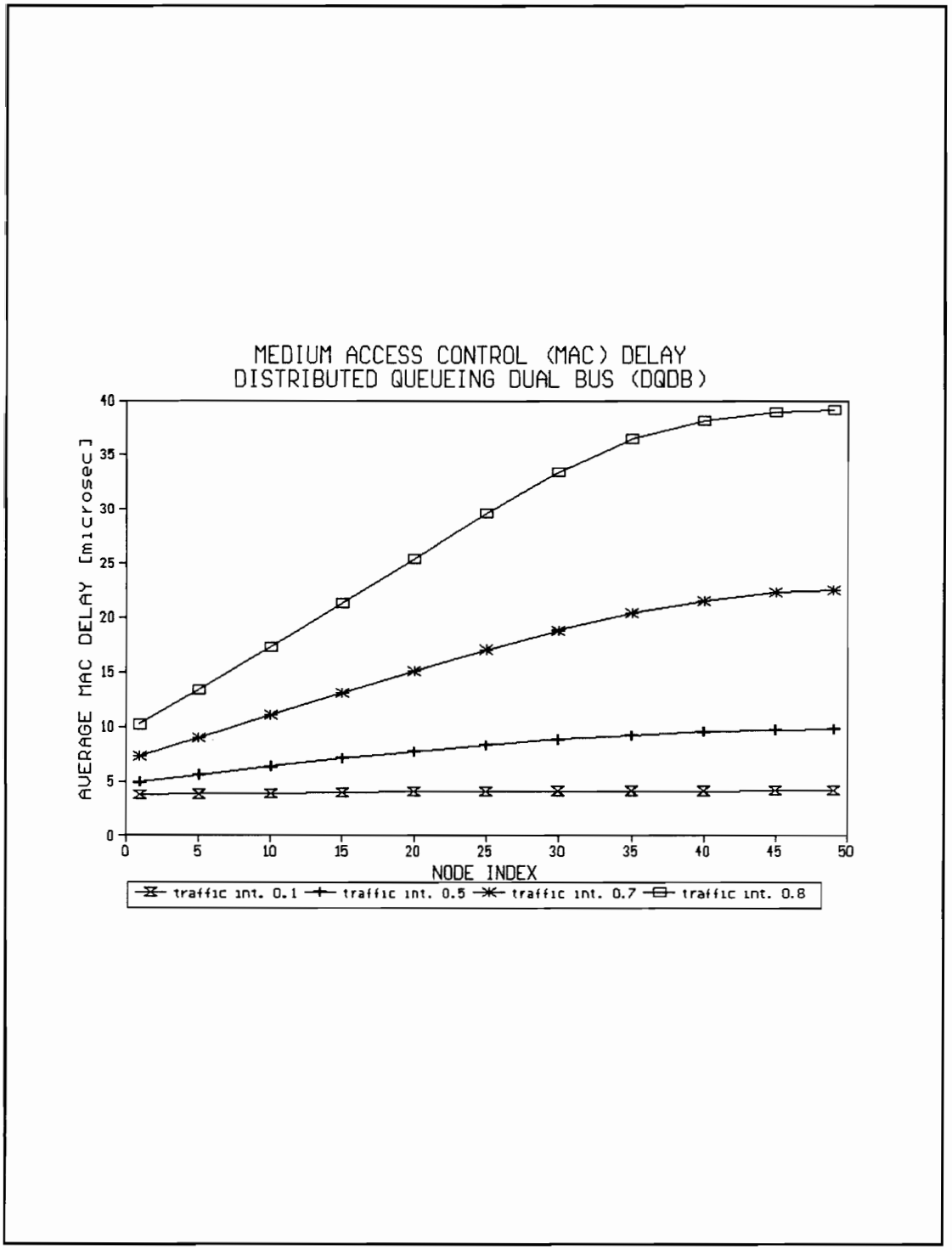

Figure 11 Average MAC dealy.

network at high traffic intensities. The graph represents the average MAC delay on bus **A** of a DQDB node. The network becomes more unfair as the traffic intensity of the network escalates. However the fairness improves when the information volume is small and the traffic intensity is low.

Figure 12 shows the average medium access delay as a function of the node index on bus **A**. The average access delay is strongly node-dependent. At traffic intensity 0.8, a maximum access delay can be found around node **40**; the average access delays of the subsequent nodes incline to decrease slightly. These findings were also observed by [2].

When the total average access delay of both media **A** and **B** is evaluated (see Figure 13), it can be observed that nodes in the middle of the network experience the highest delays, as have been previously reported by other papers [2][3]. This graph represents the delay of a DQDB network at traffic intensity 0.5.

Figure 14 shows the effect of different traffic intensities on bus **A** at each node. Again it can be noticed that the network fairness improves at low traffic intensities. On any one bus, node **1** (the node closest to the slot generator) has the lowest overall delay. However, node **35** has the highest overall delay, which was also observed by [2]. The graphs are dimensionally similar as the curve shown in [1].

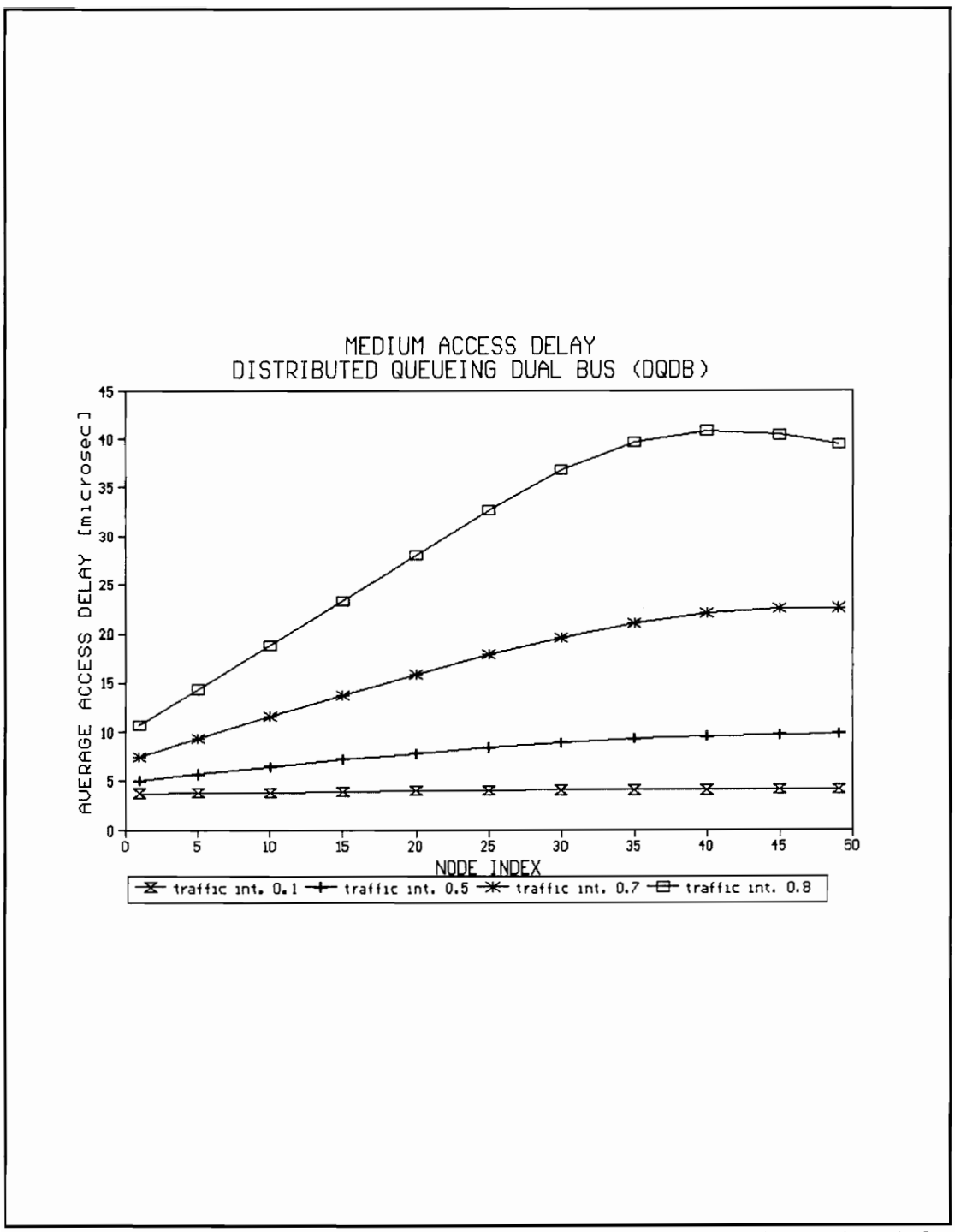

Figure 12 Average medium access delay versus node index. Delay on bus A for a one priority level DQDB network.

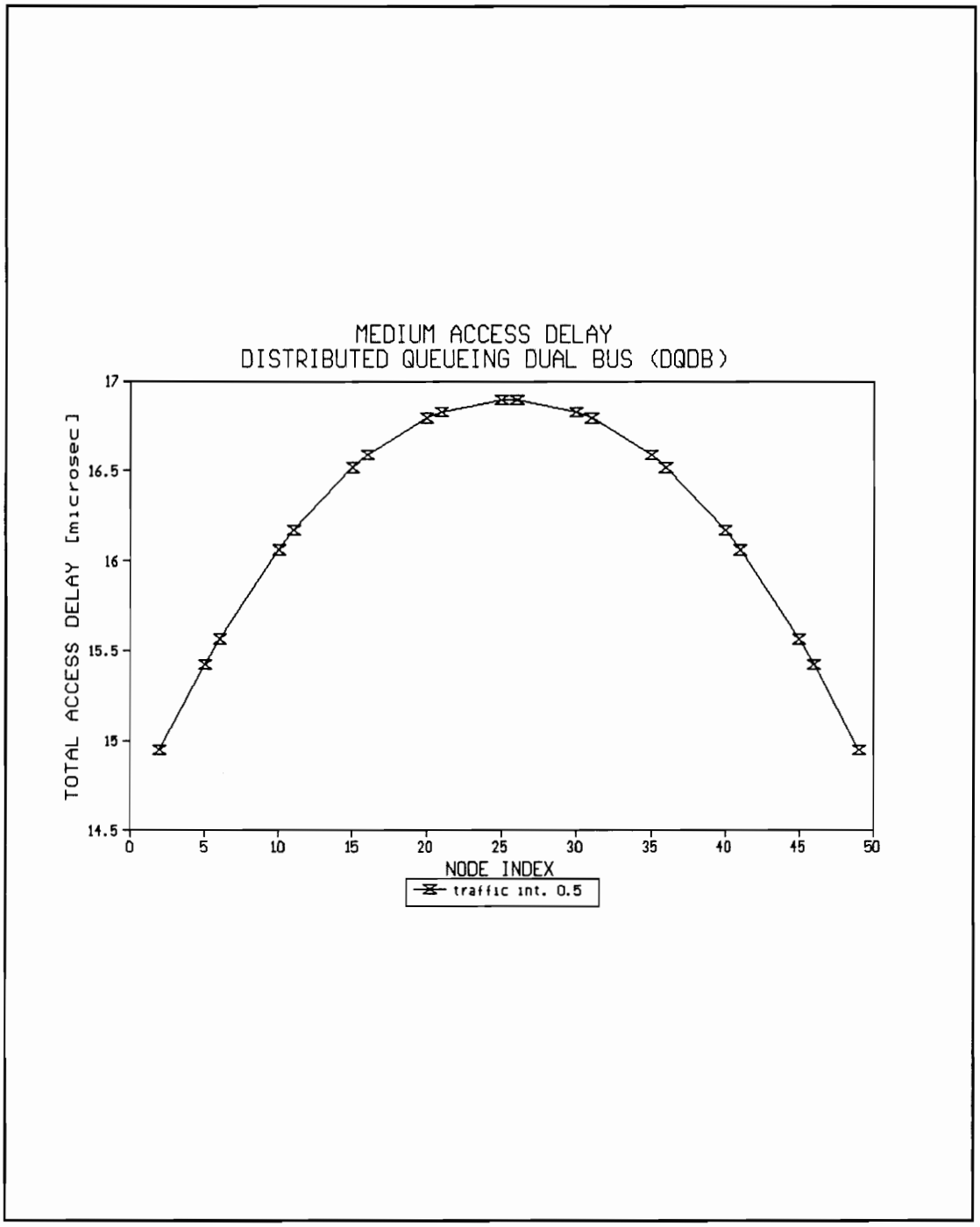

Figure 13 Total medium access delay versus node index. The graph represents the sum of the average medium access delay on bus **A** and **B**. Nodes in the middle experience the highest delay time.

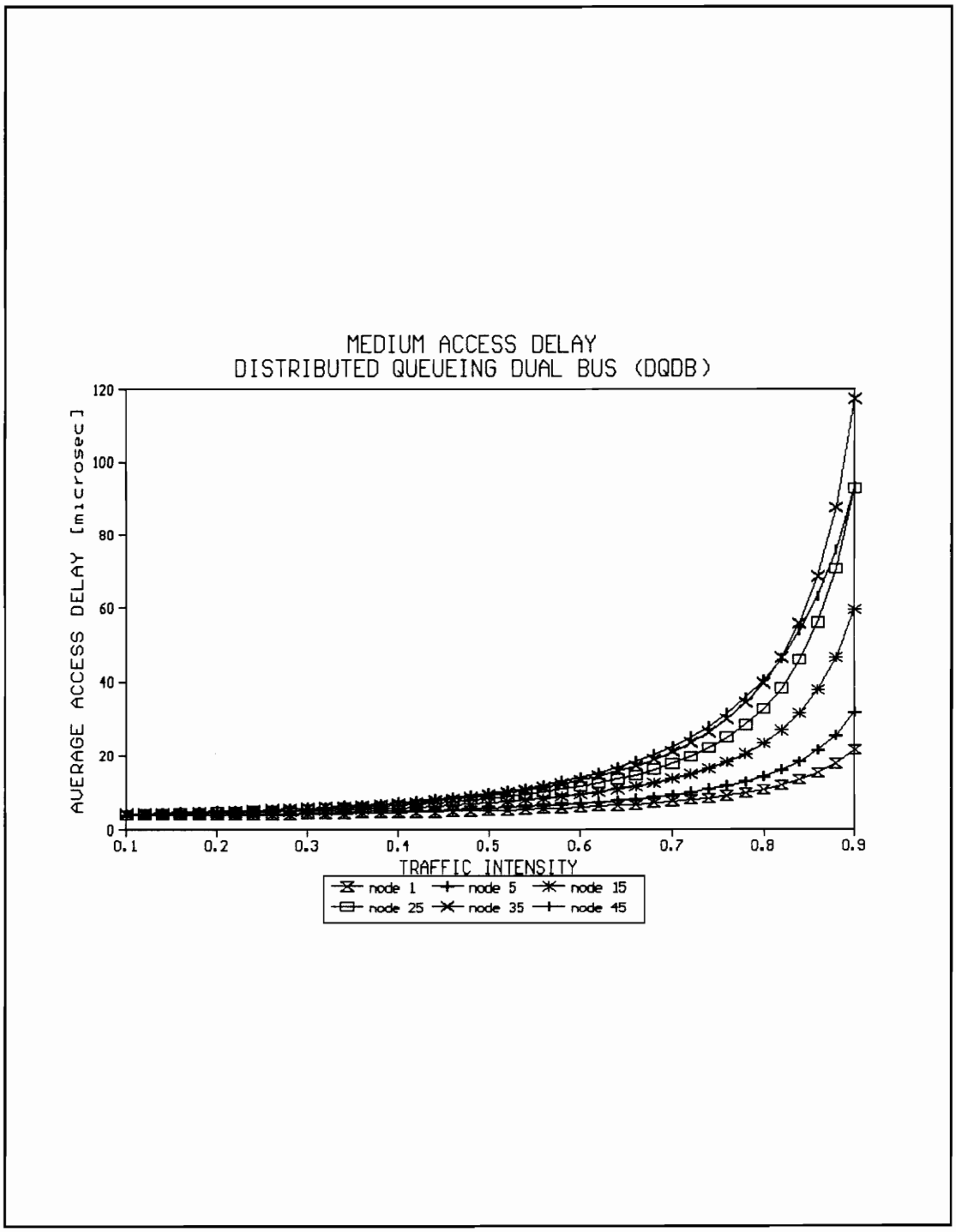

Figure 14 Average access delay versus traffic intensity. Node 35 has the highest delay time.

To illustrate the effect of slot size on a DQDB network, the slot size was reduced but other parameters of a network were kept the same. Figure 15 shows the relationship between the slot size and network fairness. As the slot size decreases; the DQDB fairness increases. The average packet medium access delay on the other hand decreases due to the smaller size of a packet. However, the communication messages must now be segmented into additional smaller packets. In this case, the delay for long messages increase. Therefore, an optimum slot size must be selected to minimize both the message and packet delays.

6.2 Non-preemptive priority service -- Three priority level

A network with three priority levels is now analyzed. For the purpose of numerical presentation, the traffic of each priority level is assigned as a fraction of the average packet arrival rate Λ_n: for first priority, a=0.1; second priority, b=0.3; and third priority, c=0.6, as have been expressed in Eqs. (38), (39) and (40). Figure 16 shows the effect of multiple priorities at selected nodes. The nodes in the middle of the network have the most *benefit*[7], thus the priority levels of the middle nodes have wide differences in the access delays. On the other hand, regardless of priority level, the

[7] The *benefit* here is referred to the existence of differences in the access delay between priority levels. There would be no *benefit* if the access delays of all priority levels are the same.

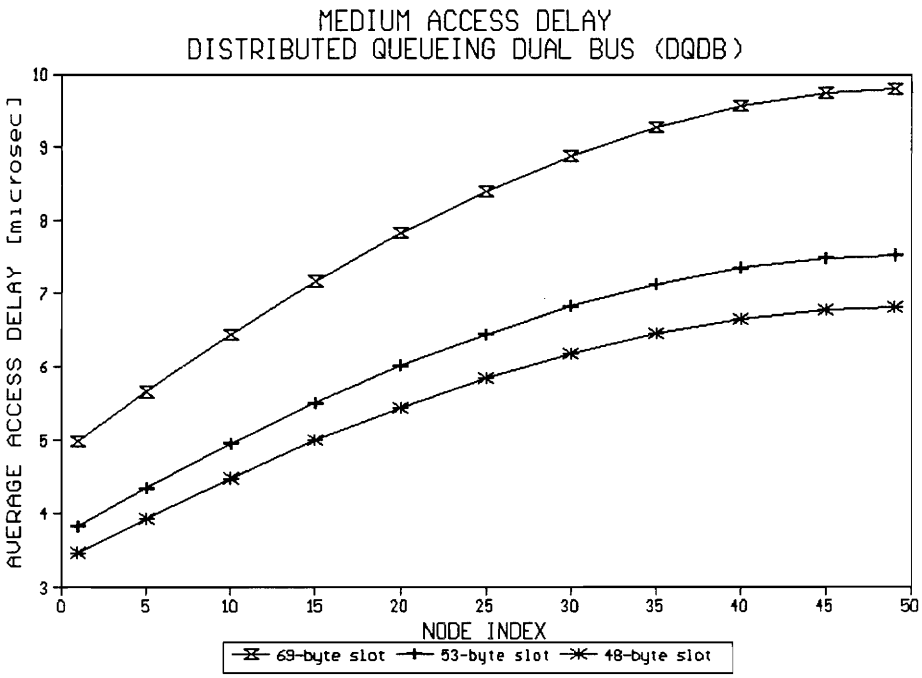

Figure 15 Relation of slot size with the network fairness.

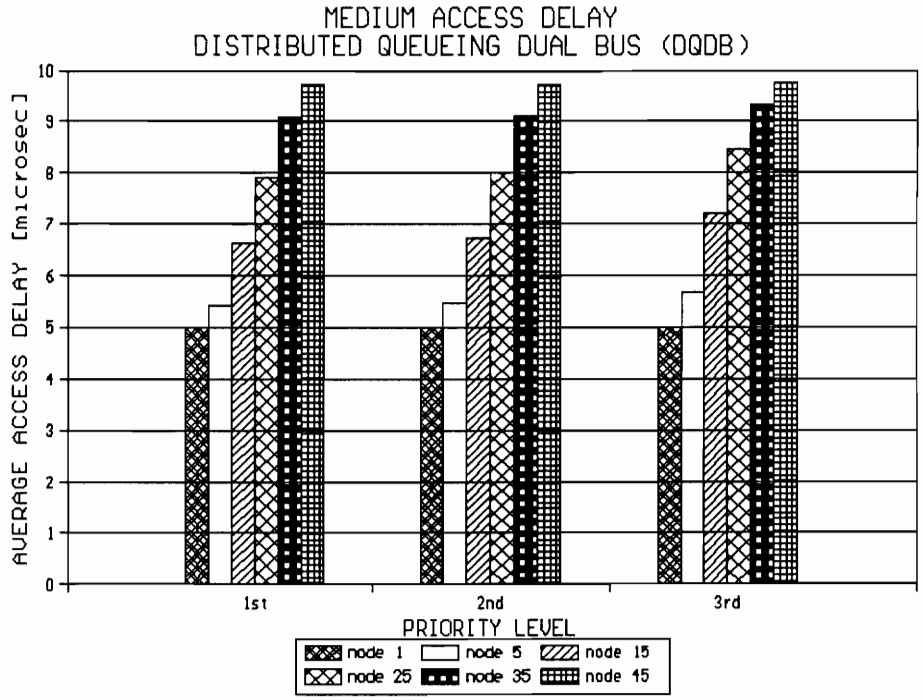

Figure 16 Average access delay versus priority level.

access delay time of the nodes at the ends of the network is approximately the same.

Figures 17 and 18 display the network delays of multiple priority levels at traffic intensity 0.5 and 0.8, respectively. Again it can be seen that the nodes in the middle of the network have the most *benefit*. Therefore, it can be concluded that the *benefit* is also a factor in the evaluation of the fairness of a network.

At a low traffic intensity, the network does not have as much *benefit* as at a higher traffic intensity (see Figure 19). At low traffic intensity, there would be more empty slots available on the media so that a data packet of any priority level will be readily granted an empty slot. At a high traffic intensity, the number of empty slots are limited. The non-preemptive priority service discipline strictly governs the MAC, and a rising *benefit* between each priority level is observed.

Figure 20 shows the effect of slot size on a network with three priority levels. As the slot size increases, the differences in the access delay between priority levels become wider. Similar result was observed by [4].

— § —

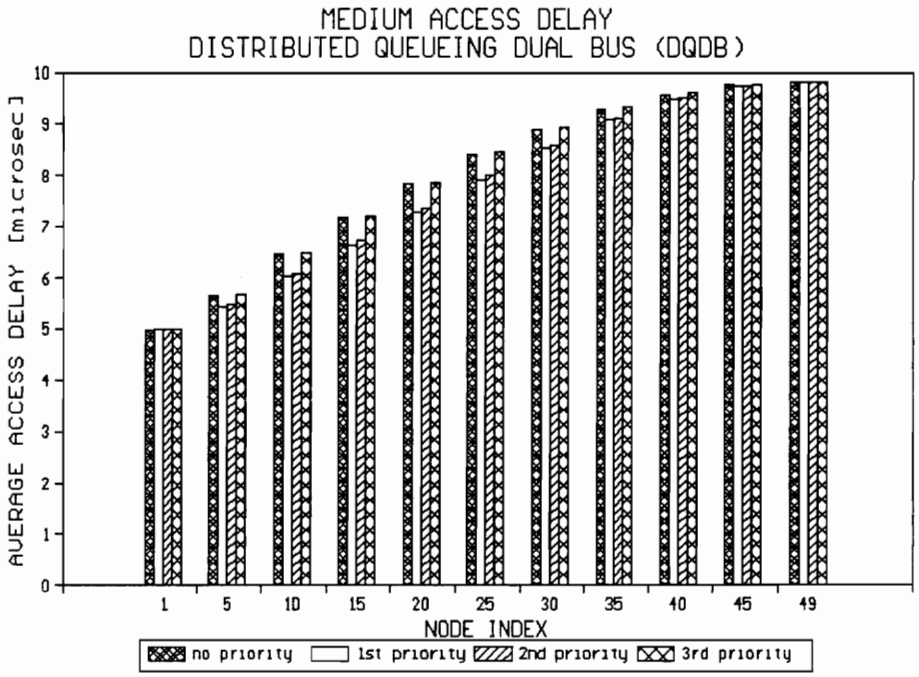

Figure 17 Priority levels at traffic intensity 0.5.

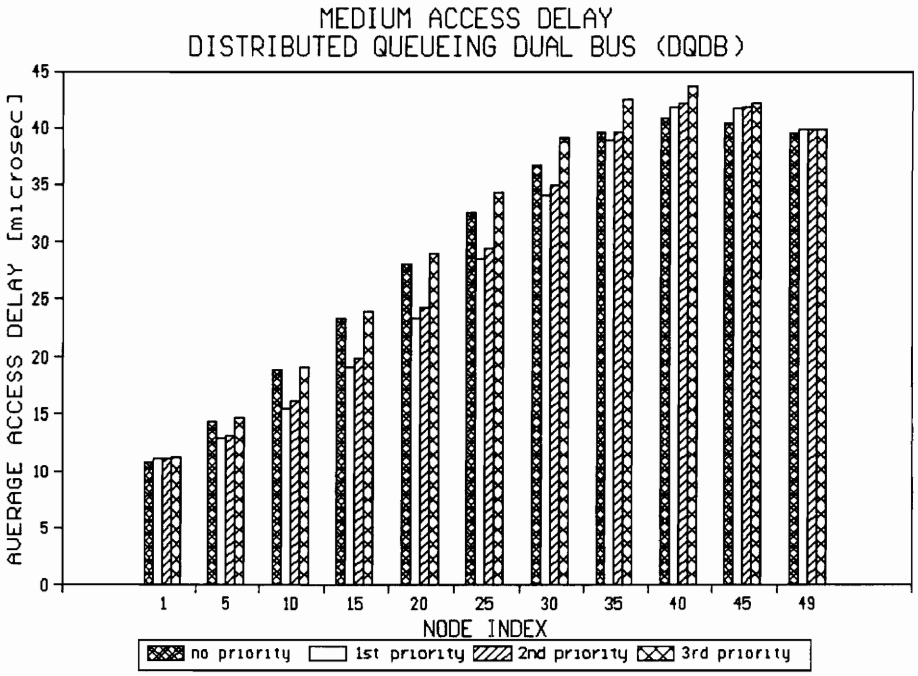

Figure 18 Priority levels at traffic intensity 0.8.

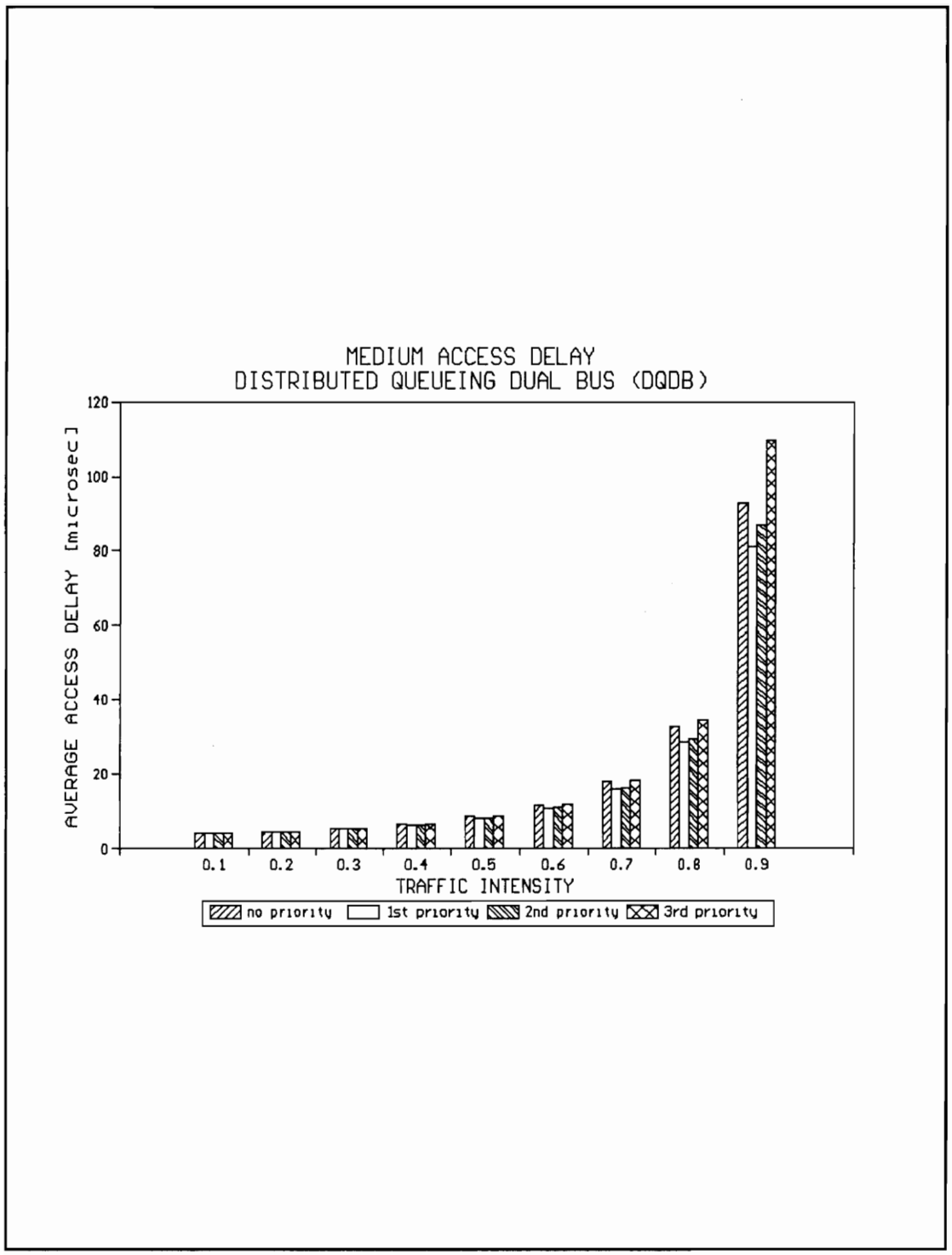

Figure 19 Priority levels at different traffic intensity.

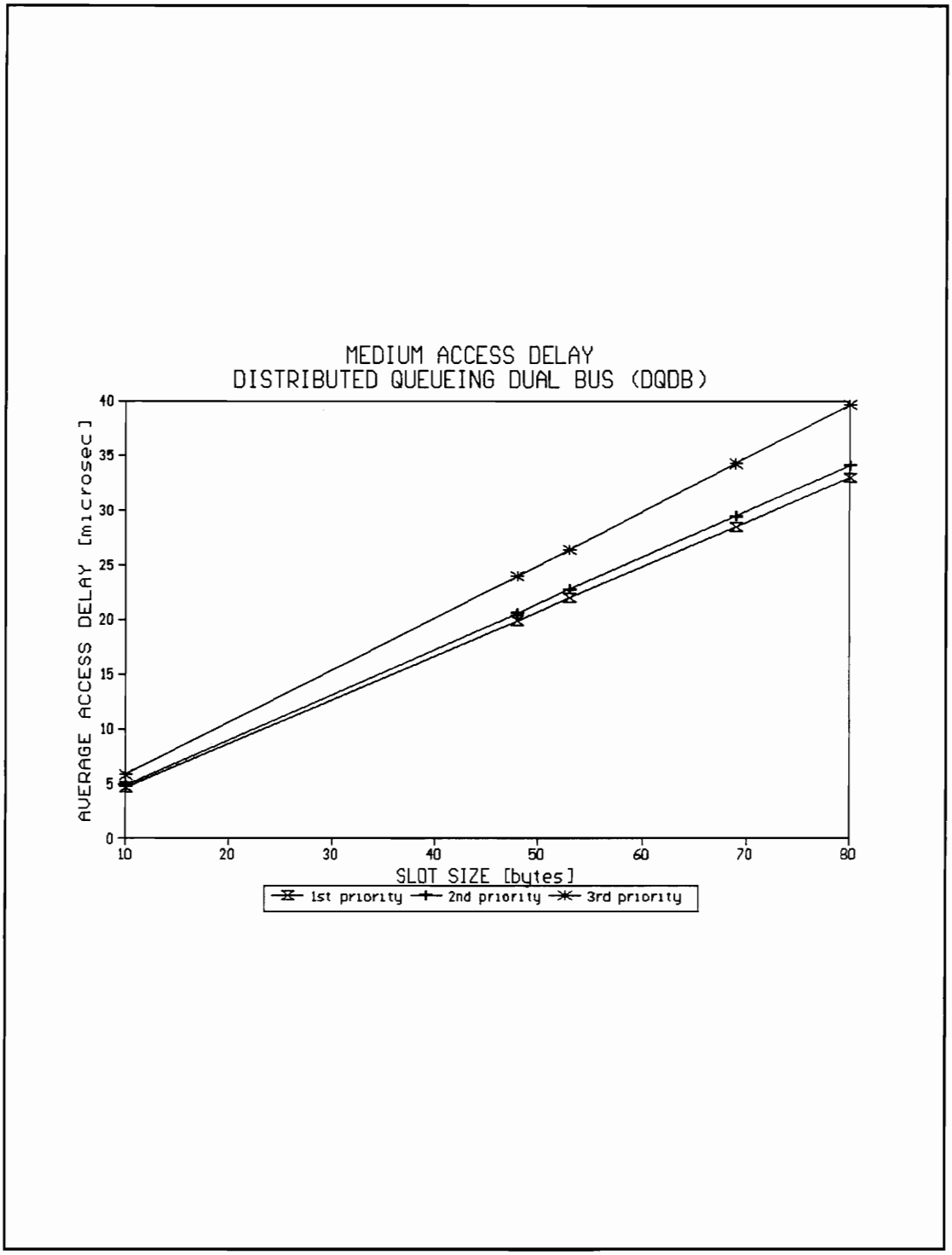

Figure 20 Average medium access delay versus slot size.

7 Conclusion

Two mathematical models which characterize the performance of DQDB MAC protocol with a single priority and with three priority levels have been obtained in this paper. The models were then applied to a 50-node-MAN operated at different traffic intensity. The experiment results have provided a quantitative view of the performance of a DQDB network.

The results of the network with single priority level are dimensionally comparable with other observed results. The medium access delay of a packet depends on the order of the node within the network; thus, creating an unfairness associated with DQDB.

Two important and new characteristics of DQDB network that have been observed in this paper are:

- When the DQDB network model performs in a multiple level environment, the *benefit* of multiple priority levels service increases as the traffic intensity increases.

- The *benefit* of priority levels service also depends on the order of the node

within the network. At high traffic intensity, the *benefit* of middle nodes is higher than that of the nodes situated at both ends of the network. This latter finding of the effect of non-preemptive priority service on a DQDB network is an additional unfairness, which has not been considered before.

There have been many methods aimed to reduce the unfairness of a network, such as bandwidth balancing (BWB) or cyclic request control (CYREC) [6]. Besides these methods, another alternate avenue is suggested. This could be achieved by controlling the aging times of the request access delays for lower priority level packets at the least favorable nodes (nodes in the middle of the network). As an example, all data packets are initially in the third priority level. The aging times of the data packets belonging to the nodes in the middle of a network are then set to certain levels, so that the data packets can permissively be promoted to second priority level. By reducing the differences between the access delays of the nodes, the unfairness of the network will be reduced. An investigation of this alternative method is being examined by the author.

Despite the unfairness which have been observed, the DQDB MAC protocol can be considered a technologically suitable method for interconnection of LANs, PBXs, high-speed office communication networks, and even video conferencing. A DQDB network possesses all the dynamic characteristics suitable for a MAN: such as the ability to handle integrated voice/data traffic, large geographic coverage, utilizing

high-bandwidth media efficiently, and a high reliability. These characteristics are also the conditions which are required by the IEEE committee for a MAN.

The DQDB can operate over a number of existing physical medium dependent (PMD) sublayers such as FDDI, the CCITTG.703 hierarchical interfaces, and SONET [12]. The average packet medium access delay of a DQDB network is well within the desired limit of one (1) millisecond, which is the acceptable access delay required for today's broadband services [6]. Moreover the DQDB's medium access delay is independent of the network size, which makes the DQDB network suitable for a expandable metropolitan area network or wide area network.

— § —

References

[1] R.M. Newman, Z.L. Budrikis and J.L. Hullett, *The QPSX MAN*, IEEE Comm. Mag., v 26 (April 1988) 20-28.

[2] P. Tran-Gia and T. Stock, *Approximate Performance Analysis of the DQDB Access Protocol*, Computer Networks and ISDN Systems, v 20 (1990) 231-240.

[3] K. Sauer and W. Schodl, *Performance Aspects of the DQDB Protocol*, v 20 (1990) 253-260.

[4] P.G. Potter and M. Zukerman, *A Discrete Shared Processor Model for DQDB*, Computer Network and ISDN Systems, v 20 (1990) 217-222.

[5] Dr. F.J. Ricci, *Performance Measures for DQDB Network Simulation*, (1991).

[6] M. Conti, E. Gregori and L. Lenzini, *A Methodological Approach to an Extensive Analysis of DQDB Performance and Fairness*, IEEE Journal on Selected Areas in Comm., v 9, no. 1 (January 1991) 76-87.

[7] M. Zukerman and P.G. Potter, *The DQDB Protocol and its Performance under Overload Traffic Conditions*, v 20 (1990) 261-270.

[8] IEEE 802.6 Working Group, *Proposed IEEE Standard 802.6 Distributed Queueing Dual Bus MAN*, Draft May, (1990).

[9] M. Schwartz, *Telecommunication Networks Protocols, Modeling and Analysis*, Addison-Wesley Publishing Co., (1988).

[10] J.L. Hammond and P.J. O'Reilly, *Performance Analysis of Local Computer Networks*, Addison-Wesley Publishing Co., (1986).

[11] W.C. Giffin, *Transform Techniques for Probability Modeling*, Academic Press Inc., (1975).

[12] R.J. Hoss, *Fiber Optics Communication Design Handbook*, Prentice Hall, (1990).

Appendix A Geometric Transform

The method of geometric transform is a very practical technique in probability theory. Through the use of geometric transform, the moments of a discrete distribution function can be obtained indirectly in a far less complicated fashion than by a direct computation [11]. The geometric transform of a function of x is defined as

$$G(f(x),z)=\sum_{x=-\infty}^{\infty} f(x)z^x \tag{A1}$$

where z is complex transform variable.

Now let x be a discrete random variable, which can only be a non-negative integer. Pr(x) is a probability distribution function with the parameter x. Then according to the definition above, the geometric transform of Pr(x) is

$$G(Pr\{x\},z)=\sum_{x=0}^{\infty} Pr\{x\}z^x \tag{A2}$$

So when the transformation is evaluated at z=1, it follows that

$$G(Pr\{x\},z)\big|_{z=1}=\sum_{x=0}^{\infty} Pr\{x\}=1 \tag{A3}$$

which is a distinct property of a probability distribution function; the total area under the curve is unity.

To get the first moment or the mean of the probability distribution, E(x), the first derivative to the geometric transform, G(Pr{x},z) ia applied and evaluated at z=1. It then follows

$$\frac{dG(Pr\{x\},z)}{dz}\Big|_{z=1}=\sum_{x=0}^{\infty} xPr\{x\}=E(x) \tag{A4}$$

which precisely is the definition of the mean of a probability distribution.

The second derivative of G(Pr{x},z) evaluated at z=1 gives

$$\frac{d^2G(Pr\{x\},z)}{dz^2}\Big|_{z=1}=\sum_{x=0}^{\infty} x(x-1)Pr\{x\} \tag{A5}$$

but this is

$$\sum_{x=0}^{\infty} x(x-1)Pr\{x\}=\sum_{x=0}^{\infty} (x^2-x)Pr\{x\}=E(x^2)-E(x) \tag{A6}$$

The variance of a probability distribution is defined as

$$\sigma^2{}_x=E(x^2)-[E(x)]^2 \tag{A7}$$

Therefore the variance is given as follow

$$\sigma^2{}_x=\frac{d^2G(Pr\{x\},z)}{dz^2}\Big|_{z=1}+E(x)-[E(x)]^2 \tag{A8}$$

— § —

Appendix B Mean Value Analysis

In this appendix, the well known results in queueing theory are extensively used to demonstrate the analysis of the queueing delay of a information packet in a non-preemptive priority service discipline. For more information on non-preemptive priority service, see [9] and [10].

In a non-preemptive priority service discipline, the average queueing delay, $\overline{Q}_j$, of a packet of priority level j is a result of three different delays:

- D_{af} - the delay due to the services of any packets of priority level higher than level j which arrive at the buffers after the tagged packet.
- D_{bf} - the delay due to the services of any packets of priority level j or higher which are already in the buffers before the tagged packet arrives to the buffer.
- D_d - the delay due to the service of a packet being served when the tagged packet arrives at the buffer.

thus it follows that

$$\overline{Q}_j = E(D_{af})_j + E(D_{bf})_j + E(D_d) \tag{B1}$$

Each of the delays can be represented as follows (see [9] for the derivations):

$$E(D_{ap})_j=\sum_{i=1}^{j-1}\rho_i\overline{Q}_j \tag{B2}$$

and

$$E(D_{bp})_j=\sum_{i=1}^{j}\rho_i\overline{Q}_i \tag{B3}$$

and

$$E(D_d)=\sum_{j=1}^{3}\frac{\tau\Lambda_{nj}\overline{Y^2}_j}{2} \tag{B4}$$

where

$$\rho_i=\tau\Lambda_{ni}\overline{Y}_i \tag{B5}$$

By substituting Eq. (B2) and (B3) into Eq. (B1) and solving for $\overline{Q}_j$, the following is obtained

$$\overline{Q}_j=\frac{\sum_{i=1}^{j}\rho_i\overline{Q}_i+E(D_d)}{1-\sum_{i=1}^{j-1}\rho_i} \tag{B6}$$

After iterative computation for the queueing delay, $\overline{Q}_i$, of each priority level in Eq. (B6), the average queueing delay of non-preemptive service discipline can be rewritten as

$$\overline{Q}_j=\frac{E(D_d)}{(1-\sum_{i=1}^{j}\rho_i)(1-\sum_{i=1}^{j-1}\rho_i)} \tag{B7}$$

— § —

Appendix C Computer Programs

This appendix lists the computer program source code used to evaluate the average packet medium access delay for both models, M/G/1 and non-preemptive priority service. The codes were edited with the Ed program in the UNIX environment, and were compiled with the UNIX C compiler and the math library.

— § —

Computer Program for M/G/1 Queueing

```
/*************************************************************************/
/*                              DON H. PHAM                              */
/* MEDIUM ACCESS DELAY EVALUATION FOR DISTRIBUTED QUEUEING               */
/*                    DUAL BUS (DQDB) MAC PROTOCOL                       */
/*                              31 Oct 1991                              */
/*************************************************************************/

/* NO PART OF THIS PROGRAM CAN BE REPRODUCED BY ANY MEAN                 */
/*       WITHOUT THE CONSENT OF THE PROGRAMMER                           */

#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>

/*********************************************************/

float
st(void)
{       float slot_time,slot_length,channel_cp;
        channel_cp=(float)150.0e6;
        slot_length=(float)(69*8);
        slot_time=slot_length/channel_cp;
        return(slot_time);
}

/*********************************************************/

float
d(void)
{       float pack_time;
        pack_time=(float)(64*8)/(float)150e6;
        return(pack_time);
}

/*********************************************************/

float
L(float trafint)
{       float l;
        l=trafint/(d()*(float)1.225e+3);
```

```
        return (l);
}

/*********************************************************/

float
A(int node, float trafint)
{    float a=(float)0.0,l;
        int z;
        l=L(trafint);
        for(z=node+1; z<=50; ++z)
        {       a=a+l;}
        return (a);
}

/*********************************************************/


float
W(int node, float trafint)
{    float w=(float)0.0;
        int z;
        if(node==49)
                w=(float)0.0;
        else
                for(z=node+1; z<=49; ++z)
                {       w=w+A(z,trafint);}
        return (w);
}

/*********************************************************/

float
R(int node, float trafint)
{       float r;
        r=A(node,trafint)*d();
        return (r);
}

/*********************************************************/
```

```
float
Q(int node, float trafint)
{       int z;
        float q=(float)0.0;
        if(node==1)
                q=(float)0.0;
        else
                for(z=1; z<node; ++z)
                        {q=q+R(z,trafint);}
        return (q);
}

/*********************************************************/

float
P(int k, int node, float trafint)
{       float p,a,b;
        a=Q(node,trafint);
        if(k==1)
                b=(float)1.0;
        else
                b=(float)pow((double)a, (double)(k-1));
        p=b*((float)1.0-a);
        return (p);
}

/*********************************************************/

float
F(int x, int node, float trafint)
{       float a,b,c,g;
        float xfact=(float)1.0;
        float f=(float)0.0;
        int k,v;
        for(v=1; v<=x; ++v)
        {       xfact=xfact*(float)v;}
        g=W(node,trafint)*d()*Q(node,trafint)/(1.0-trafint);
        for(k=1; k<=10; ++k)
        {       a=(float)k*g;
                if (x==0)
                        b=(float)1.0;
```

```
                else
                        b=(float)pow((double)a, (double)x);
                c=(float)exp((double)(-a));
                f=f+(b*c*P(k,node,trafint)/xfact);
        }
        return (f);
}

/*************************************************************/

float
Y(int u, float trafint)
{       float y,a,b;
        a=trafint*trafint;
        if(u==1)
                b=(float)1.0;
        else
                b=(float)pow((double)a, (double)(u-1));
        y=b*((float)1.0-a);
        return (y);
}

/*************************************************************/

float
MY(float trafint)
{       int u;
        float my=(float)0.0;
        for(u=1;u<=30;++u)
        {my=my+((float)u*Y(u,trafint));}
        return (my);
}

/*************************************************************/

float
EY2(float trafint)
{       float ey2,a,b;
        float z=(float)0.0;
        int u;
        a=MY(trafint);
```

```
        for(u=1;u<=30;++u)
        {       b=Y(u,trafint);
                z=z+((float)(u*(u-1))*b);
        }
        ey2=z+a;
        return (ey2);
}

/*************************************************************/

float
MX(int node, float trafint)
{   float mx=(float)0.0;
        int x;
        for(x=0;x<=10;++x)
        {       mx=mx+((float)x*F(x,node,trafint));}
        return (mx);
}

/*************************************************************/

float
MK(int node, float trafint)
{   float mk=(float)0.0;
        int k;
        for(k=1;k<=12;++k)
        {       mk=mk+((float)k*P(k,node,trafint));}
        return (mk);
}
/*************************************************************/

float
VX(int node, float trafint)
{       float vx,a,b;
        float z=(float)0.0;
        int x;
        a=MX(node,trafint);
        for(x=0;x<=30;++x)
        {       b=F(x,node,trafint);
                z=z+((float)(x*(x-1))*b);
        }
```

```
        vx=z+a-(a*a);
        return (vx);
}

/*************************************************************/

float
EK2(int node, float trafint)
{       float ek2,a,b;
        float z=(float)0.0;
        int k;
        a=MK(node,trafint);
        for(k=1;k<=30;++k)
        {       b=P(k,node,trafint);
                z=z+((float)(k*(k-1))*b);
        }
        ek2=z+a;
        return (ek2);
}

/*************************************************************/

float
ET(int node, float trafint)
{       float et;
        et=MX(node,trafint)+MK(node,trafint)*MY(trafint);
        return (et);
}

/*************************************************************/

float
VT(int node, float trafint)
{       float vt,vrh,s;
        s=MK(node,trafint)*MY(trafint);
        vrh=(EK2(node,trafint)*EY2(trafint))-(s*s);
        vt=VX(node,trafint) + vrh;
        return (vt);
}

/*************************************************************/
```

```
float
T(int node, float trafint)
{       float t,a,b,c,d,n;
        a=A(node,trafint)*st();
        b=ET(node,trafint);
        c=VT(node,trafint);
        n=((float)2.0*a*b) + (a*a*c) - (a*a*b*b);
        d=(float)2.0*(a-(b*a*a));
        t=(n/d)*(st()*(float)1e6);
        return (t);
}
/*********************************************************/

main()
{       float L(float), A(int,float), R(int,float);
        float Q(int,float);
        float P(int,int,float), F(int,int,float), Y(int,float);
        float MX(int,float), MK(int,float), VX(int,float);
        float MY(float);
        float EK2(int,float), ET(int,float), VT(int,float);
        float EY2(float);
        float T(int,float), W(int,float), st(void), d(void);
        FILE *pf;
        int i,j;
        int n[21]={1,2,5,6,10,11,15,16,20,21,25,
                26,30,31,35,36,40,41,45,46,49};
        float t[41]={.1,.12,.14,.16,.18,.2,.22,.24,.26,.28,
                        .3,.32,.34,.36,.38,.4,.42,.44,.46,.48,
                        .5,.52,.54,.56,.58,.6,.62,.64,.66,.68,
                        .7,.72,.74,.76,.78,.8,.82,.84,.86,.88,.9};
        printf("\n\n\n\n\n\n\n");
        printf("                        PLEASE WAIT                     \n");
/*********************************************************/
        pf=fopen("proj.dat","w");
        for(i=0; i<=20; ++i)
                {    for(j=0; j<=40; ++j)
                                {fprintf(pf,"%f\n",T(n[i],t[j]));}
                        fprintf(pf,"\n");
                }
        fclose(pf);
/*********************************************************/
```

```
        printf("                    ALL DONE!\n\n");
        printf("            DATA IS OUTPUT TO 'PROJ.DAT'\n");
        return(0);
}
/* END */
/*

AVERAGE MEDIUM ACCESS DELAY EVALUATION

NETWORK PARAMETERS:

        ACCESS TECHNIQUE: DQDB
        NUMBER OF STATIONS: 50
        TRAFFIC: NON-ISOCHRONOUS (ASYNCHRONOUS SERVICE)
        SLOT SIZE: 69 BYTES (5 BYTE HEADER, 64 BYTE PAYLOAD)
        CHANNEL BIT RATE: 150 Mbps
        FIXED-LENGTH PACKETS (MESSAGE)
        PACKET LENGTH EQUALS SLOT SIZE
        PACKET ARRIVAL RATE: SAME FOR ALL STATIONS
        M/G/1 QUEUEING PROCESS

*/
/* 31 Oct 1991 */
```

Computer Program for None-Preemptive Priority Queueing

```
/*****************************************************************/
/*                          DON H. PHAM                          */
/* MEDIUM ACCESS DELAY EVALUATION FOR DISTRIBUTED QUEUEING */
/*                  DUAL BUS (DQDB) MAC PROTOCOL                 */
/*                          12 Nov 1991                          */
/*****************************************************************/

/* NO PART OF THIS PROGRAM CAN BE REPRODUCED BY ANY MEAN */
/*       WITHOUT THE CONSENT OF THE PROGRAMMER           */

#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>

/*************************************************************/

float
st(void)
{       float slot_time,slot_length,channel_cp;
        channel_cp=(float)150.0e6;
        slot_length=(float)(69*8);
        slot_time=slot_length/channel_cp;
        return(slot_time);
}

/*************************************************************/

float
d(void)
{       float pack_time;
        pack_time=(float)(64*8)/(float)150e6;
        return(pack_time);
}

/*************************************************************/

float
L(float trafint)
{       float l;
        l=trafint/(d()*1.225e+3);
```

```
        return (l);
}

/*********************************************************/

float
A(int node, float trafint)
{   float a=(float)0.0,l;
        int n;
        l=L(trafint);
        for(n=node+1; n<=50; ++n)
                {   a=a+l;}
        return (a);
}

/*********************************************************/

float
AA(int node, float trafint, int priority_level)
{   float aa;
        if(priority_level==1)
                aa=(float)0.1*A(node,trafint);
        if(priority_level==2)
                aa=(float)0.3*A(node,trafint);
        if(priority_level==3)
                aa=(float)0.6*A(node,trafint);
        return (aa);
}

/*********************************************************/

float
GG(int node, float trafint, int priority_level)
{   float gg=(float)0.0;
        int n,pl;
        if(node==49)
                gg=(float)0.0;
        else
                for(pl=1; pl<=priority_level; ++pl)
                        {
                         for(n=node+1; n<=49; ++n)
```

```
                        {    gg=gg+AA(n,trafint,pl);}
                    }
        return (gg);
}

/*********************************************************/

float
RR(int node, float trafint, int priority_level)
{    float rr;
        rr=AA(node,trafint,priority_level)*d();
        return (rr);
}

/*********************************************************/

float
R(int node, float trafint)
{    float r;
        r=A(node,trafint)*d();
        return(r);
}

/*********************************************************/

float
Q(int node, float trafint)
{    int n;
        float q=(float)0.0;
        if(node==1)
                q=(float)0.0;
        else
                for(n=1; n<node; ++n)
                        {    q=q+R(n,trafint);}
                return (q);
}

/*********************************************************/

float
QQ(int node, float trafint, int priority_level)
```

```
{    int n,pl;
        float qq=(float)0.0;
        if(node==1)
                qq=(float)0.0;
        else
                for(pl=1; pl<=priority_level; ++pl)
                        {
                         for(n=1; n<node; ++n)
                                {    qq=qq+RR(n,trafint,pl);}
                        }
        return (qq);
}

/*********************************************************/

float
P(int k, int node, float trafint)
{       float p,a,b;
        a=Q(node,trafint);
        if(k==1)
                b=(float)1.0;
        else
                b=(float)pow((double)a, (double)(k-1));
        p=b*((float)1.0-a);
        return (p);
}

/*********************************************************/

float
F(int x, int node, float trafint, int pl)
{    float a,b,c,g;
        float xfact=(float)1.0;
        float f=(float)0.0;
        int k,v;
        for(v=1; v<=x; ++v)
                {    xfact=xfact*(float)v;}
        g=GG(node,trafint,pl)*d()*QQ(node,trafint,pl)/(1.0-trafint);
        for(k=1; k<=10; ++k)
                {    a=(float)k*g;
                        if (x==0)
```

```
                        b=(float)1.0;
                else
                        b=(float)pow((double)a, (double)x);
                c=(float)exp((double)(-a));
                f=f+(b*c*P(k,node,trafint)/xfact);
        }
    return (f);
}

/*********************************************************/

float
Y(int u, float trafint)
{       float y,a,b;
        a=trafint*trafint;
        if(u==1)
                b=(float)1.0;
        else
                b=(float)pow((double)a, (double)(u-1));
        y=b*((float)1.0-a);
        return (y);
}

/*********************************************************/

float
MY(float trafint)
{       int u;
        float my=(float)0.0;
        for(u=1;u<=30;++u)
                {my=my+((float)u*Y(u,trafint));}
        return (my);
}

/*********************************************************/

float
M2Y(float trafint)
{   float m2y,a,b;
        float z=(float)0.0;
        int u;
```

```
        a=MY(trafint);
        for(u=1;u<=30;++u)
                {    b=Y(u,trafint);
                        z=z+((float)(u*(u-1))*b);
                }
        m2y=z+a;
        return (m2y);
}

/*********************************************************/

float
MX(int node, float trafint, int priority_level)
{    float mx=(float)0.0;
        int x;
        for(x=0;x<=10;++x)
                {    mx=mx+((float)x*F(x,node,trafint,priority_level));}
        return (mx);
}

/*********************************************************/

float
MK(int node, float trafint)
{    float mk=(float)0.0;
        int k;
        for(k=1;k<=12;++k)
                {    mk=mk+((float)k*P(k,node,trafint));}
        return (mk);
}

/*********************************************************/

float
M2X(int node, float trafint, int priority_level)
{    float m2x,a,b;
        float z=(float)0.0;
        int x;
        a=MX(node,trafint,priority_level);
        for(x=0;x<=30;++x)
                {    b=F(x,node,trafint,priority_level);
```

```
                    z=z+((float)(x*(x-1))*b);
                }
        m2x=z+a;
        return (m2x);
}

/*********************************************************/

float
M2K(int node, float trafint)
{   float m2k,a,b;
        float z=(float)0.0;
        int k;
        a=MK(node,trafint);
        for(k=1;k<=30;++k)
                {   b=P(k,node,trafint);
                    z=z+((float)(k*(k-1))*b);
                }
        m2k=z+a;
        return (m2k);
}

/*********************************************************/

float
ET(int node, float trafint, int priority_level)
{       float et;
        et=MX(node,trafint,priority_level)+MK(node,trafint)*MY(trafint);
        return (et);
}

/*********************************************************/

float
E2T(int node, float trafint, int priority_level)
{   float e2t;
e2t=M2X(node,trafint,priority_level)+M2K(node,trafint)*M2Y(trafint);
        return (e2t);
}

/*********************************************************/
```

```
float
WO(int node, float trafint)
{    float wo=(float)0.0;
        int pl;
        for(pl=1; pl<=3; ++pl)
{wo=wo+(AA(node,trafint,pl)*st()*E2T(node,trafint,pl)/(float)2.0);}
        return (wo);
}

/*************************************************************/

float
W1(int node, float trafint)
{    float w1,n,d,p1;
        p1=AA(node,trafint,1)*st()*ET(node,trafint,1);
        n=WO(node,trafint);
        d=(float)1.0 - p1;
        w1=n/d;
        return (w1);
}

/*************************************************************/

float
W2(int node, float trafint)
{    float w2,n,d,p1,p2;
        p1=AA(node,trafint,1)*st()*ET(node,trafint,1);
        p2=AA(node,trafint,2)*st()*ET(node,trafint,2);
        n=WO(node,trafint);
        d=((float)1.0-p1) * ((float)1.0-p1-p2);
        w2=n/d;
        return (w2);
}

/*************************************************************/

float
W3(int node, float trafint)
{    float w3,n,d,p1,p2,p3;
        p1=AA(node,trafint,2)*st()*ET(node,trafint,1);
        p2=AA(node,trafint,1)*st()*ET(node,trafint,2);
```

```
        p3=AA(node,trafint,3)*st()*ET(node,trafint,3);
        n=WO(node,trafint);
        d=((float)1.0-p1-p2) * ((float)1.0-p1-p2-p3);
        w3=n/d;
        return (w3);
}

/*************************************************************/

float
T3(int node, float trafint)
{   float a,t3;
        a=W3(node,trafint)+ET(node,trafint,3);
        t3=a*st()*(float)1e6;
        return (t3);
}

/*************************************************************/

float
T2(int node, float trafint)
{   float a,t2;
        a=W2(node,trafint)+ET(node,trafint,2);
        t2=a*st()*(float)1e6;
        return (t2);
}

/*************************************************************/

float
T1(int node, float trafint)
{   float a,t1;
        a=W1(node,trafint)+ET(node,trafint,1);
        t1=a*st()*(float)1e6;
        return (t1);
}

/*************************************************************/

main()
{       float st(void),d(void),L(float);
```

```
        float A(int,float),AA(int,float,int),R(int,float);
        float GG(int,float,int),Q(int,float),RR(int,float,int);
        float QQ(int,float,int),P(int,int,float);
        float F(int,int,float,int);
        float Y(int,float),MY(float),M2Y(float);
        float MX(int,float,int);
        float M2X(int,float,int),MK(int,float),M2K(int,float);
        float ET(int,float,int),E2T(int,float,int);
        float WO(int,float);
        float W3(int,float),W2(int,float),W1(int,float);
        float T3(int,float),T2(int,float),T1(int,float);
        float a,b,c;
        int i,j;
        int n[11]={1,5,10,15,20,25,30,35,40,45,49};
        float t[9]={.1,.2,.3,.4,.5,.6,.7,.8,.9};
        FILE *pf;
        printf("\n\n\n\n\n\n\n");
    printf("              PLEASE WAIT                    \n");

/*************************************************************/

pf=fopen("projaa.dat","w");
        for(i=0; i<=10; ++i)
                {       for(j=0; j<=8; ++j)
                                {       a=T1(n[i],t[j]);
                                        b=T2(n[i],t[j]);
                                        c=T3(n[i],t[j]);
                                        fprintf(pf,"%f %f %f\n",a,b,c);
                                }
                }
fclose(pf);

/*************************************************************/

printf("                    ALL DONE!\n\n");
printf("           DATA IS OUTPUT TO 'PROJAA.DAT'\n");
return(0);
}
/* END */
/*
```

AVERAGE MEDIUM ACCESS DELAY EVALUATION

NETWORK PARAMETERS:

ACCESS TECHNIQUE: DQDB
NUMBER OF STATIONS: 50
TRAFFIC: NON-ISOCHRONOUS (ASYNCHRONOUS SERVICE)
SLOT SIZE: 69 BYTES (5 BYTE HEADER, 64 BYTE PAYLOAD)
CHANNEL BIT RATE: 150 Mbps
FIXED-LENGTH PACKETS (MESSAGE)
PACKET LENGTH EQUALS SLOT SIZE
PACKET ARRIVAL RATE: SAME FOR ALL STATIONS
NONPREEMPTIVE PRIORITY QUEUEING PROCESS
THREE PRIORITY LEVELS

*/
/* 12 Nov 1991 */

Vita

Don H. Pham was born in Saigon, Vietnam in 1965, and grew up in Fairfax, Virginia. Don first became interested in electronics when he received an APPLE IIe computer with 64K RAM and a RADIO SHACK 160 IN ONE project kit. When Don tired of experimenting with the kit, he entered *Virginia Military Institute (VMI)*, where he obtained his Bachelor of Science degree in Physics with Distinction in 1988. After serving as an Infantry Officer in the United States Army, Don entered *Virginia Polytechnic Institute (Virginia Tech.)* in September 1990 to read for his Masters degree in Electrical Engineering. He spent two internships with GTE Spacenet in Northern Virginia, where he gained valuable insight into telecommunication technology. His interest includes fast packet switch, frame relay and network protocol. Recently, he has been engaging in research on Asynchronous Transfer Mode (ATM). Don defended his thesis at the Virginia Tech., Telestar Graduate Center on 11 February 1992.

Fairfax, Virginia
26 February 1992