DATA ADAPTOR UNIT FOR AN ELECTRONIC EXCHANGE:

SYSTEM DESIGN AND SIMULATION STUDY

KASPER REININK

A Dissertation submitted to the Faculty of Engineering,
University of the Witwatersrand, Johannesburg,
for the Degree of Master of Science in Engineering

Johannesburg 1981

ABSTRACT

The development of pulse code modulation systems and electronic exchanges, together with the increase in volume of data traffic, have made it desirable to establish a means for handling both voice and data traffic in the telephone network. A technique is required whereby integrated voice and data switching may be accomplished in an electronic exchange.

It is proposed that a data adaptor unit be incorporated in electronic exchanges for the purpose of concentrating data traffic onto PCM highways. A conceptual design for the architecture of such a unit is presented and its performance evaluated. The protocols for connecting data adaptors into a network, and the use of flow control procedures for regulating data traffic within the network, are described. The hardware implementation of the unit is not considered.

The proposed data adaptor operates on the packet switching principle. It is comprised of three types of modules each dedicated to carrying out a particular communication function. The modules are the interface processor that interacts with data transmitting devices, the network processor for sending data over PCM channels between adaptors, and the supervisory processor for regulating activities within the adaptor unit.

Specific features developed for the adaptor unit include a protocol allowing the sharing of a physical PCM channel by several logical paths, as well as the transmission of only incorrect packets; the partitioning of memory along topological lines to aid in flow control. An algorithm, situated in the supervisory processor, is presented for controlling packet exchanges between modules in the adaptor.

A simulation facility, written in FORTRAN IV, was developed to determine the performance capabilities of the data adaptor. Two networks based on the proposed unit were simulated. Response and throughput measurements have been listed. The effect of certain model parameters on network performance, such as the packet retransmission interval, were examined. The simulation plots listed indicate the characteristics of a data network consisting of adaptor units.

Comparison of the performance results attained by the hypothetical network with those of practical networks, indicates that the proposed data adaptor has the response and throughput capability to serve as the basis for a data traffic system. A unit path length response of 18 milliseconds was obtained for the hypothetical network at mean line utilization 0,46. The channel capacity was 64000 bits per second, and the mean packet size 470 bits.

I hereby declare that this dissertation is
my own work and that it has not previously
been submitted for a degree at any University

K Reinink

ACKNOWLEDGEMENTS

The work described in this dissertation was carried out in the Electrical Engineering Department at the University of the Witwatersrand, Johannesburg.

The author extends his sincere appreciation to the following people for their support and guidance:

Professor H E Hanrahan, Department of Electrical Engineering, for his supervision of the work.

Professor M G Rodd, Department of Electrical Engineering, for his constructive comments on the draft presentation.

The author expresses his gratitude to the Council for Scientific and Industrial Research, and the University of the Witwatersrand, for their generous financial assistance.

Finally, the author thanks Mrs Brooks for typing the dissertation.

CONTENTS

TABLES

CHAPTER 1

DATA ADAPTOR FOR AN ELECTRONIC EXCHANGE

1.1 Introduction

The introduction of pulse code modulation (PCM) techniques and electronic exchanges for both the transmission and switching of information has led to a shift in thinking away from frequency division multiplexing to that of time division multiplexing systems. Whereas, in conventional exchanges, computers and terminals required modems to transmit their information over telephone lines, the PCM exchange system requires no such devices since all information is transmitted in digital fashion.

The 30/32 PCM system allows for a higher transmission rate than the existing rates of 4800 to 9600 bits per second at which data travel over the analogue lines.[2] This system consists of thirty speech and two control channels for signalling purposes, each of capacity 64000 bits per second.[1,2] Switching takes place via time-space-time division multiplexing, to connect the various subscribers of the network.

Developments such as terminal inquiry systems, bulk data transmission in the form of files, facsimile mail transmission and electronic fund transfer, means that the volume of data traffic exchanged between geographically separated devices is expected to increase significantly.[1] The transmission of data will become a major source of traffic alongside that of voice communications.

CHAPTER 1

DATA ADAPTOR FOR AN ELECTRONIC EXCHANGE

1.1 Introduction

The introduction of pulse code modulation (PCM) techniques and electronic exchanges for both the transmission and switching of information has led to a shift in thinking away from frequency division multiplexing to that of time division multiplexing systems. Whereas, in conventional exchanges, computers and terminals required modems to transmit their information over telephone lines, the PCM exchange system requires no such devices since all information is transmitted in digital fashion.

The 30/32 PCM system allows for a higher transmission rate than the existing rates of 4800 to 9600 bits per second at which data travel over the analogue lines.[2] This system consists of thirty speech and two control channels for signalling purposes, each of capacity 64000 bits per second.[1,2] Switching takes place via time-space-time division multiplexing, to connect the various subscribers of the network.

Developments such as terminal inquiry systems, bulk data transmission in the form of files, facsimile mail transmission and electronic fund transfer, means that the volume of data traffic exchanged between geographically separated devices is expected to increase significantly.[1] The transmission of data will become a major source of traffic alongside that of voice communications.

The above considerations imply that the use of a PCM telephone network would result in the rapid assimilation of most forms of data traffic. It must be kept in mind, however, that telephone networks, PCM or otherwise, have been designed in the first place to facilitate the handling of voice traffic.

There exist certain differences in the characterisation of voice and data traffic. The transmission of data usually occurs in bursts, whereas the transmission rate in telephone channels is constant. Different coding techniques are made use of in data communications. Widespread channel capacities ranging from a few bits per second to (ideally) millions of bits per second are required. Telephone traffic requires only a single fixed capacity channel.

Certain requirements are to be met if communication between computers is to take place. Examples are, negligible delay in setting up a call, provision for receiving messages at all times from remote terminals and other centres, automatic code and format translation.

The introduction of certain improved technologies over the past few years means that alternative solutions may be possible for data transmission, satisfying computer communication demands. The introduction of TDM techniques for information transmission in telephone networks has led to increased throughput capabilities up to 64000 bits per second per channel. The cost of processing and storing data has decreased to such an extent that it has become feasible to

dynamically allocate communication resources, making it possible to integrate traffic requiring a variety of bandwidths within a single network.

A technique is required whereby integrated voice and data switching may be accomplished in an electronic exchange that makes use of pulse-code-modulation channels for information transmission.

1.2 Review of Data Transmission

Three types of network have been used for the transmission of data traffic:[3]

(1) Circuit switching systems that require a complete path to be set up from end to end, and only when the connection has been established may messages be multiplexed into the system. Blocking of calls and therefore call connection times may become a problem.

(2) Message switching that requires bulk storage devices for the buffering of entire messages before data is forwarded to the destination; these systems are generally more suitable for large file transfers.

(3) Packet switching systems where a message is divided into shorter packets, each of which works its way through the network from node to node. The packet-switch method possesses

the capability of satisfying most of the requirements for adequate computer-communication.

It is a characteristic of packet-switching that, once a call has been set up, the channel resources are only used when a device transmits a message. When the pair of subscriber devices are idle, the channel may be allocated to other users. Provision is made for receiving messages at all times without on each occasion proceeding through the motions of call connection. Code and format translations are possible.(4)

The differences between computer and telephone type traffic have resulted in various solutions to the data transmission problem. These include:

(1) the use of modems for transmitting data over the analogue lines of the public telephone system,

(2) the use of circuit-switch based networks for data transmission. The Electronic Data Switching System makes use of PCM transmission links and circuit switching data exchanges that include synchronous data multiplexers.(5) DATRAN is a specialized common carrier operating a fast-connect switched data network, switching carried out by means of computer-controlled time-space-time division multiplexing facilities.(6)

(3) dedicated message and packet-switching circuits have been constructed, e.g., commercial time-sharing networks (TYMNET), airline reservation facilities (SITA), and computer resource sharing networks (ARPA).[6] Certain manufacturers have constructed their own networks, such as IBM's Synchronous Network Architecture that enables a variety of machines to be interconnected over geographically dispersed regions.[7]

(4) ideas of integrating voice and data traffic systems. A number of analyses have been conducted to determine the conditions for which either a packet or a circuit switch based technique is to be used, the two techniques being incorporated into a single node in a network.[8,9] The majority of integrated voice and data switching systems have been confined to introducing a master frame format of a statistical TDM facility, allocating certain slots to voice and other slots to data information.[10]

Available integrated voice/data switching facilities have been based on the circuit-switch method. Packet-switching has generally been dedicated to specific applications. Required is a system incorporating both types of methods for switching voice and data traffic.

A survey was carried out to obtain an idea of the state of communication processors. The article by Newport and Ryzlak[11] contains a review of the use of small computers for communication processing. They discuss the software and hardware requirements for network processors, data concentrators, and message-switching systems.

A corresponding article by Mills[12] on communication software deals with network control, message processing, and error recovery. Steel and Mattson[13] describe a 16-bit general purpose minicomputer configured to operate as a communications processor. A minicomputer-multiprocessor built specifically to meet expanding traffic requirements, and to provide extreme reliability is the Pluribus system. It performs packet-switching in the ARPA network.[14]

The above constitute a small selection of papers on communication processors. Most deal with the software and hardware utilities applicable to minicomputers. A paper describing the technology used in the implementation of packet processors has been published by Roberts.[15] He identifies three generations of architecture. In the late 1960's limited capacity minicomputers in a single-level hierarchy distributed network were utilized. A typical example is that of the ARPA network. The second generation networks utilized more powerful minicomputers in a two-level network with central office switching centres containing minicomputer subnets, e.g., TELENET. In the late 1970's, with the advent of the microprocessor, the third generation of networks is in the process of development in the form of multi-microcomputer systems serving as nodes in a three-level hierarchy.

A report by Sarch[16] confirms that there is a trend to multi-microcomputer configurations, due to the decreasing costs of micro-processors and memory. It is pointed out, however, that careful consideration must be given to the overheads involved in managing all

these decentralized processors. Each microprocessor is being seen as dedicated to single tasks, such as coding, diagnostics, interfacing, etc.

The above leads to the observation that mini-computers have been largely used to date to perform the packet-switch function, that the application of multi-microcomputer architectures for communication is regarded favourably, but that not much has been done to construct such computer structures. The idea of micro-computer based modules for packet-switching in electronic exchanges indicates the possibility of realizing an integrated voice-data switching system.

1.3 Description of the Data Adaptor Unit

It is proposed that a data adaptor unit, based on microprocessor-type elements, be included in electronic exchanges for the purpose of handling data traffic. Adaptor units will make use of the packet-switch method, which is best suited to meet computer communication requirements. A solution for the architecture of such a data adaptor is presented in terms of functional modules, the communication protocols used to link data units into a network, and flow control procedures for regulating traffic in the network.

1.3.1 The Nodal Architecture

The architecture (see Figure 1.1) consists of three types of modules, interface and network processors, and a supervisory or switch processor. Each module is envisioned to consist of a microprocessor

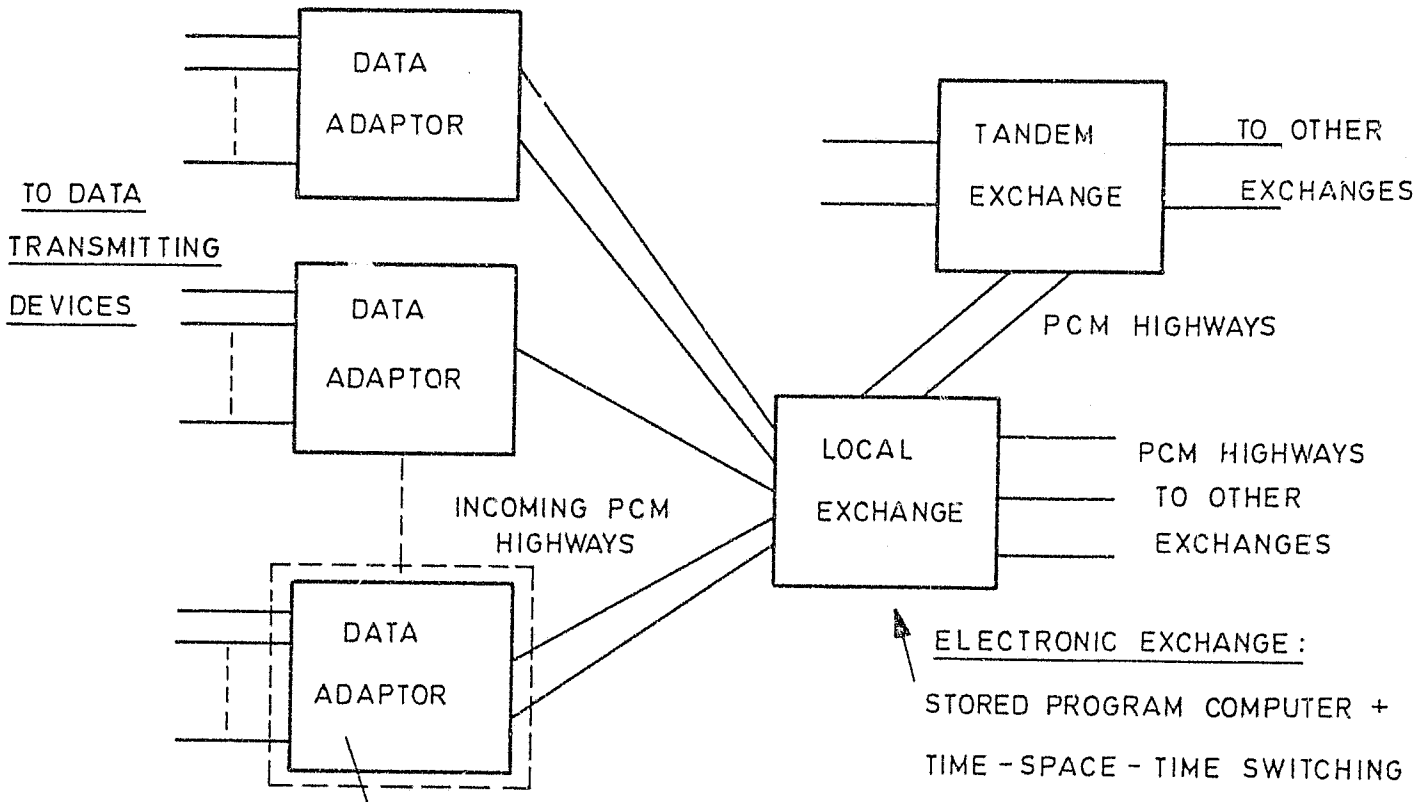


FIGURE 1(a) DATA TRAFFIC CONCENTRATION

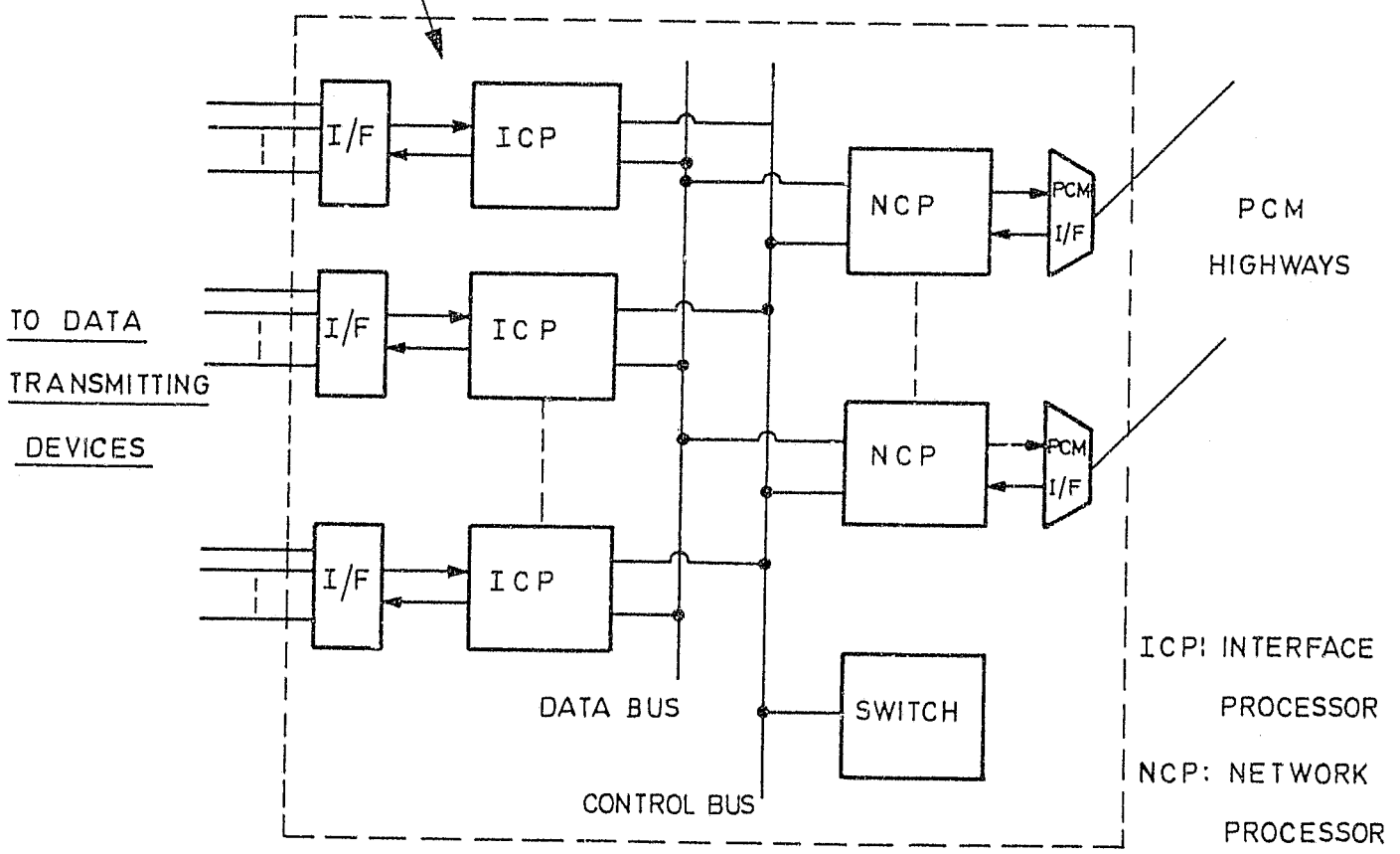


FIGURE 1(b) DATA ADAPTOR UNIT

plus memory and appropriate interface integrated circuits. The modules are linked together via two separate, asynchronous busses, termed the data and control bus.

Each of the modules performs a specific function. The interface processors (ICP's) interface with user devices, and perform functions such as code and format translations, packetizing of messages, and reassembly. The network processors (NCP's) control the transmission of packets via PCM channels that connect nodes with one another. The switching processor controls the activities within the node. At regular intervals, it determines the status of the node by polling the NCP's and ICP's. The polled information is processed, and commands are subsequently issued to the interface and network processors, directing these modules as to which packets to send, and when transmission is to take place. The switching processor takes into account factors relevant to nodal operation, such as the availability of buffering in modules, and the congestion experienced on internodal channels.

Data packets are transferred between ICP and NCP modules via the data bus. Information exchange between the switch and the other processors in the node, takes place over the control bus. Note that the detailed design of the microprocessor elements is not of concern here.

The basic operation of the data adaptor unit is summarized as follows. The ICP modules interact with network users. Packets for network transmission are exchanged with NCP's via the data bus. The NCP's ensure packets are transmitted to, or received from, neighbouring nodes. The switch processor supervises the activities of the ICP and NCP modules over the control bus.

1.3.2 The Network Protocols

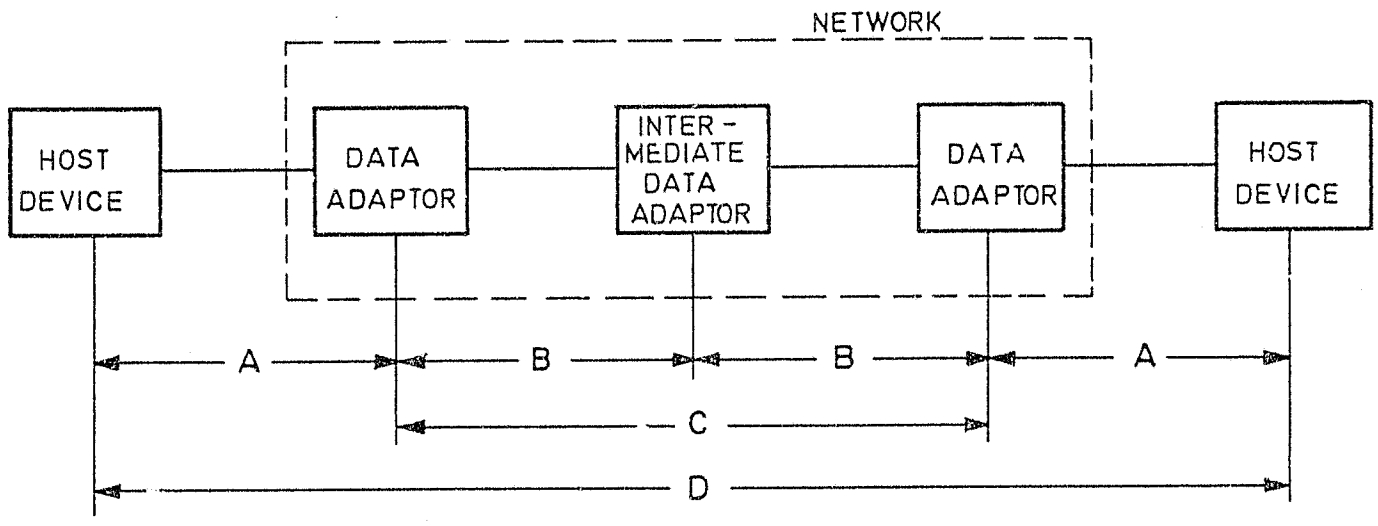
The protocols implemented for the network are shown in Figure 1.2. Depicted are two host devices that require high bandwidth communication facilities on demand. The traffic between the two host devices is bursty, with long gaps in time between the transmission of messages.

The host protocol is undefined. It is both specific to the pair of communicating host devices, and transparent to the network.

The host-network protocol allows communication of user devices with the network. Although the codes and formats used may be device specific, certain protocol procedures necessary for controlling the data flow are common to all ICP-host links.

The information flow between nodes is implemented via the internodal protocol. The protocol is based on the send-and-wait procedure commonly used in packet networks. It is a high-level bit-oriented protocol, allowing up to eight packets to be transmitted before an acknowledgement is required, thus ensuring full utilization of a channel. The protocol differs from commonly encountered bit-oriented protocols, such as Synchronous Data Link Control, in that the acknowledgement process relies upon associating a bit position in a transmitted frame with a packet to be acknowledged, rather than modulo sequence numbers.

The end-to-end protocol is used to control the traffic flow between two ICP's each connected to a pair of communicating host devices. It allows for the pipelining of long file transfers, and for the efficient



A : HOST NETWORK PROTOCOL

B : INTERNODAL PROTOCOL

C : END - TO - END PROTOCOL

D : HOST PROTOCOL

FIGURE 1-2 NETWORK PROTOCOLS

transmission of short interactive messages. User devices are connected via logical links, realized by the end-to-end protocol. Once a connection has been established, data may be transmitted at any time, the necessary channel and buffer resources being allocated when required.

No central or supervisory node is present in the network to control data flow between nodes. The above protocols are required to cooperate in the transmission of a message from one host to another.

1.3.3 Flow Control

Traffic flow control measures may be divided into the two broad categories of local and global control. Global flow techniques extend across the entire network domain and, generally, have the function of limiting the number of packets in the network. It has been implemented in conjunction with the end-to-end protocol. Only a maximum specified number of packets per logical link may be in transit across the network at any one time.

The local flow measures operate within the domain of a node and its immediate neighbours, to prevent the occurrence of congestion in the nodal channels. Such a measure has been implemented in the switching processor of each node. Data traffic within the node is classified into the two catagories:

(i) traffic entering the network,

(ii) traffic in transit or leaving the network.

The second class of packets is given priority over the first. The measure has been implemented by the use of time-dependent priority scheduling.

1.4 Chapter Survey

This chapter has served to introduce the structure of the data adaptor unit. It was proposed that the data adaptor, based on the packet-switch method, be incorporated into an electronic exchange to deal exclusively with data traffic. The following chapters expand on the system design of the adaptor unit.

Chapter 2 concentrates on the local and global flow control measures incorporated into the data adaptor. The throughput and delay characteristics of packet networks is discussed. It is shown that excessive utilization of communication channels results in performance degradation.

The simulation facility used to model and test the performance of networks based on the proposed data adaptor is described in Chapter 3. The input parameters required to describe a network are extensively listed. The facility concentrates on simulating the operation of a node; simulation of a network occurs by defining the topological connections between nodes.

A functional description of the interface processor module is contained in Chapter 4. An end-to-end protocol is described. A graphical demonstration of the protocol workings is included. Finally, the buffer management technique used in the interface processor is outlined.

Chapter 5 describes the functions of the network processor module. The internodal protocol and processor buffering structure are defined. To aid in flow control, memory is partitioned according to topological connections of a node to its neighbours. A nodal link was simulated to gain insight into the parameters affecting nodal performance.

The method of packet interchange between modules within a node is discussed in Chapter 6. Nodal architecture is outlined. The role of the switching processor in controlling the activities of the network and interface processors is described. The chapter concentrates on the nature of the algorithm executed by the switch processor to perform the controlling action.

The performance of two simulated networks is listed and discussed in Chapter 7. A preview of the network parameters used, the assumptions made in the simulation, and the indices required for defining network performance, is contained. Tests conducted include, the simulation time required for the models to arrive at a state of equilibrium, response and throughput measurements, local flow control behaviour, and the effect of certain nodal parameters, the retransmission interval, data bus rate and switch algorithm period, on network performance.

Conclusions as to the feasibility of the proposed data adaptor are set out in Chapter 8. A number of suggestions for further work in this area is included.

CHAPTER 2

FLOW CONTROL MEASURES

2.1 Introduction

Chapter 2 studies flow control aspects pertaining to a network consisting of a number of connected data adaptor units. The application of such control measures is in fact necessary for all packet-switch based networks.

The network is considered as consisting of a set of components or resources, e.g., buffers, channels, processors, that must be shared amongst a number of logical links for the packet transmission phase to be executed. An excessive number of conflicts over resources can result in the network performance of packet response and throughput being degraded. Graphical plots have been included to illustrate this effect. To prevent the occurrence of excessive resource sharing conflicts, flow control procedures need to be instituted. Both local and global techniques have been incorporated into the data adaptor unit, and these are described.

Two terms relevant to this chapter are defined. 'Global Flow Control' is a procedure by which the network regulates traffic flow between a source user device and a destination user device. 'Congestion Control' or 'Local Flow Control' is a procedure whereby distributed network resources are protected from over-subscription. In general successful operation of global flow control procedures for every pair of communicating processes does not guarantee that the network resources will remain uncongested, hence the need for both global and local control.

2.2 Resource-Sharing in Packet Networks

One of the main problem areas to be found in packet-switching technology is that of the need for distributed resource sharing. Resources must be allocated on demand at various points in the network if successful transmission of data is to take place. The performance of the network is linked to the efficiency of resource allocation.

The principle of resource allocation is briefly explained. The computer network has, as one of its main resources, high capacity channels (others are buffers, processors, data busses, etc.). These channels are allocated to users for short durations of time - the time to transmit a packet of specified size via the 6 400 BPS link. If all users were to send data to the network for transmission at the same point in time, the capacity of the channel to transport the information would be exceeded. In practice such an event will rarely occur, and then only for a short duration.

As the number of messages to the network is increased per unit time, more conflicts will arise regarding the sharing of resources. In extreme conditions deadlock may occur in which two or more competing demands have each been assigned a subset of their necessary resources; neither can proceed until one of them collects some additional resources which, currently, are assigned to the other, and neither is willing to release any resources currently assigned to him. When too many conflicts for resources take place the network performance degrades; an increased time to transport a packet from source to destination host

results and the average number of packets transmitted per unit time, decreases.

The importance of the above discussion is that the data adaptor unit cannot be considered in isolation from its neighbouring nodes. A data adaptor acts, not only as a source and a destination of messages in its interaction with host devices, but it also serves as intermediate node, receiving and transmitting packets enroute from and to its neighbours. The unit must be regarded as one part of a network system for data transmission. The success of a nodal architecture can only be determined when a number of units are linked together into a network and the network system is able to achieve a satisfactory level of performance.

It is important to note that the functioning of any data unit is, to a large extent, determined by the operation of its neighbours. The difficulty of designing successful packet networks with adequate performance levels occurs due to the distribution of the resources in geographically dispersed communication units, where access to these resources arises from asynchronous processes in a highly bursty fashion. Not only is the demand process bursty, but it is also high unpredictable, in the sense that the instants when the demands arise are not known ahead of time.

The problem of resource sharing in a distributed environment manifests itself in the flow control problems. The problem of flow

control is to regulate the rate at which data crosses the boundary of the communication network.

It is to be noted that the data network envisaged would be decentralized. No one node would control the operation of the network in centralized fashion; this is impractical. By the time control data were to reach the supervisory node, and appropriate control action taken, the information received might have become invalid.

2.3 Characteristics of Packet Networks

Investigations carried out by means of analyses and simulation[20] as well as data gathering on experimental networks, particularly ARPANET,[21] have shown that the response and throughput characteristics of packet networks, may be depicted as shown in Figure 2.1.

As the input load in packets per second is increased, the response of the network remains fast at the value of To milliseconds, the no-load packet delay, until an input load of γ^* is reached. Thereafter the network becomes 'unstable' due to an excessive number of resource-sharing conflicts occurring. The response time then increases at a virtually exponential rate.

The throughput curve shows a similar characteristic. Increasing the input load past a certain point (in the region of γ^*) will cause

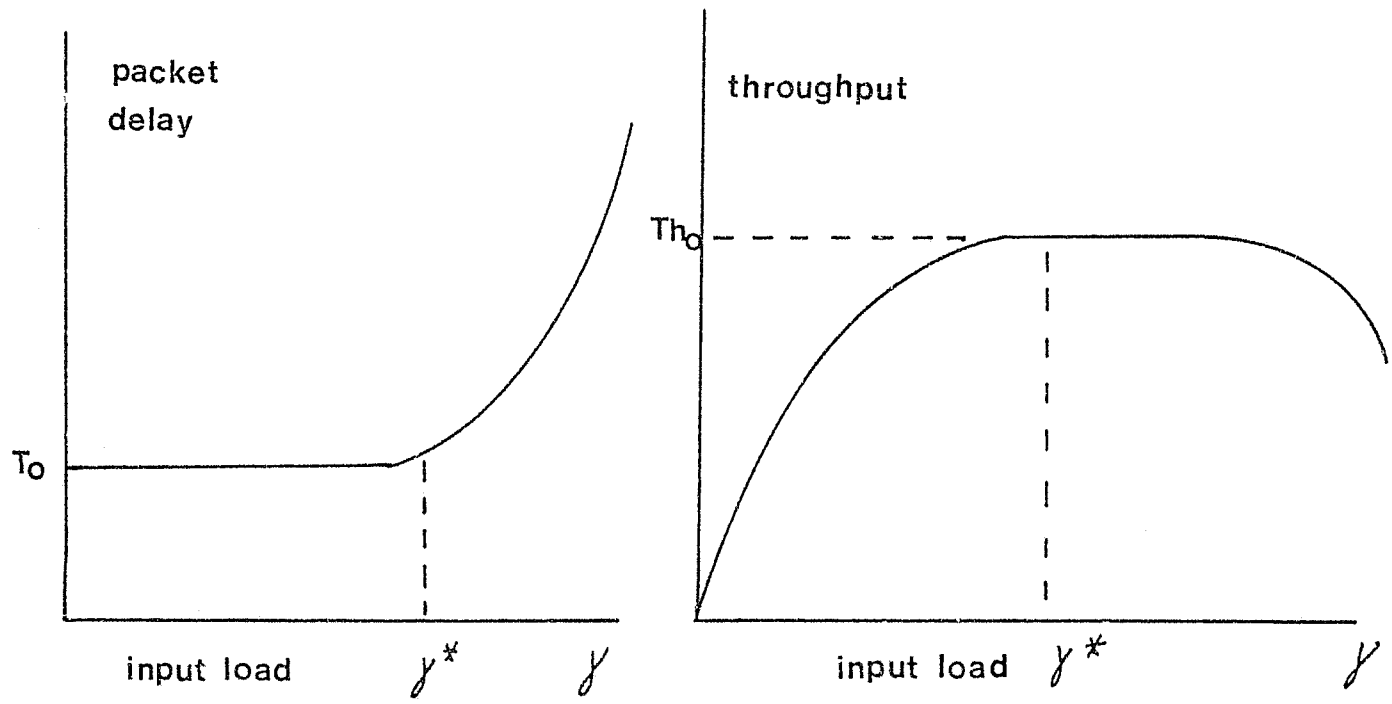


Figure 2.1 Network Characteristics

the throughput in packets per second to start decreasing. The interpretation is that under normal conditions the input load equals the network capacity; a point is reached, however, where the input will start to exceed the capacity of the network to transport the packets and performance degradation occurs.

The purpose of the flow control measures is to prevent the network from entering the region where the performance indices of response and throughput become degraded, and the amount of work performed by the system effectively decreases. The above graphs are somewhat deceptive in the sense that absolute input load values do not of themselves result in congestion taking place, rather resources become overloaded,

e.g., a channel may become blocked.

Equations for a simplified threshold model of network delay, derived in Kleinrock,[21] are restated.

Consider an M-channel, N-node network. The channels are assumed to be noiseless, the i'th channel having capacity Ci bits/sec. The nodal processing times are assumed negligible. The packet lengths are assumed to follow an exponential distribution with mean $1/\mu$ bits. All nodes in the network have infinite storage capacity. A fixed routing procedure is defined. Define λ_i as the average number of packets per second that travel over the i'th channel. In order to evaluate the delay at a channel imbedded in the network it is necessary to make use of Kleinrock's independence assumption: each time a packet is received at a node, a new length $\bar{b}$ is chosen independently from the distribution:

$$p(b) = \mu e^{-\mu b} \qquad b \geqslant o \qquad (2.1)$$

This assumption has been shown to be reasonable for networks of moderate connectivity, i.e., that most nodes should have more than one channel entering and more than one leaving.

Representing the i'th channel by a M/M/1 queueing system, i.e., Poisson arrival (M) and exponential server (M) distributions, single server, the mean packet delay is given by:

$$T = \sum_{i=1}^{M} \frac{\lambda i}{\gamma} \left| \frac{1}{\mu Ci - \lambda i} \right| \qquad (2.2)$$

where γ is the total input load in packets/sec.

If a relatively homogeneous set of Ci capacities is assumed then, as the load in the network is increased, no individual term in the above summation will dominate, until the flow in one channel io approaches the capacity of the channel. At that point T will grow rapidly. The mean packet delay curve will thus express a threshold behaviour.

The no-load delay To is expressed in terms of the mean path length $\bar{n}$ as

$$To = \bar{n} \sum_{i=1}^{M} \left(\frac{\lambda i}{\lambda}\right)\left(\frac{1}{\mu Ci}\right) \qquad (2.3)$$

The saturation load γ^* corresponds to the smallest value of γ at which some channel is saturated; this is the point at which $\lambda io = \mu Cio$ where io is the critical channel. The above equation corresponds to pio = 1. In practice saturation in a channel generally occurs with the utilization factor, p, taking on values of 0,7 and greater.

Control measures must be instituted to maintain the network at the response and throughput levels of To milliseconds and Tho packets/sec respectively, despite possible further increases in the input traffic load. If effective measures are not taken, congestion will result.

2.4 Global Flow Control

Global flow techniques extend across the entire network domain, and generally have the function of limiting the number of packets in

the network. The global flow control has been implemented in conjunction with the end-to-end or source node-to-destination node protocol.

The link connecting two communicating host devices consists of a number of resources, such as channels, processors, buffers, etc., that must be allocated on demand to the host connection for communication to take place. This connection is termed a logical link. It consists of both hardware and software components, e.g., memory, processing capacity. The link is not dedicated by providing a physical channel between the two hosts as in circuit switching. Rather, once a call establishment procedure has been evoked, the link may be identified by addresses stored in the supervisory processors in nodes. At any time thereafter a message may be forwarded to the source node by the host device. The required resources will be allocated in order for transmission of the packets, constituting the message, to take place.

The global control measure has been instituted by limiting the number of packets in transit per logical link at any one time. For proper functioning it is essential to obtain some idea of the number of logical links that will be operational. [illegible]ll be described by some statistical distribution, in terms of me[illegible]packet lengths and interarrival times, possibly as a function of the time of day.

Long messages are divided into segments by the host devices, the

segments being divided into packets by the nodes. The packets are initially retained in the ICP's. The flow control procedure ensures that only a specified maximum number of packets may be in transit, in the case of the simulations, eight packets. The remaining packets (if any) must wait until the first eight have been transmitted. In effect, blocks of eight packets are sent into the network at a time, under control of the end-to-end protocol. Thus, although the number of messages stored in the interface processors may be large, the number allowed into the store-and-forward section (as realized by the network processors) is strictly limited.

2.5 Local Flow Control

Local flow measures operate within the domain of a node and its immediate neighbours, to prevent the occurrence of congestion in the region of the node. The local flow measures are implemented in the switching processor of each node. The switch samples the state of the node at regular intervals, processes the status data, and sends commands to the ICP and NCP modules directing their activities. Two types of flow measures are considered, the allocation of priorities to packets, and the partitioning of memory along topological lines.

2.5.1 Priority Allocation Technique

The switch algorithm has been based on exploiting certain properties of a store-and-forward network, i.e., its queueing structure. The nodal operation may be reduced to the concept of the servicing of a set of

queues in the ICP and NCP modules by the switch processor. The algorithm does not rely upon lengthy mathematical calculations based on statistical traffic flow measures, as is the case in adaptive routing, but makes use of time-dependent priority evaluations. The use of time-dependent priority evaluations for preventing congestion is not treated in the literature.

Packets have been grouped into the following two classes:

i) packets entering the network;

ii) packets in transit, or leaving the network.

Each class is allocated a priority value that is incremented with respect to time (see Figure 2.2).

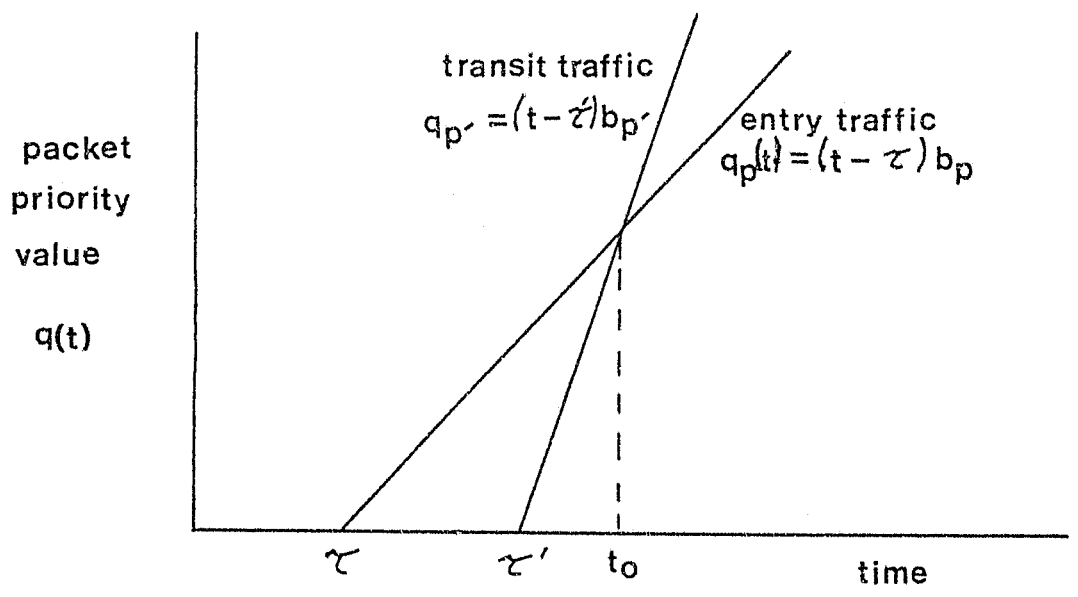


Figure 2.2 Priority Assignment

The function of the switch is to control the flow of traffic proceeding from the ICP modules to the store-and-forward section, realized by the NCP modules, on a local level. A node that starts to become congested will grant priority to transit traffic at the expense of the entry traffic, thereby causing less packets to gain admittance to the network.

The time-dependent priority system provides a set of variable parameters, bi, where

$$0 \leqslant bp \leqslant bp'$$

These parameters may be used to adjust the relative waiting times for the packets in the queue. Assume that the packet under consideration arrives at time τ and is assigned at time t a priority qp(t) where

$$qp(t) = (t - \tau)bp$$

where t ranges from τ until the time at which the packet's service time is completed. Whenever the service centre (i.e., the switch) is ready for another packet, it chooses to service that packet with the highest instantaneous priority q(t). If a tie for the highest priority occurs, the tie is broken by the first-come-first-served rule.

Figure 2.2 shows the manner in which packets from two priority groups interact. A packet from priority group p (entry traffic) arrives at time τ and attains priority at a rate bp. At time τ' another packet enters the node and attains priority at a rate bp' (transit traffic group). If the service facility becomes free at any time between τ and to, the packet from group p will be serviced in preference to that of the packet

from group p'. For any time after to, the transit traffic packet will be serviced beforehand.

Kleinrock has analyzed such a queueing system.[21] The analysis models a node as consisting of an M/G/1 queue - Poisson arrival rate, general service time distribution, and single server. Such forms an approximation to the queue maintained by the switch processor servicing ICP and NCP module requests.

The mean waiting times for group p and group p' packets in a non-preemptive time-dependent discipline is shown in Figure 2.3. The mean waiting times as a function of the queue utilization factor p, for the parameters b1/b2 = 1, b1/b2 = 0,2, and b1/b2 = 0,1 are plotted. The utilization factor p is defined to be the ratio of the mean arrival rate to the capacity of the system to service the arrivals. Note that it is not the absolute values bp and bp', but the ratios bp/bp' that are important. At the lower utilization factors ($p < 0,4$) the priority designations have little effect and the waiting times for both traffic groups is the same. At the larger values ($p > 0,7$) the mean waiting time of transit traffic is far less than that of entry traffic.

The maximum service rate of the data adaptor in packets per second is finite. By granting transit traffic servicing priority (as opposed to entry traffic) less packets wanting to gain entry to the network are serviced. The overall effect is to limit the number of packets that gain entry to the network under heavy traffic conditions. In Chapter 7, simulation results for various parameter ratios bp/bp' are listed,

showing the effect of this technique on mean packet response in a network.

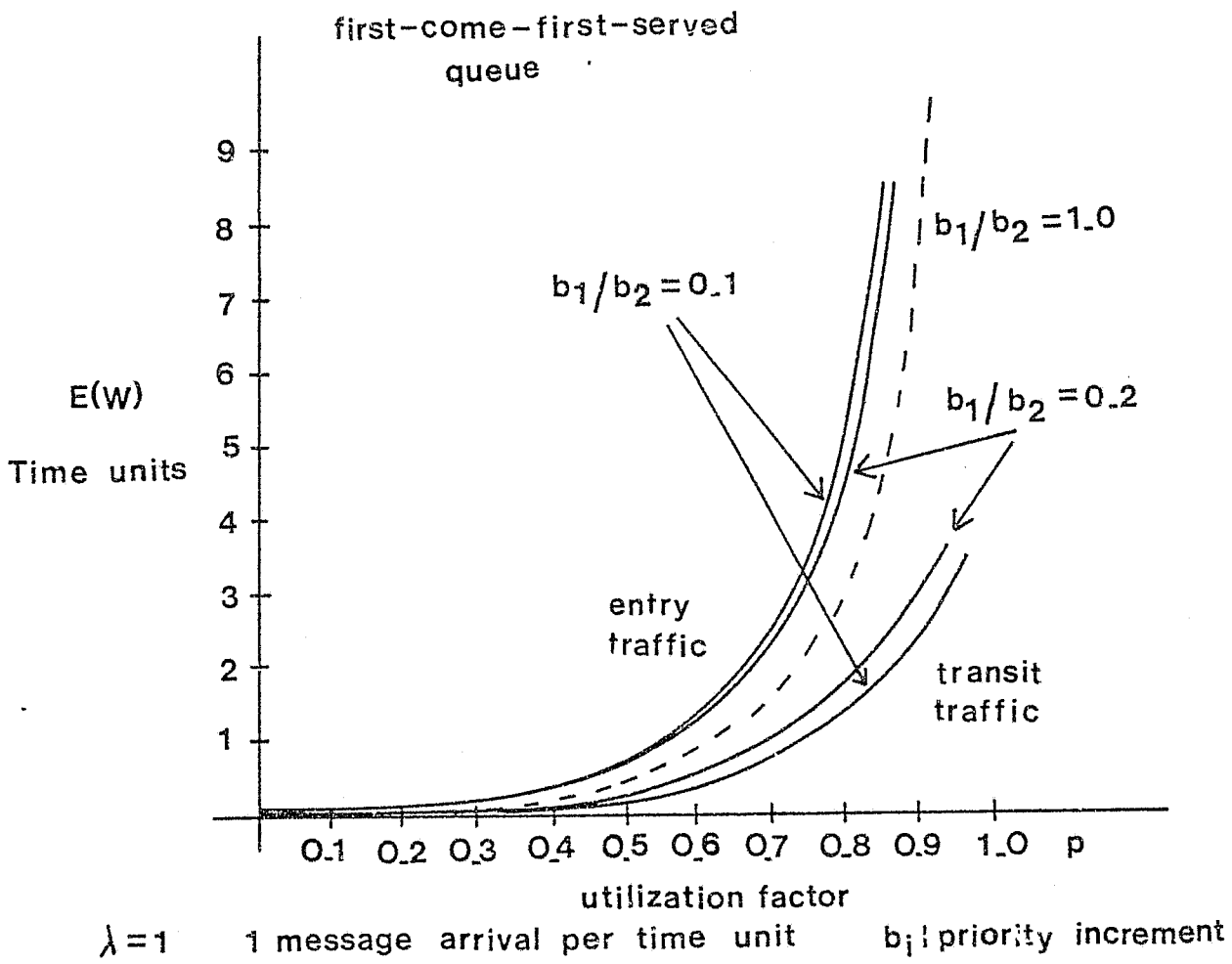


Figure 2.3 Mean Waiting Times for Delay Dependent Traffic

2.5.2 Memory Partitioning Technique

A second local flow control measure is realized by the internodal protocols. An NCP module is connected to each end of a PCM channel, implementing the data transmission protocol between two nodes. When the buffers of a destination NCP become blocked, a signal is sent to the source NCP, indicating this state. The source NCP will then cease transmitting packets until the elapse of a time-out period. Thereafter another attempt is made to transmit the data.

The source NCP will indicate the blocking of its destination NCP buffers to its supervisory processor. The supervisor will subsequently ensure that no packets are transmitted to the source NCP by the other nodal modules, until the elapse of the time-out period, this being conveyed to the switch by the source NCP module.

The above discussion needs to be further clarified. Packets requiring to make use of the said NCP are divided into groups. The groups are classified according to topological attributes of the network (this aspect is further dealt with in Chapter 5). Each NCP contains a number of buffers dedicated to each of the groups; in addition a common buffering area is provided for use by all classes. Blocked buffering at a destination NCP usually means that the buffers belonging to a particular group are all occupied. The NCP therefore ceases the transmission of a particular class of packets (conveyed to the supervisory processor), whilst other classes are sent as normal.

The implication of use of this measure is that data flow of different groups enroute from a number of different source-destination nodes through the same intermediate node, will not cause excessive interference with each other. If this measure were not included, it would be possible for traffic enroute from one source-destination link whose service rate was poor and intensity high, to block traffic from other links passing through the same NCP.

2.6 Review

This chapter has described the resource-sharing aspect of packet-switching networks. It was pointed out that a data adaptor unit cannot be considered in isolation, but forms part of a network. The characteristics of such networks, and the subsequent need for flow control measures to prevent performance degradation, were described. Both global and local flow control measures have been incorporated. The global technique operates by limiting the number of packets per logical link in transit. It is implemented by means of the end-to-end protocol. Two local flow techniques have been incorporated in the switching processor:

1) the allocation of different time-dependent priority increments to entry and transit traffic,
2) control of the transmission of packets to the network communication processors.

CHAPTER 3

SIMULATION FACILITY

3.1 Introduction

The object of the simulation program is to model the interactions of the flow control measures and protocols that cause packets to be transported from one end of the network to the other. The performance and characterisation of a network could then be determined as a function of the control measures and protocols.

It was found that simulation could be conducted either by utilizing available packages, or by designing a model in a high-level language.

A general Purpose Systems Simulator (GPSS version 3) was available. GPSS is based on the modelling of events. Entities such as packets, messages, jobs, customers, etc., are modelled as passing through a queueing system. A certain amount of time was spent experimenting with this package. It was found to be unsuitable as the emphasis of the work lay in modelling the mechanisms of the network that cause data transmission to take place, rather than in the entities or data packets themselves.

The high-level languages available were FORTRAN, PL/1 and Pascal. In terms of language structure and the implementation of data types, Pascal and PL/1 were favoured, but only a compiler was available for each. In contrast, a FORTRAN interpreter and compiler existed.

The turn-around time for batch jobs could be considerable at the computer system where the simulation program was developed. Accordingly, the model was written in Fortran, on the basis that the advantages to be gained by writing and debugging the program interactively, outweighed those to be gained by using the data structure of Pascal or PL/1.

The simulation model developed is non-mathematical. It simulates the transition of states in the protocols for ensuring synchronization of sender and receiver, and the subsequent transmission of information.

The remainder of the chapter provides an overview of the simulation facility developed.

3.2 The Simulation Structure

This section is concerned with introducing the structure of the simulation program developed to study the proposed network. Extensive use of arrays was made to represent the various parameters and quantities of the network. The array descriptions are contained in Appendix A, whilst the program listing, together with appropriate documentation, is to be found in Appendix B.

The simulation model may be divided into two main sections, the initialization and the execution blocks, as in Figure 3.1.

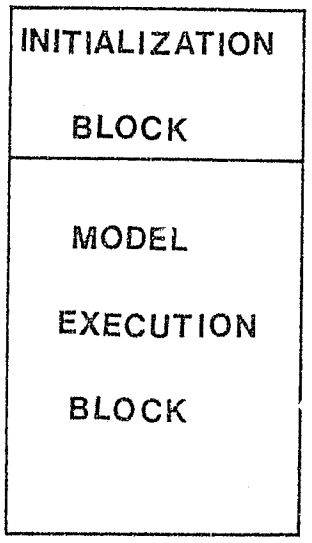


Figure 3.1 Main Simulation Structure

3.2.1 Initialization Block

The initialization block is concerned with reading in the appropriate network parameters, and initializing the array structures. In addition this block provides for initializing the execution block into a state of readiness for simulating the network. The execution block contains the necessary code for simulating the network on an event-by-event basis. The output of data describing the performance of the network is also included in this block.

3.2.2 Execution Block

The execution block may further be subdivided into a number of modules, each simulating a particular aspect of the network. These modules are shown in Figure 3.2.

3.2.3 Description of the Simulation Modules

The node-to-node communication module simulates the protocol used to transmit data between two nodes via the network communication processors.

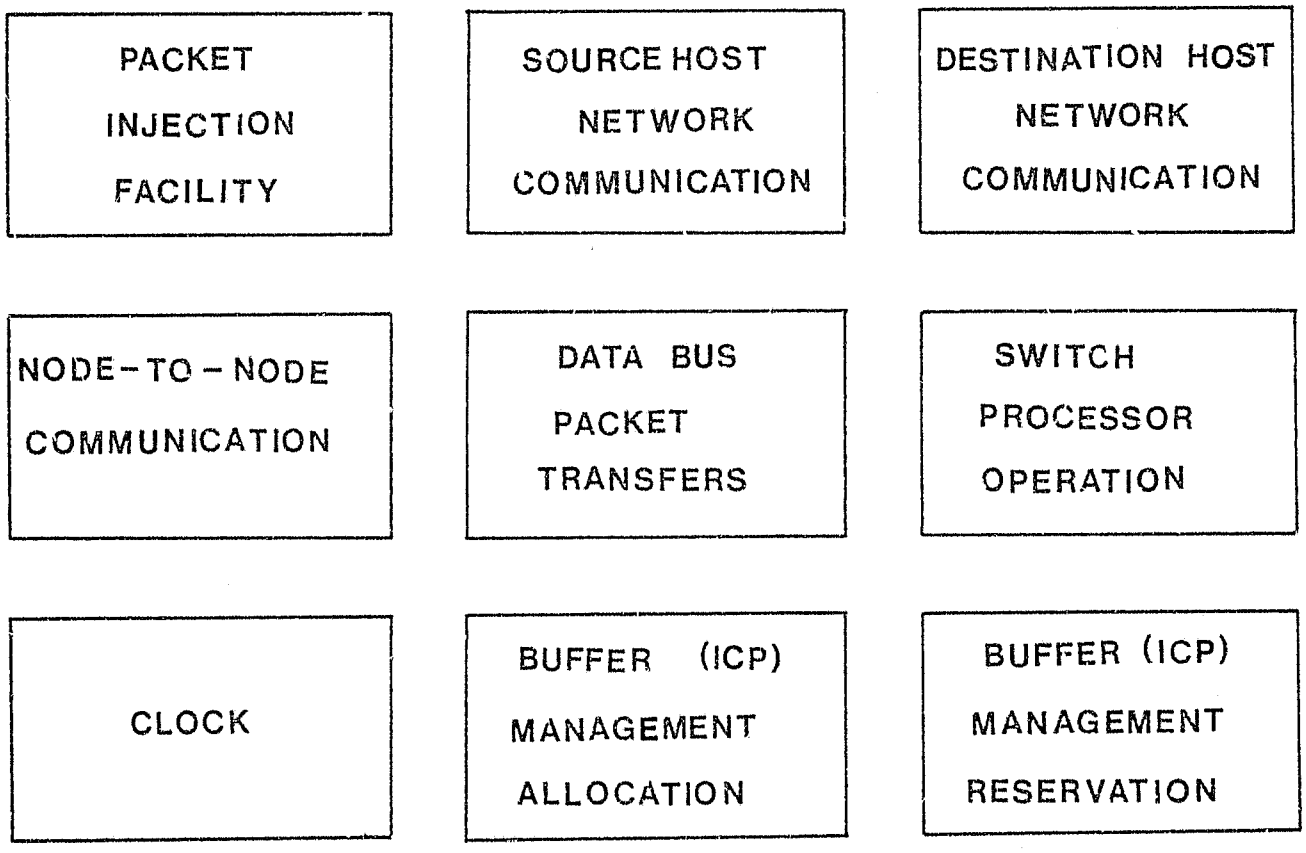


Figure 3.2 Simulation Modules

The communication between processors (ICP's and NCP's), and the packet transfers via the data bus within a node are modelled by the data bus packet transfer module.

The switching processor module contains the switching algorithms used to direct the nodal activity. This module simulates the activities of sampling the nodal status, processing the status data, and transmitting subsequent commands to the ICP's and NCP's.

The above modules constitute the network simulation components for modelling the signalling and flow control measures.

A number of additional modules have been included. Two of these, the buffer management allocation and reservation blocks, simulate the processes of allocating storage for message segments and packets in the ICP's. The allocation module models the granting of buffer blocks to logical devices; the reservation module maintains a list of requests for storage when the available ICP buffers are all occupied. The reservation module contains an algorithm that allows priorities to be allocated to different groups of messages, e.g., short interactive messages and long file transfers.

The packet injection facility generates the traffic to the network. The facility is used to generate data messages and packets with Poisson interarrival times and geometric or exponential length distributions.

Finally, the clock module is used as a timing mechanism for events. It keeps track of all events, and activates the relevant mechanisms in the main simulation section when the event occurs. This module maintains a clock using real numbers to represent time.

The structure of the simulation is such that the above modules may be placed in any order. Valid simulation of the network is not dependent on the order in which the modules are executed. This has been done specifically to allow additional modules to be incorporated if it is desired to simulate aspects of the network in greater detail

than represented here; or to simulate other protocols and flow control techniques.

The format of the program is shown in Figure 3.3. The clock and buffer management modules are subroutines. The remaining modules, constituting the execution block, are grouped together into a single loop.

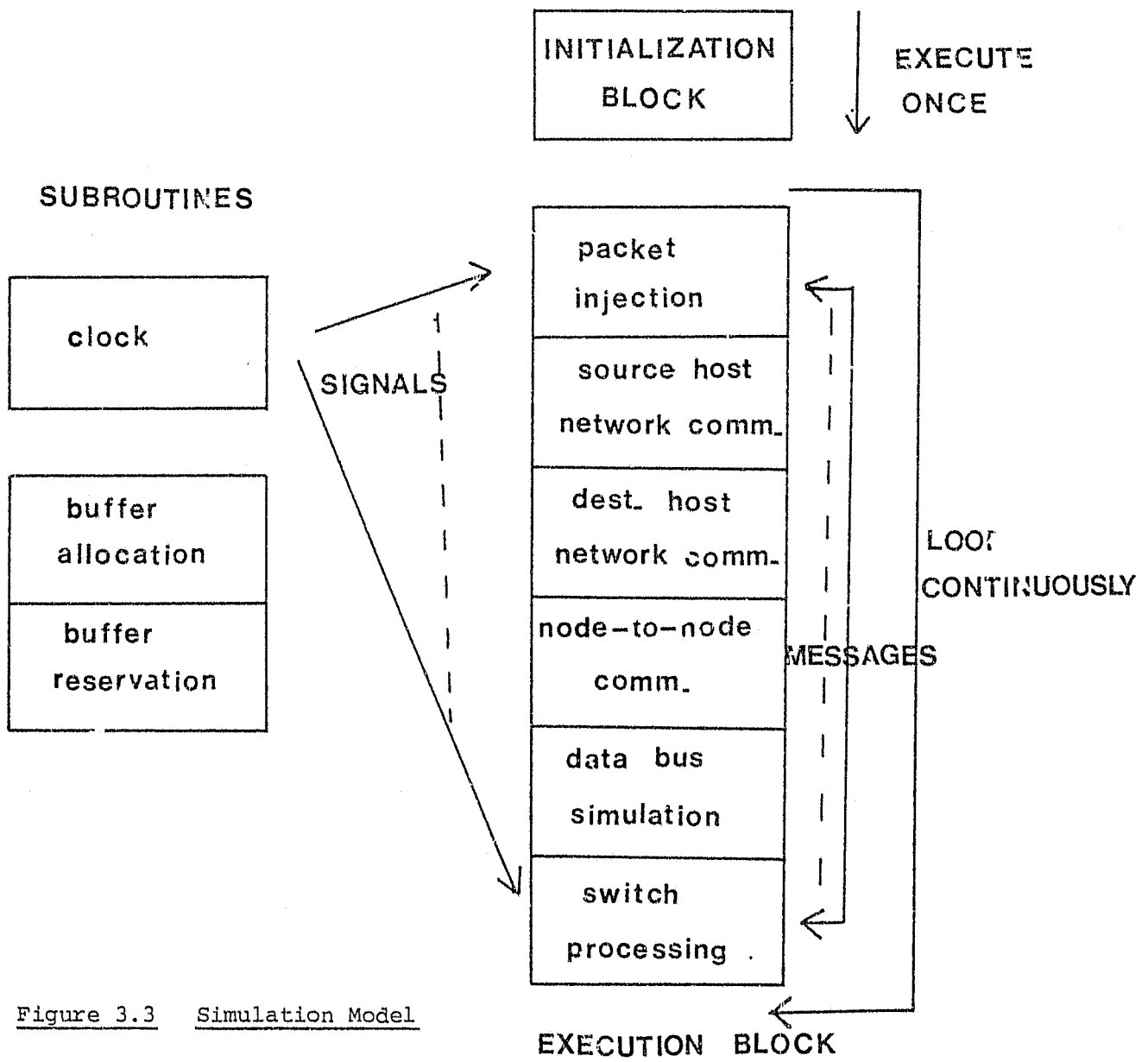


Figure 3.3 Simulation Model

A double execution of the loop is termed a 'cycle'. The duration of the simulation is directly controlled by the number of cycles specified for the loop. To determine the performance characteristics of the network (see Chapter 7) initial runs were first conducted to determine the number of cycles required for the model to reach a state of equilibrium. These tests are conducted for different input loads.

3.2.4 Module Communication

Each module is an autonomous unit, simulating a particular aspect of the network. The following terminology, i.e., 'message' and 'signal', refer to the simulation itself.

The intercommunication between modules takes place via messages or signals. A signal is transmitted from one module to another when an event is to take place or has occurred. A message constitutes information passed from one module to another. The characteristics of a packet passing through the network may be passed between modules via messages. An example of signal use occurs when a packet is transmitted across a communication channel; such an event takes a certain amount of time. When the packet has reached the destination, i.e., use of the channel is terminated, then this event is indicated to the appropriate module via a signal generated by the clock module.

Signalling takes place between the clock module and the modules constituting the execution block. The clock enables each module to regulate the simulation of its own events relative to the events occurring in other modules with reference to the same time scale. An event

occurring in module n may cause an event to take place in module n + m, or V.V., such causal action being transferred between two modules via a message or signal (see Figure 3.4). It is important to note that such action implies that the execution of the code for the event in module (n + m) must take place in the same instant of simulated time as the events of module n. The code is therefore scanned more than once during each cycle, each cycle depicting a point in simulated time. Consider Figure 3.4(b). During the first scan a message is passed from module n + m to module n; during the second scan, events in module n, activated by the message, are simulated. The example in Figure 3.4(a) requires only one scan, the second having no effect. In the simulation model, two scans per cycle are sufficient.

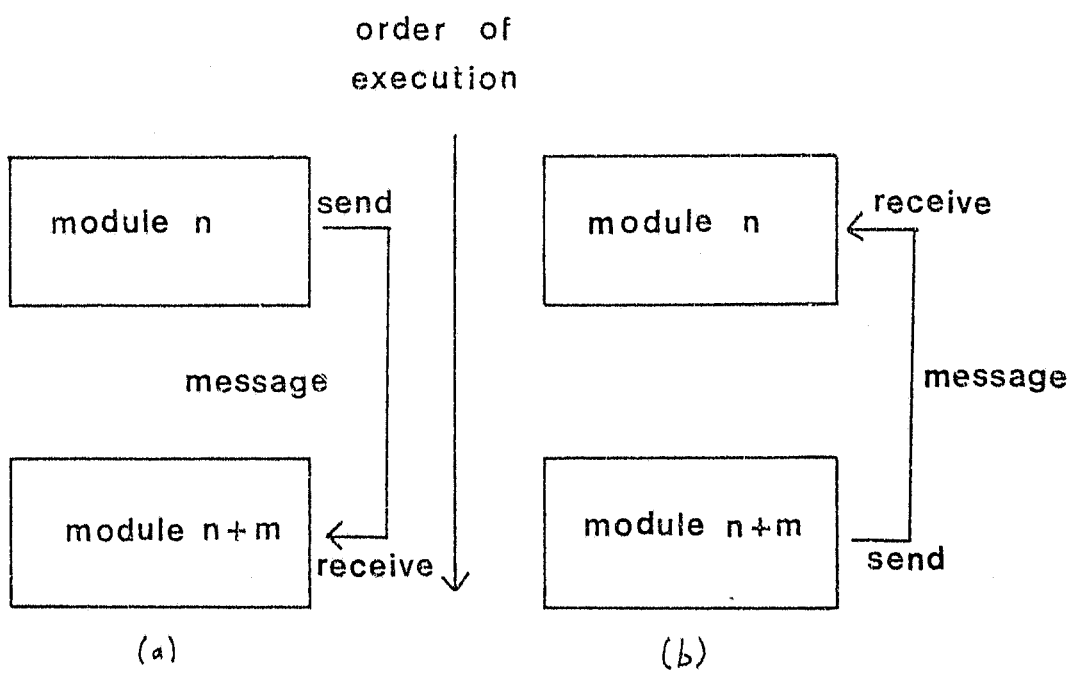


Figure 3.4 Message Passing

3.3 Input Parameters for the Simulation

This section deals with the data that is required to define a network to be simulated. The parameters that define network topology and operation are considered. Description of the input parameters will take place with reference to the example shown in Figure 3.5. The reasons for choosing this topology are considered in Chapter 7, where its characteristics are described.

3.3.1 Network Topology Simulated

The topology consists of five nodes connected via PCM channels. Hierarchically-based networks consist of nodal clusters. The simulated topology approximates such a cluster. Simulation of a network with more than five nodes would require an excessive amount of computer time to determine the network characteristics. Each node contains a number of channel controllers that control the transmission of packets between nodes. There is a total of fourteen such controllers (also referred to as network communication processors). Each node contains eight interface communication processors that enable the network to interact with host devices. Fifty logical links and fifty virtual circuits have been implemented. A logical link refers to the connection between two communicating host devices. The network resources, e.g., lines, buffers, are allocated on demand.

3.3.2 Traffic Level Control by Virtual Circuits

In order to raise the level of traffic within the network without having to resort to excessive use of computer time, the concept of a virtual circuit has been defined. A virtual circuit refers to the

Ni : channel or network control processor

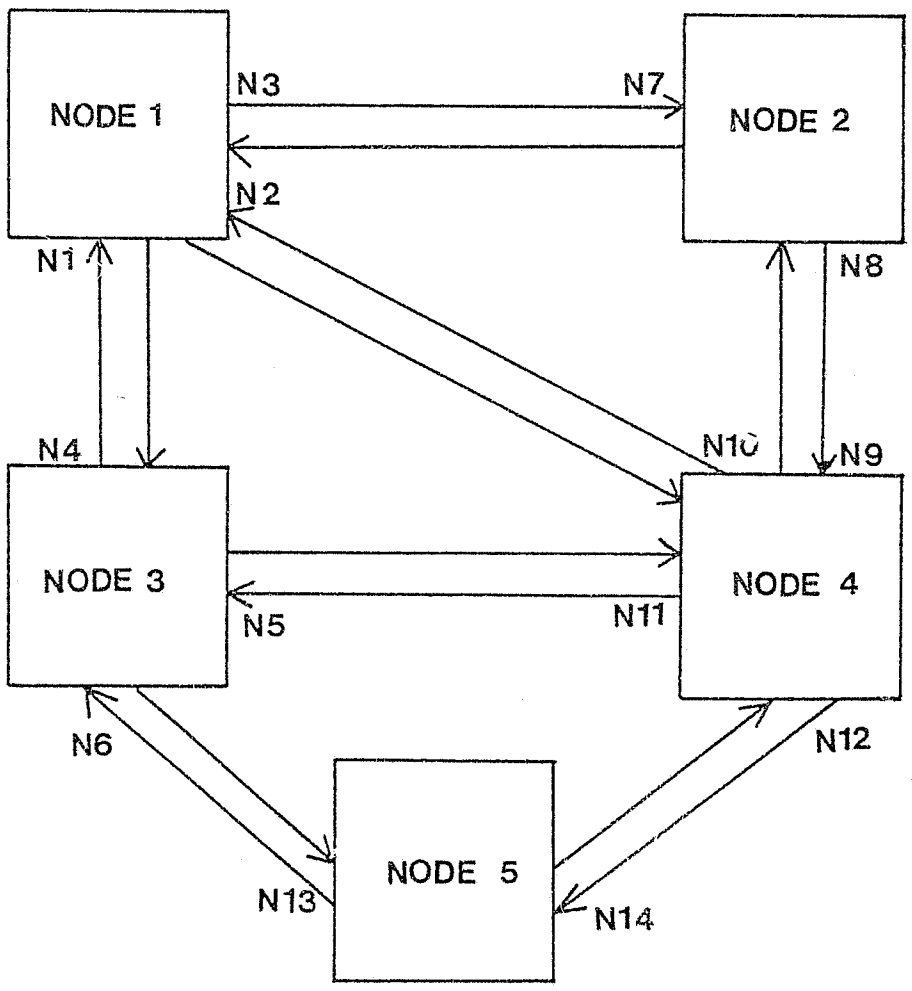


Figure 3.5 Network Simulation Topology

link existing between a source ICP and destination ICP. Packets are generated at a given rate at the source ICP, transmitted via the network according to a fixed routing matrix, and sunk by the destination ICP. No simulation of host-network protocols is done. The level of traffic in the network and in particular along each route may be controlled by

varying the mean packet generation rate and length. The distributions of interarrival time and length have been chosen to be Poisson and geometric or exponential, respectively.

3.3.3 Link Specifications

The parameters for defining logical links and virtual circuits are shown in Table 3.1. Each link and circuit is identified by a number. For each link it is necessary to specify the source node - and ICP numbers, and the destination node - and ICP numbers. Each ICP, NCP and node is identified via an integer. The values may be identical for the above three groups, e.g., ICP and NCP may both be identified by the integer 3, the context will distinguish between them.

The source and destination channel capacity slots refer to the capacity of the connection between user device and network in bits per second. The source and destination channels may be of different line rates.

The length and arrival slots refer to the mean values of the statistical distributions incorporated in the main program to generate packets or messages.

Finally, the contents of the slot denoted by 'NO MSGS' enables one to limit the number of packets or messages generated for the duration of the simulation. This feature is extremely useful for debugging purposes. A negative value in this slot causes an unlimited number to be generated.

Link No	Source Node	Source ICP	Dest. Node	Dest. ICP	Source Channel Capacity	Dest. Channel Capacity	Mean Length	Mean Arrival	NO MSGS
1	1	1	1	7	7200	7200	450	10	10
⋮									
50	5	40	5	34	7200	7200	450	10	10

(a) Logical Link Table

Link No	Source Node	Source ICP	Dest. Node	Dest. ICF	Source Channel Capacity	Dest. Channel Capacity	Mean Length	Mean Arrival	NO MSGS
1	1	1	1	7	-	-	450	10	-5
⋮									
50	5	40	5	34	-	-	450	10	2

(b) Virtual Circuit Table

Table 3.1 Logical Link and Virtual Circuit Parameters

3.3.4 Node-Processor Mapping

The ICP's and NCP's forming part of a particular node must be specified as shown in Table 3.2

Node No	ICP No							
1	1	2	3	4	5	6	7	8
⋮								
5	33	34	35	36	37	38	39	40

Mapping: Node-to-ICP

Node No	NCP No			
1	1	2	3	0
⋮				
5	13	14	0	0

Mapping: Node-to-NCP

Table 3.2 Nodal Mappings

A zero in a slot simply indicates an empty location. A unique mapping exists between each node and the NCP's and ICP's it contains.

The topology is formulated by specifying which two NCP's are linked. Table 3.3 shows, for example, that NCP1 is linked to NCP4. The result is a connection between node 1 and node 3. The fixed topology produced is that shown in Figure 3.5.

Source NCP	1	2		13	14
Dest. NCP	4	10		6	12

Table 3.3 NCP-to-NCP Connection

3.3.5 Packet Route Map

The route that packets are to follow is implemented via a nodal route map, as shown in Table 3.4. Each packet will be a member of either a logical link or virtual circuit, the source and destination ICP's of which will have been specified as in Table 3.1.

		Dest. Nodal No.				
		1	2	3	4	5
Present Nodal No.	1	0	3	1	2	1
	2	7	0	7	8	8
	3	4	4	0	5	6
	4	10	9	11	0	12
	5	13	14	13	14	0

Table 3.4 Route Map

Routing functions are performed under the control of the supervisory processor in each node. By consulting the route map, the supervisor can direct a packet to the relevant NCP, once the present and destination

nodal positions are known. If the present and destination nodal positions coincide, then the supervisor knows that the destination node has been reached, and that the packet must be directed to the relevant ICP.

For example, a packet at node 2, having as destination node 5, will be directed to NCP8 (of node 2). Referring to Figure 3.5, the packet will be transmitted to NCP9 of node 4. From node 4 it will be sent to node 5 via NCP12 and NCP14, situated within each of the above nodes.

In the event of there being more than one channel between two nodes, an additional map for each extra channel is to be specified. A choice then exists as to which channel the packet is to be directed to. The choice has been based on assigning the packet to the NCP with the smallest queue.

3.3.6 Bus Multiplexing Table

A table is needed to specify the number of packets multiplexed on the data bus of each node, every period. No attempt will be made to define further this concept; these parameters have been included for the sake of completeness.

Node	1	2	3	4	5
NO PCKS MUX	11	10	1[illegible]	12	10

Table 3.5 Bus Multiplexing

3.3.7 The Partition Table

Packets enroute from one node to another may be grouped into a number of classes. These classes are a function of the network topology. Buffer blocks in the NCP are dedicated to each class. The term partition refers to the class of buffers set aside for a specific group of packets. Partitions have been implemented to prevent the traffic flow enroute to one destination from blocking flow enroute to other destinations. Such would occur if one class of traffic were to occupy all buffers in an NCP. See Table 3.6.

In addition, a node plot matrix of partitions must be provided to enable the supervisory processor to classify the packets into classes. If the buffers allocated to a particular class of packets are all occupied, then the supervisor will ensure that no packets belonging to that class will be transmitted to the NCP with blocked buffers. See Table 3.6.

Consider NCP 1 of node 1 (see Figure 3.5). Packets passing through NCP 1 will be enroute either to node 3 or node 5, such may be verified by referring to the packet route map. Accordingly, partitions denoted by 3 and 5 are set aside in NCP1. The simulation program provides for up to eight possible partitions. The node plot matrix enables the supervisory processor at node 1 to classify the packet with destination node 5 to be 3.

NCP's	Partitions							
	1	2	3	4	5	6	7	8
1	3	5	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0
⋮								
14	2	4	0	0	0	0	0	0

Processor Partition Matrix

Destination Node

Source Node	1	2	3	4	5
1	0	2	3	4	3
2	1	0	1	4	4
3	1	1	0	4	5
4	1	2	3	0	5
5	3	4	3	4	0

Node Plot Matrix

Table 3.6 Partition Table

3.3.8 Network Parameters

Finally, Table 3.7 contains a set of parameters that may easily be altered for a given topology to study the network performance. The parameters contained in the table are all variables, unlike the above input data (which are constants), and may be changed during execution.

3.3.9 Summary

The defined parameters specify the topology of the network, the number and types of processors in each node, the logical links and virtual circuits for generating traffic, and the route map for packets.

Parameter	Description
N	Total number of logical links and virtual circuits
NUMBER	Duration of simulation run in cycles
SCAN	Number of loops of main block per cycle
BUFBLK	ICP buffer block size in bits
NSOURC	Number of buffer blocks allocated at ICP per logical link
SGMENT	Message segment size in bits
NONDS	Total number of nodes in network
NOICPS	Total number of ICP's in network
NONCPS	Total number of NCP's in network
PSZNO	Packet size in bits
MNOBFS	Number of buffer blocks in NCP - send or receive
BITLNG	Packet overhead in bits
MAXCH	Maximum number of channels between any two nodes
NOI	Maximum number of processors in any node
NOPCKS	Maximum number of packets that model can simulate
CMAX	Number of switch cycles per period
NMAXIC	Maximum number of ICP's in any node
NMAXIN	Maximum number of NCP's in any node
QUS	Total storage locations provided for algorithm execution
TXMAX	Retransmission interval for incorrect packets
PXMAX	Partition retransmission interval
MVXRT	Data bus rate in bits per second
PMEANC	Mean packet interarrival time for virtual circuits
SWPROC	Estimated switch algorithm execution duration
NVC	Number of virtual circuits
NRDN	Number of packets generated - exponential length, arrival time
NRRN	Number of packets generated per instant for virtual circuit
NODPRC	Estimated NCP processing time per packet
NPZ	Estimated processing time for data bus transmission

Table 3.7 Network Parameters

Parameter	Description
	SUBROUTINE ACCESS
BLOCK(I,5)	Number of single packet blocks (ICP) - memory
BLOCK(I,6)	Number of multipacket blocks (ICP) - memory
BLOCK(I,7)	Number of multisegment blocks (ICP) - memory
BLOCK(I,8)	Number of blocks of common memory (ICP)
	SUBROUTINE RESERVE
NPRSIN	Single packet priority increment (ICP)
NPRMUL	Multipacket priority increment (ICP)
NPRSOU	Source segment priority increment (ICP)
NRSV	Number of locations for requests

Table 3.7 (Contd.) Network Parameters

Each of these parameters is addressed by a subscript of a particular array. The size of the arrays allowed is dependent on the amount of storage available to the computer user. The Network Simulation was run on an IBM System/370 using a 512K memory partition.

The above method for defining a network topology has been tested and proven by implementing and simulating four different networks. The input data listings of three of the networks, the Network Simulation, the Nodal Simulation, and the Interface Simulation are defined in Appendix B. The fourth topology, not listed, was used to develop and debug the simulation.

3.4 The Nodal Simulation Model

The main execution block consists of a number of modules that are used to simulate the operation of a node. The nodal simulation model is illustrated in Figure 3.6. Shown is the interaction between the various modules. The block structure is a reflection of the nodal architecture as reproduced in Figure 3.7.

3.4.1 Simulation of Nodal Replicas

The simulation network is structured by producing a number of replicas equivalent to that of the model shown in Figure 3.6. The input parameters, discussed in a previous section, are used to connect these replicas so that a network topology may result. The number of replicas required is equal to the number of nodes simulated.

The relevant control measures are contained in the simulation blocks of the nodal model, and these are required to communicate with the corresponding measures implemented in other nodal models.

For example, the source-to-destination protocol is implemented in the source and destination ICP's. The simulation of this protocol is contained in the source-host-network and destination-host-network blocks of each nodal model. The simulation of a logical link therefore requires the two nodal models, one for the source ICP and the other for the destination ICP, to communicate. This communication takes place via the transfer of the simulated packets from one ICP to the other, or alternately between one nodal model and another.

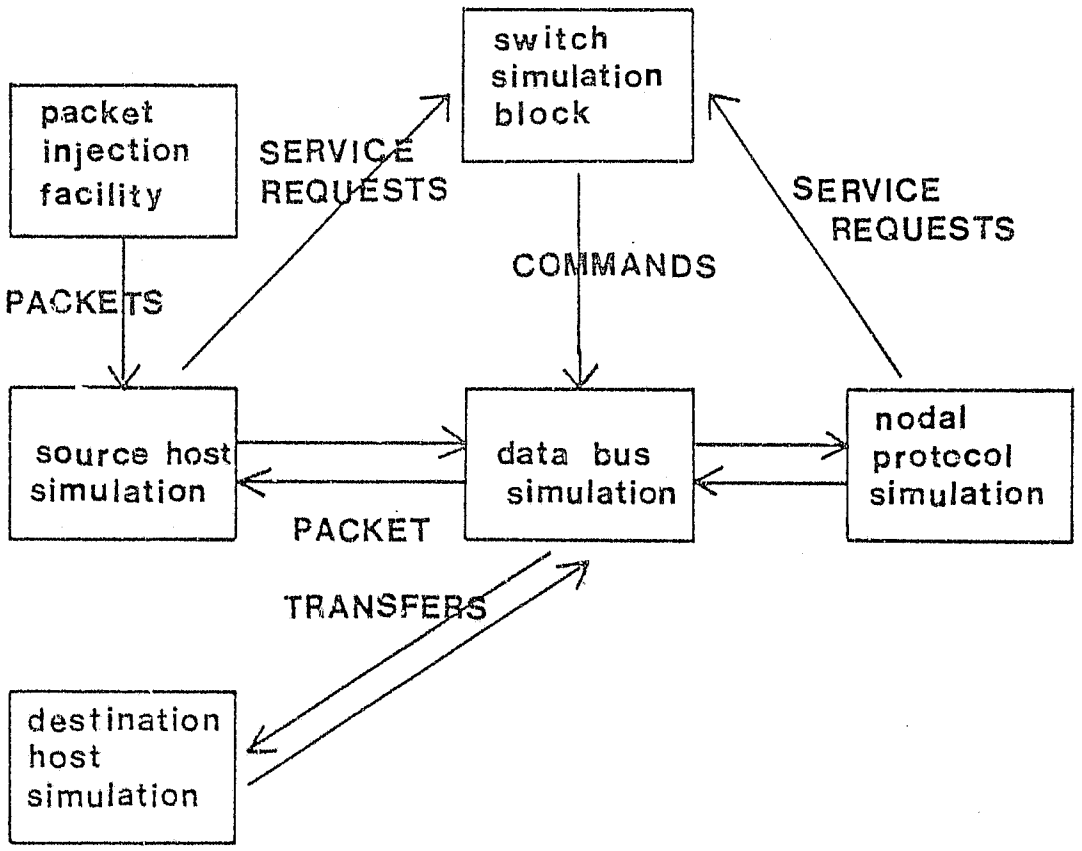


FIGURE 3-6 NODAL SIMULATION MODEL

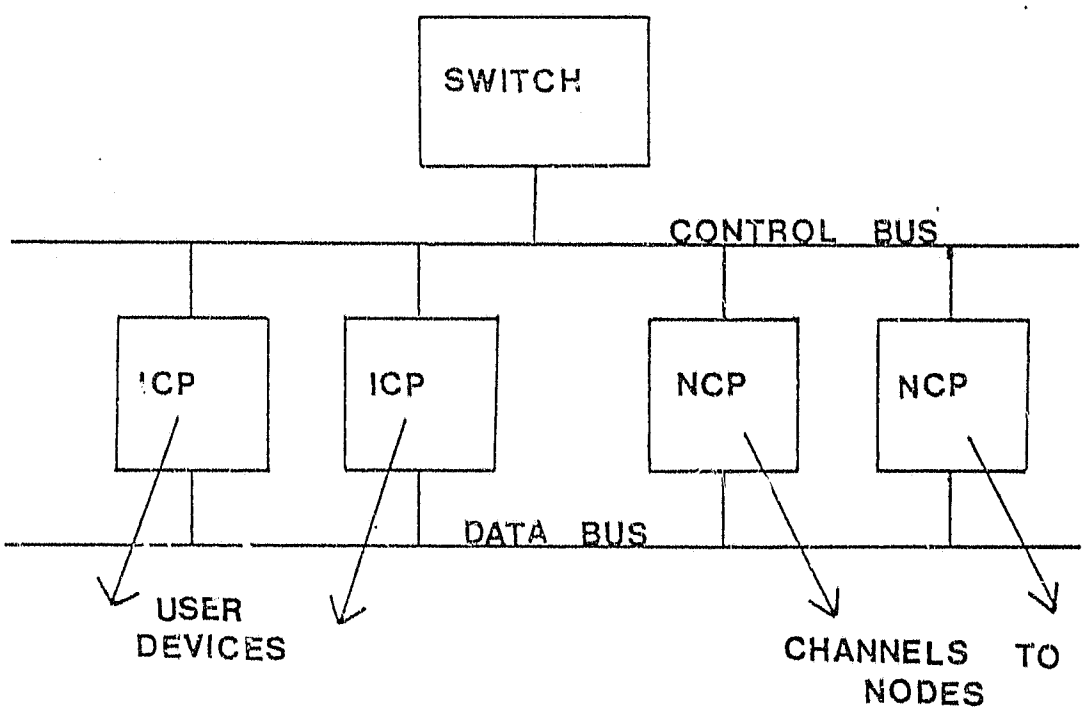


FIGURE 3-7 NODAL ARCHITECTURE

The simulation model is very much simplified compared to practical networks, but it nonetheless attempts to reproduce the essential features of traffic control in a network, i.e., the simulation models a number of nodes, each containing the necessary protocols and flow measures, communicating with each other so as to transport packets.

3.4.2 Description of the Block Structure

Referring to Figure 3.6, the packet injection facility generates the messages and packets that are used to simulate the transmission of data to the network by logical links and virtual circuits. The source-host-network block receives these packets and messages. In the case of logical links the user-network protocol is simulated. The source host block formats the packets. The destination-host-network block simulates the destination user protocol, including the reassembly function. Both these blocks contain the source node-destination node protocol for controlling the flow of data between two communicating user devices.

The nodal protocol block simulates the signals exchanged between two NCP's linked via a common channel. All requests for packet transmission via the data bus are sent to the switch simulation block. The switch block processes the requests and transmits commands to the data bus simulation block. This block then transfers the packets between the following blocks: source host-network, destination host-network, and nodal protocol. It is noted that packet transfers between each of the above three blocks may take place in either

direction, since both data and control packets, e.g., acknowledgements, are simulated.

3.4.3 Implementation of Models Using Arrays

The above structure may be easily implemented in FORTRAN. The nodal simulation model is replicated for each node in the network; the source host-network and the destination host-network protocol blocks are replicated for each ICP, and the nodal protocol block for each NCP in the network. The switch and data bus simulation blocks are implemented for each node. These are all exact replicas, and by making use of arrays, each block need only be coded once. The various simulation entities are identified by the appropriate array subscript, hence the need for numbering each ICP, NCP and node via the input parameters.

The following generalization concerning the simulation may now be made. The messages, transferred between nodal models, or, analogously between nodes, are the actual packets. Included are both the data packets generated by user devices, and control packets used by the nodes for traffic control purposes. The messages exchanged between blocks, within a nodal module, include packets and simulated messages. A typical example of a simulated message is the service request by an ICP or NCP to the switch; in this case, data describing the request is transferred from one block to another by writing into the location of a specific array. The switch block periodically examines the array contents and processes any data found.

3.5 Review

This chapter has provided an overview of the simulation model structure. The primary simulation modules used and the means for module intercommunication were described. The input parameters that determine the nature of the network to be simulated were dealt with at some length. Finally a description of the interaction of the modules constituting the facility has been included.

This overview thus provides an idea of the nature of the facility. The network topology and parameters may be experimented with, but a change in nodal architecture or protocol requires a module replacement. Provided the module interaction rules are adhered to, such may be achieved.

A severe limitation of the facility was found to be the extensive amount of computer time required before the model reaches a state of equilibrium. This is apparently so in the simulation of all complex models with a queueing-type structure.[53] Use of the facility to determine performance of a network was on the whole satisfactory.

CHAPTER 4

THE INTERFACE COMMUNICATION PROCESSOR

4.1 Introduction

The nodal architecture consists of three units, the channel controller, the switching processor, and the interface communication processor. This chapter will outline the role of the interface communication processor or ICP in the data network.

It is the function of the ICP to provide an interface between the user environment and the packet network. The interface is required for two reasons. It is used to connect different machines to the network, e.g., teletypewriters, terminals, and computers. This means that facilities must be provided for allowing two otherwise incompatible machines to communicate, e.g., translation of user codes to a single standardized network code. The ICP is also used to isolate the packet network from the user environment. Access to the network is controlled by the interface and switching processors. Unrestricted access to the network could result in performance degradation.

An ICP is envisaged to be microcomputer-based with facilities for data, as opposed to computational processing, e.g., bit, byte, and block manipulation. Users are connected to an ICP via a serial link with line rates from 50 to 9600 Baud. Lines with rates from 9600 to 64000 Baud are best connected to the network via a channel controller as described in the next chapter. In this case, packets and not messages must be transmitted by the user (the distinction is detailed in Section 4.3).

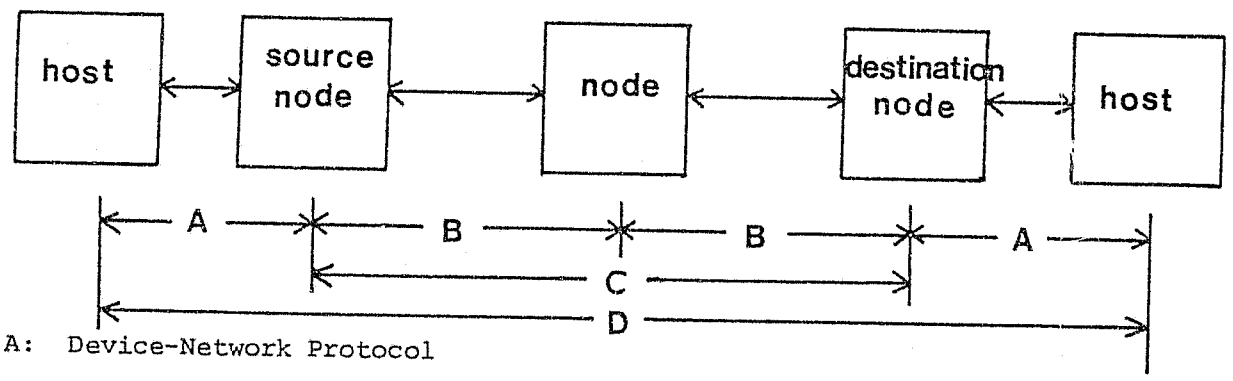
To prevent excessive overhead, identical user machines would be connected to a single interface processor. A node may therefore be connected to a large number of different network users by utilizing several ICP's each with software appropriate for interfacing to a specific type of machine. It is noted that such a machine may be a multiplexor or computer. This entails the use of a single physical line by possibly several logical processes.

A user will transmit variable length message segments to an ICP. These segments may not exceed a specified maximum size; in the simulation the size is 4000 bits. The segments are divided into packets (size 400 bits) by the ICP, and transmitted through the network to a destination ICP. The destination ICP is required to reassemble the packets back into the original segment, the segment subsequently being sent to the destination host. In this network no provision is made for ensuring that packets of a given segment arrive in sequence at the destination ICP. The ICP must therefore sort these into the correct order; information for reassembly is included in the packet header. In addition, the ICP's must regulate data flow between each other, and between the network and users. A protocol has been defined for ICP's to maintain correct data flow.

The ICP thus serves as a type of 'front-end' processor to the network. It interacts with both user and packet network, receiving and translating messages into suitable formats for subsequent transmission to either hosts or the network.

The network transmission formats are shown in Figure 4.1. The host protocol is device dependent. The node-to-node protocol has the function of ensuring that packets are transmitted correctly between two nodes. The source-destination node protocol serves to regulate the flow of data between two ICP's. The device-network protocol interacts closely with the source-destination node protocol to control the amount of data admitted to the network.

The succeeding sections detail further the user-network protocol, the source-destination node protocol, and the buffer management technique, incorporated into the ICP.



A: Device-Network Protocol

B: Node-to-Node Protocol

C: Source-Destination Node Protocol

D: Host Protocol

Figure 4.1 Network Protocol

4.2 User-Network Protocol

Due to the close interaction between the device-network and source-to-destination protocols, both are described here.

The network should be capable of transmitting both short (20 to 1000 bits) and long messages (10^4 to 10^7 bits). For interactive users with short messages low delay is important, whilst for the transfer of long data files high throughput is necessary. The goals of transmitting both long and short messages in optimal fashion are, however, contradictory. For low delay, a small packet size is necessary to cut transmission time and to shorten queueing times; for high throughput a large packet size is necessary to decrease overhead in bits per second and the processing overhead per bit. Long queues may be needed to provide sufficient buffering for full circuit utilization. It is therefore difficult to satisfy both goals, and a compromise must be made.

For this packet switching network it has been decided to emphasize the delay aspect, but at the same time the protocol devised does allow long file transfers to take place. If the messages sent by a particular device are generally very lengthy (greater than 10^6 bits) and the response time is to be low, it is perhaps better to employ a dedicated link between the machine and the destination.

Figures for response and throughput performance indices of the network cannot at this stage be given; these are examined in subsequent chapters. Presented here is the protocol enabling devices to communicate via the network; performance is determined by the

network mechanisms and is influenced by packet and segment sizes.

The outline of the protocol for controlling data flow between two devices, via a logical link, is shown in Figures 4.2 and 4.3 for single and multisegment transmissions respectively. Table 4.1 gives the meanings of abbreviations used in Figures 4.2 and 4.3. It is noted that these signals are essentially logical signals; their implementation will vary from device to device, e.g., a RFNS (#n)(Ready for Next Segment process number n) signal may be a complex control packet to a computer, or a simple asynchronous TTY eleven bit command word. The interfacing of different devices to a network is a complex matter. These signals form the basis of a flow control structure that should be general enough to be expanded and implemented in an ICP for any particular situation that may arise.

Key to Figures 4.3 and 4.4

Sj	:	segment number j
CALLES (#n)	:	call signal for process number n
TRSM (#n)	:	call signal acknowledgement
ACK (#n)	:	acknowledgement signal between device and network for data received
Sj (EOM)	:	final segment in message
SD (EOM)	:	acknowledgement signal for message transmitted (Source-dest)
RFNS (#n)	:	Ready for next segment (process number n)
TRM (#n)	:	Transmit next segment, destination node ready
(#n)	:	Indicates logical link process numbers
EOM	:	End of message

Table 4.1 Notation for Transmission Figures

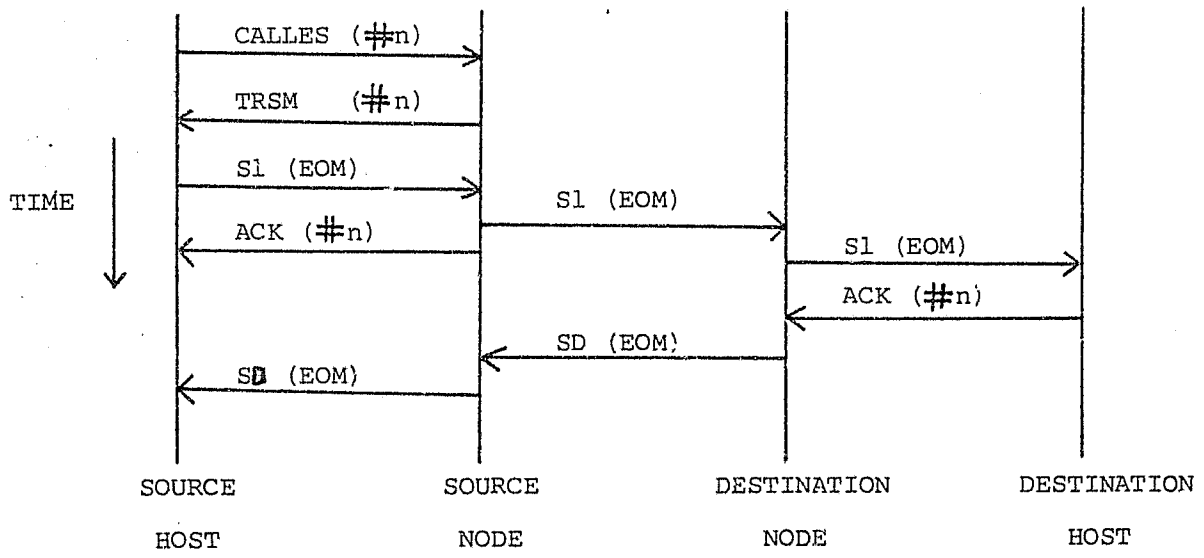


Figure 4.2 Single Segment Transmission

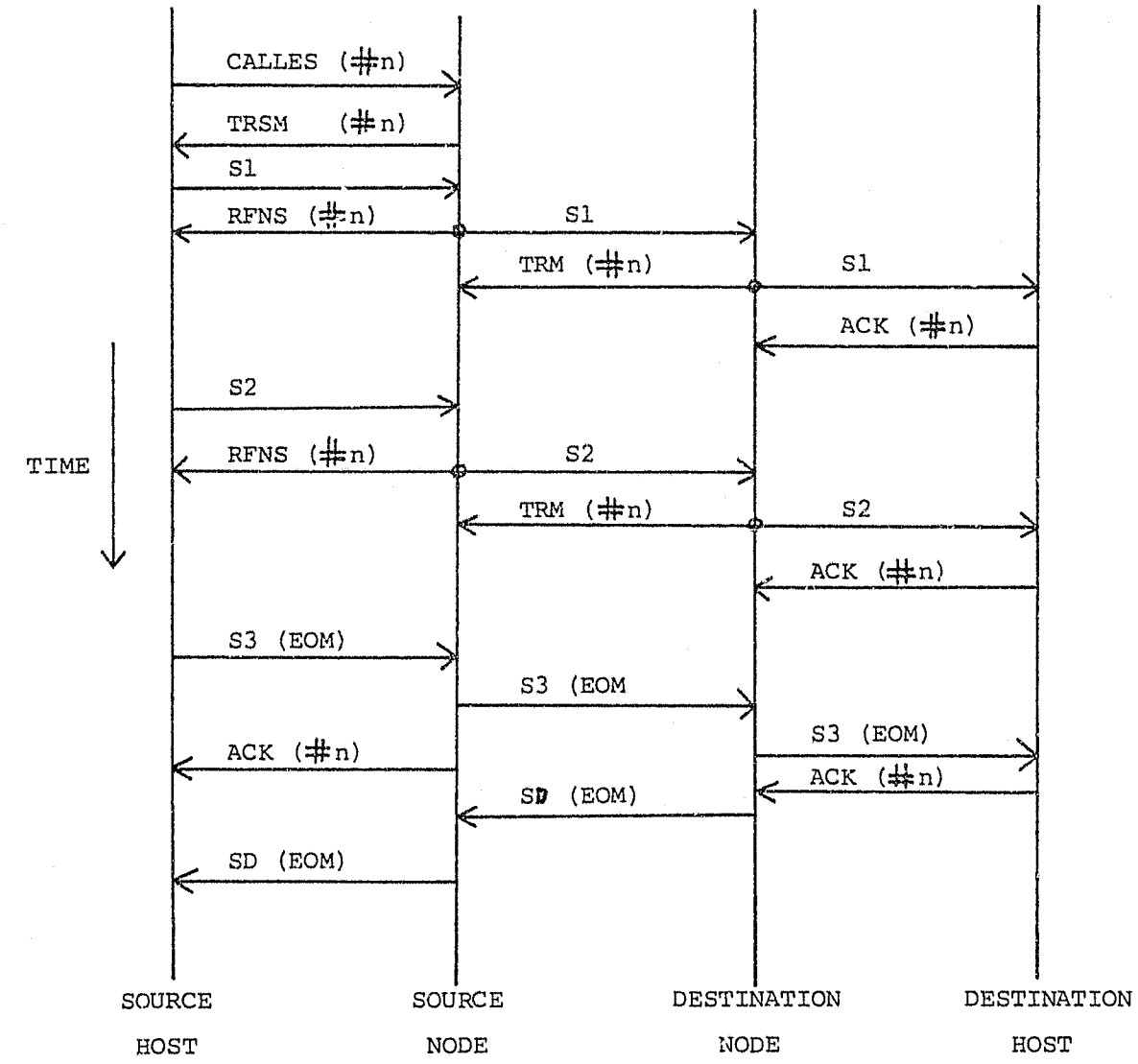


Figure 4.3 Multisegment Transmission

4.2.1 Source Host to ICP Interaction

A device or process requiring to transmit data informs the relevant ICP of its request via a 'CALLES' signal. The ICP provides the necessary buffers to receive the data and returns a 'TRSM' signal to the calling device. A segment limited to some maximum size is forwarded to the ICP by the device.

To ensure that no data is lost due to insufficient buffering, the transmitting process must divide long messages into segments; dividing messages only affects those users sending long files. The segment, when received by the ICP is processed (e.g., code translation) and subdivided into packets of smaller length. For obvious reasons the packet and segment sizes chosen should be such that a segment consists of an integral number of packets.

As soon as these packets are ready for network transit, a 'RFNS' signal is sent to the caller, provided an End of Message (EOM) was not indicated in the preceding segment just received. The caller will will subsequently transmit its next segment to the ICP.

By using double buffering, a source ICP will be receiving segments and transmitting packets through the network at the same time for a specified logical link. The second and subsequent segments, when stored and processed, will be transmitted only when a 'TRM' signal is received by the source ICP from the destination ICP. This signal may or may not have arrived by the time the second segment is ready. Once 'TRM' is received and the segment is ready, the segment (in the

form of packets) is sent to the destination ICP and a 'RFNS' is sent to the device. The above process is then repeated for successive segments until an 'EOM' arrives.

4.2.2 Destination Source ICP Interaction

The destination ICP works on the same principle utilizing double buffering. When a segment received at the destination ICP has been reassembled from its packets, it is transmitted to the destination host, at the same time a 'TRM' signal is sent to the source ICP. The 'TRM' signal serves to synchronize the packet flow between source and destination ICP. The 'ACK' signals the destination ICP of correct segment reception. Once the 'TRM' signal has been sent to the source ICP, no further 'TRM' signals are sent until the destination ICP receives an 'ACK' from the destination host and a segment from the source ICP.

4.2.3 Synchronization of Processes

Figure 4.4 depicts the synchronization tools used for communication processes. These processes take place in four separate machines, two host and two nodal processors, and formulate a single logical link.

The 'wait-for-and-receive' segment and the 'send' segment blocks indicate that these two processes are to occur concurrently.

The scheme shown in Figure 4.4 is not a flowchart indicating program execution. An explanation of the send and receive blocks is

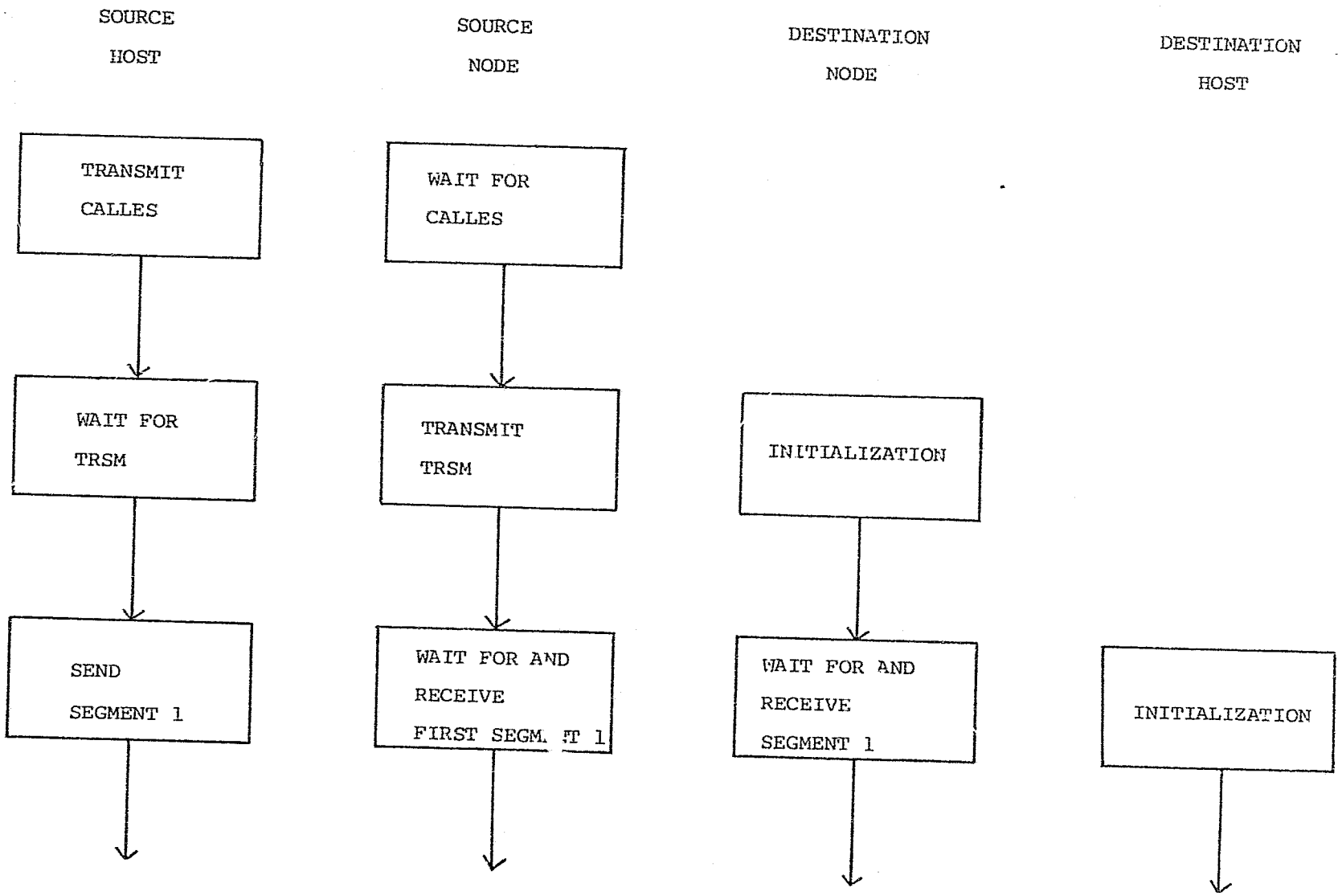


Figure 4.4(a) Synchronization of Communicating Processes on a Logical Link

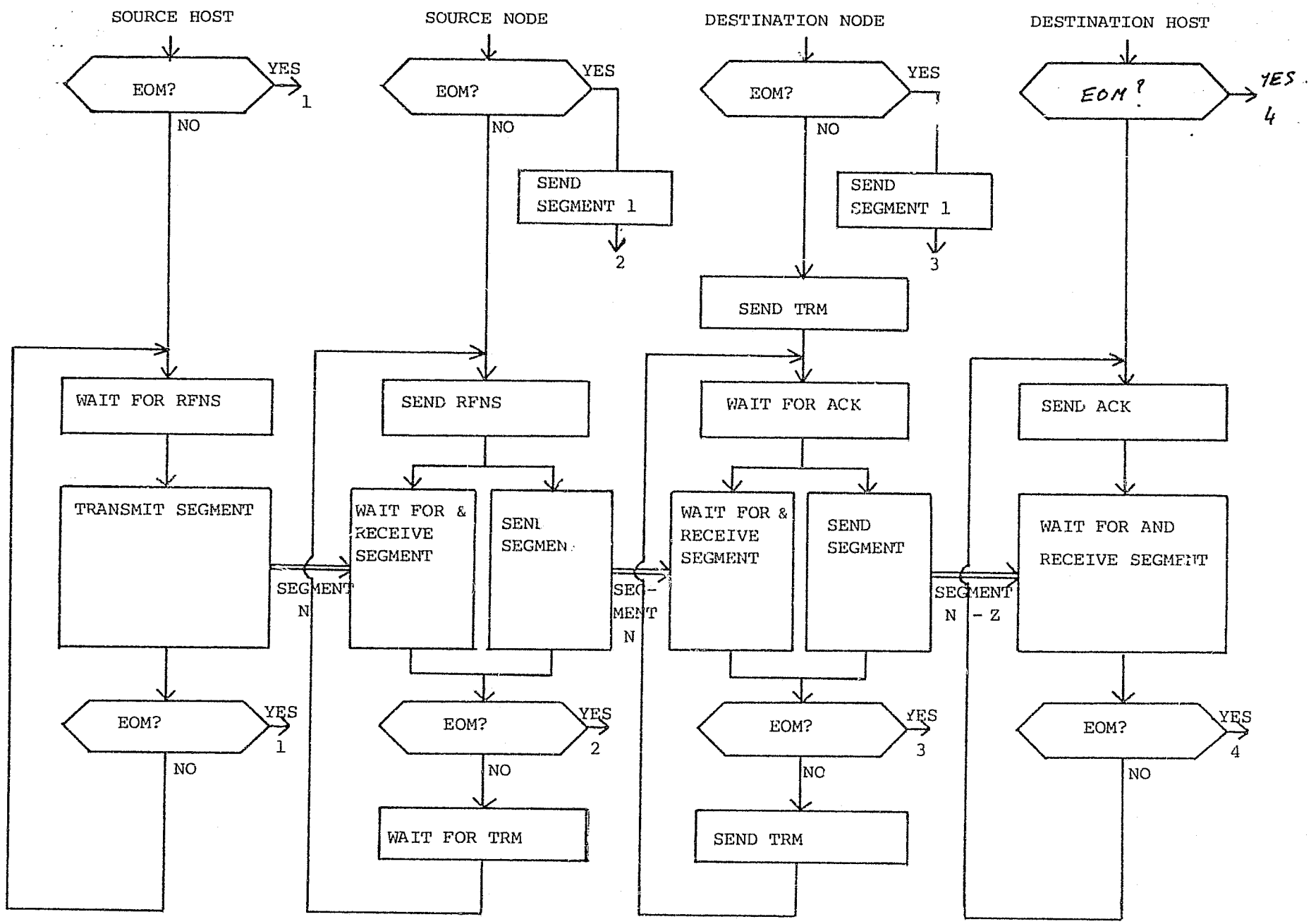


Figure 4.4(b) Synchronization of Communicating Processes on a Logical Link

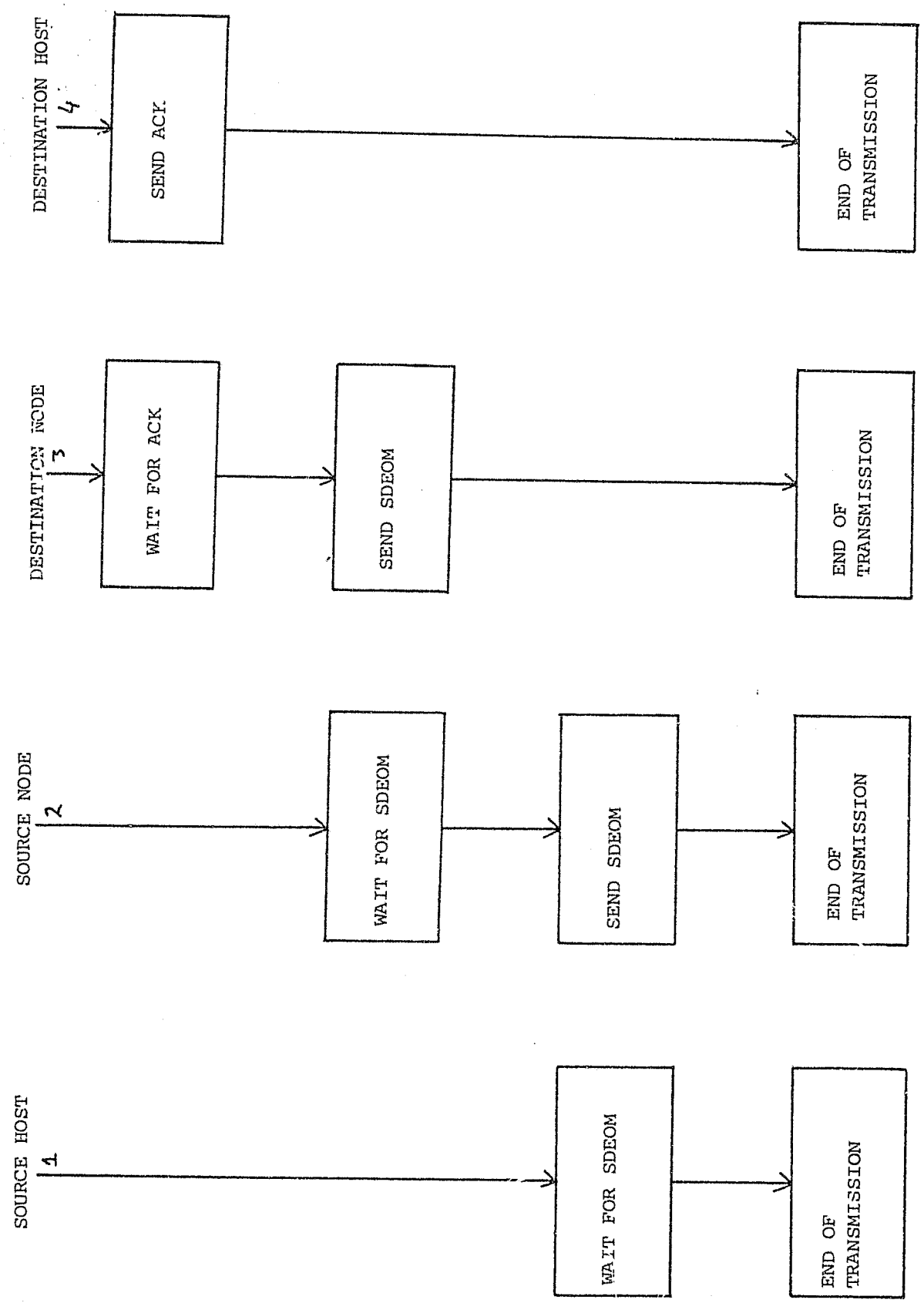


Figure 4.4(c) Synchronization of Communicating Processes on a Logical Link

necessary; the rest of the schematic is intuitively obvious. The send and receive blocks illustrate a pipeline for transmitting data segments through the network.

The following is an explanation for the source node in Figure 4.4. After a CALLES has been received and a TRSM signal transmitted by the source node, the node waits for and receives the data segment. Provided the segment is not the last one of the message, the node transmits a RFNS signal to the source host for the next segment and, at the same time, sends the segment just received to the destination node. The node subsequently waits for the segment (from the host) to arrive and stores the characters. As soon as the segment from the host and a TRM signal from the destination node has been received, the cycle starts anew with the source node transmitting a RFNS to the host and the recently arrived segment to its destination. The cycle is ended when the last segment in the message (EOM) arrives. It is noted that these communication processes take place in the interface processors of the nodes.

4.2.4 Pipelining Data Through the Network

The flow control protocol serves to limit the number of packets transmitted through the network by a logical link at any one time. For normal operation, at any one point in time, one segment will be entering the network, one will be in the process of leaving, and one is in transit through the network - assuming that the message is of length greater or equal to three segments. This aspect is demonstrated in Figure 4.5. Since the network is able to transmit

information at a higher rate (64000 bits per second) than the serial links connecting the devices to the network (50 to 9600 Baud) the time gaps between successive segment transmissions are small - the time to transmit a RFNS signal via a serial link, provided the TRM signal has already arrived. Near continuous transmission between devices is possible.

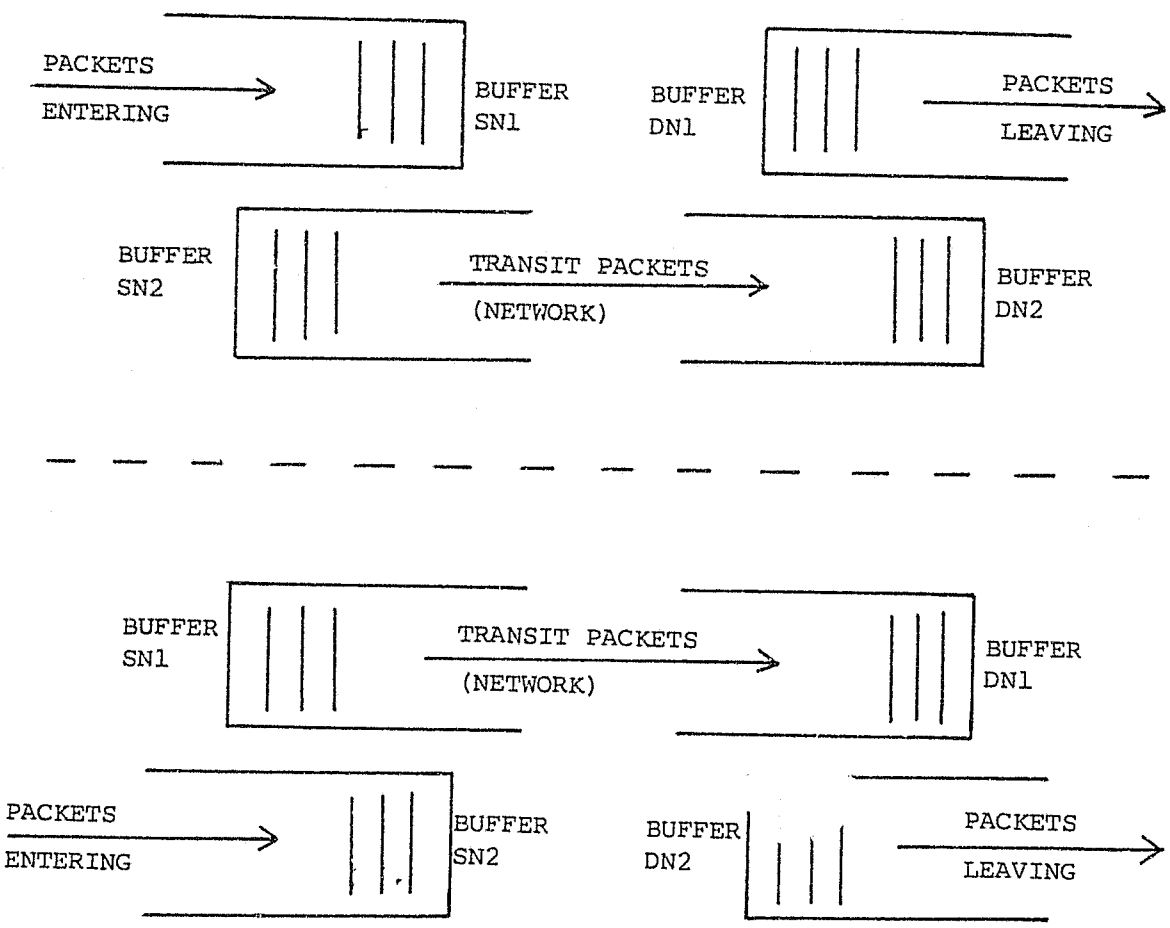


Figure 4.5 Pipelining Messages via Double Buffering

4.2.5 Termination of the Transmission

When an EOM is reached, an acknowledgement is transmitted by the source ICP to the source host indicating correct segment reception. It is noted that the RFNS and TRM signals used previously are acknowledgement signals by default. Bit errors in the segment transmitted to the ICP by the source host results in a negative acknowledgement signal being sent, requiring the retransmission of the segment. Correct packet transmission within the network is the responsibility of the channel controller (see Chapter 5).

The segment bearing an EOM header is sent to the destination ICP, and subsequently to the host. The data transmission phase is terminated when the destination ICP transmits SDEOM to the source host, once the destination ICP has received the final acknowledgement from the destination host.

The logical connection is not terminated, however; a subsequent block of data may be transmitted at any time by proceeding through the above process again. Short interactive messages follow the same procedure; in this case a single segment is sent only.

4.2.6 Segment Sizes

It must be understood that the ICP's on both sides of the link have no way of predicting the length of the message to be handled. Accordingly, variable length messages have been divided into segments so that buffer management and flow control problems may be simplified. It is also noted that the size of the segment does not affect the

packet network operation, since the network handles smaller entities (packets) to keep delay times low.

The size of the segment should not be too large, the effect of line errors must be taken into account. Large segment sizes are needed to keep overheads down and throughput high, but line errors may result in the necessity to retransmit an entire segment if the protocol is to be maintained at an elementary level. This problem has been left open as it is essentially device and line dependent, wherein the segment size alternately selected is determined by the individual case. Segment sizes may differ from logical link to logical link.

4.2.7 Source ICP to Destination ICP Transmission

The ICP-to-ICP protocol interacts closely with that of the interface processor communication with a host. When the first segment of a message has been received by the ICP, and the segment divided into packets, one of the packets, denoted by the 'first packet', is transmitted to the destination ICP. This first packet has the function of informing the destination ICP of the impending transmission, and provides details to the ICP of the required number of buffer blocks. The destination ICP must determine whether the destination host is in a position to accept the call. If the destination ICP and host are free to receive the data message, the source ICP is informed and transmission may begin. If the destination ICP or host cannot accept the data, the source ICP, after receiving the appropriate control packet TRMNCK, will delete the segment from its buffers and signal

the calling process to attempt another call after some random interval of time.

The communication between source and destination ICP's occurs by way of control packets; if two ICP's already happen to be in contact, the control packet may be piggy-backed onto a data packet. In effect this occurs when the first packet of a message is sent to prepare the way for the rest of the message.

The above procedure is followed for multipacket messages. Single packet messages, i.e., messages of length less than or equal to that of the chosen packet size, are transmitted without initially reserving buffers at the destination ICP. A certain amount of delay is therefore to be expected for multipacket segments whilst the destination ICP allocates buffers. Single packets do not incur this delay, allowing low network transit times for short interactive data traffic.

4.2.8 Demonstration of the ICP protocols

Figures 4.6 to 4.9 have been included to indicate the pipeline effect and speed control for the ICP protocol.

Figure 4.6 illustrates multi-segment, Figure 4.7, single segment, and Figure 4.8, single packet transfers.

The synchronization of transmission is basically achieved as follows: initiation of a segment transmission along 'Dest Node to 'Dest Host' results in a TRMT signal sent to the source ICP. The TRMT

KEY TO GRAPHS
SH-SN: SOURCE HOST TO SOURCE NODE
SN-DN: SOURCE NODE TO DESTINATION NODE
DN-DH: DESTINATION NODE TO DESTINATION HOST
DH-DN: DESTINATION HOST TO DESTINATION NODE
DN-SN: DESTINATION NODE TO SOURCE NODE
SN-SH: SOURCE NODE TO SOURCE HOST

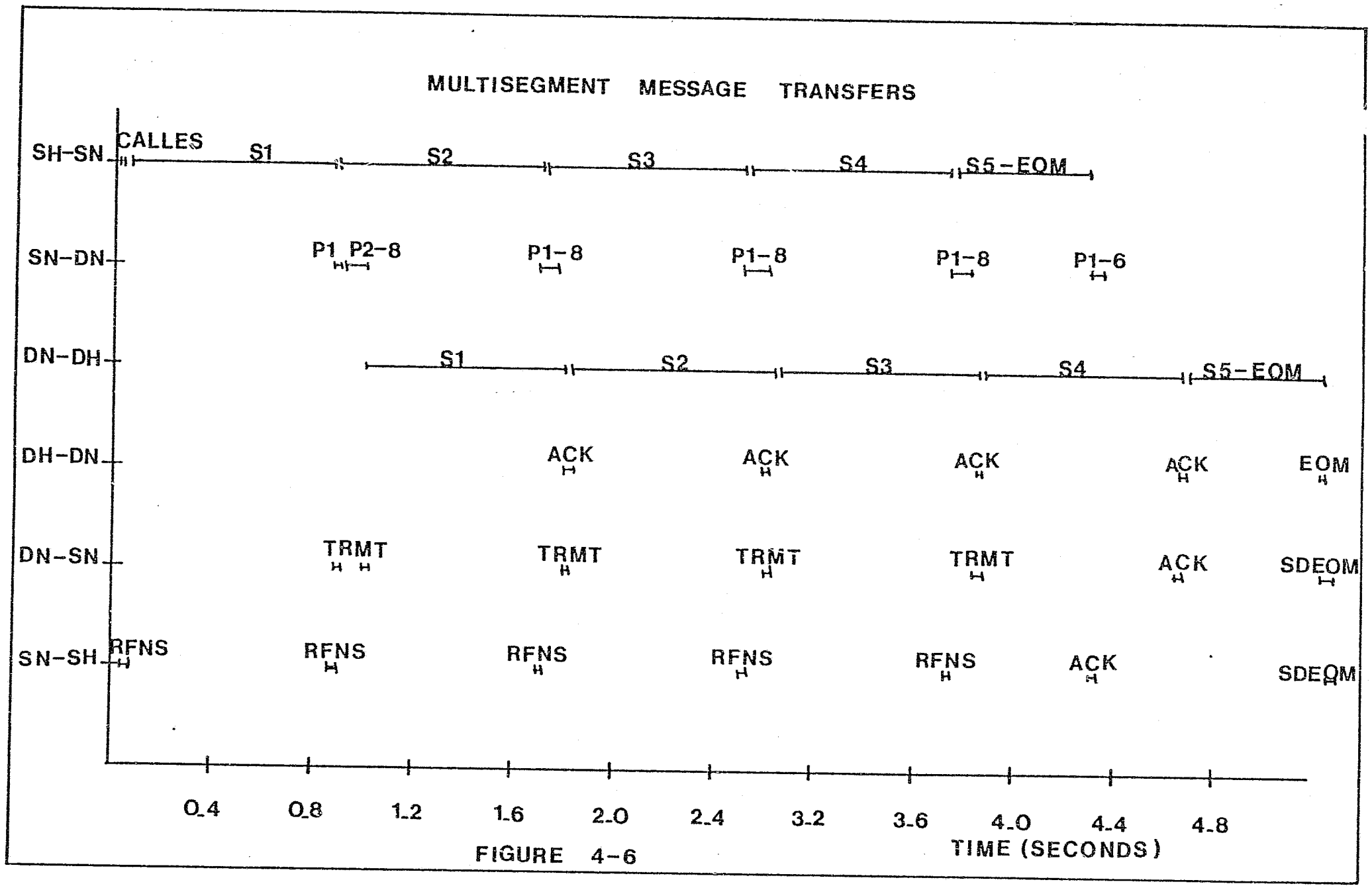

MULTISEGMENT MESSAGE TRANSFERS
SH-SN
CALLES
S1
S2
S3
S4
S5-EOM
SN-DN
P1 P2-8
P1-8
P1-8
P1-8
P1-6
DN-DH
S1
S2
S3
S4
S5-EOM
DH-DN
ACK H
ACK H
ACK H
ACK H
EOM H
DN-SN
TRMT H H
TRMT H
TRMT H
TRMT H
ACK H
SDEOM H
SN-SH
RFNS H
RFNS H
RFNS H
RFNS H
RFNS H
ACK H
SDEOM H
0.4
0.8
1.2
1.6
2.0
2.4
3.2
3.6
4.0
4.4
4.8
TIME (SECONDS)

FIGURE 4-6

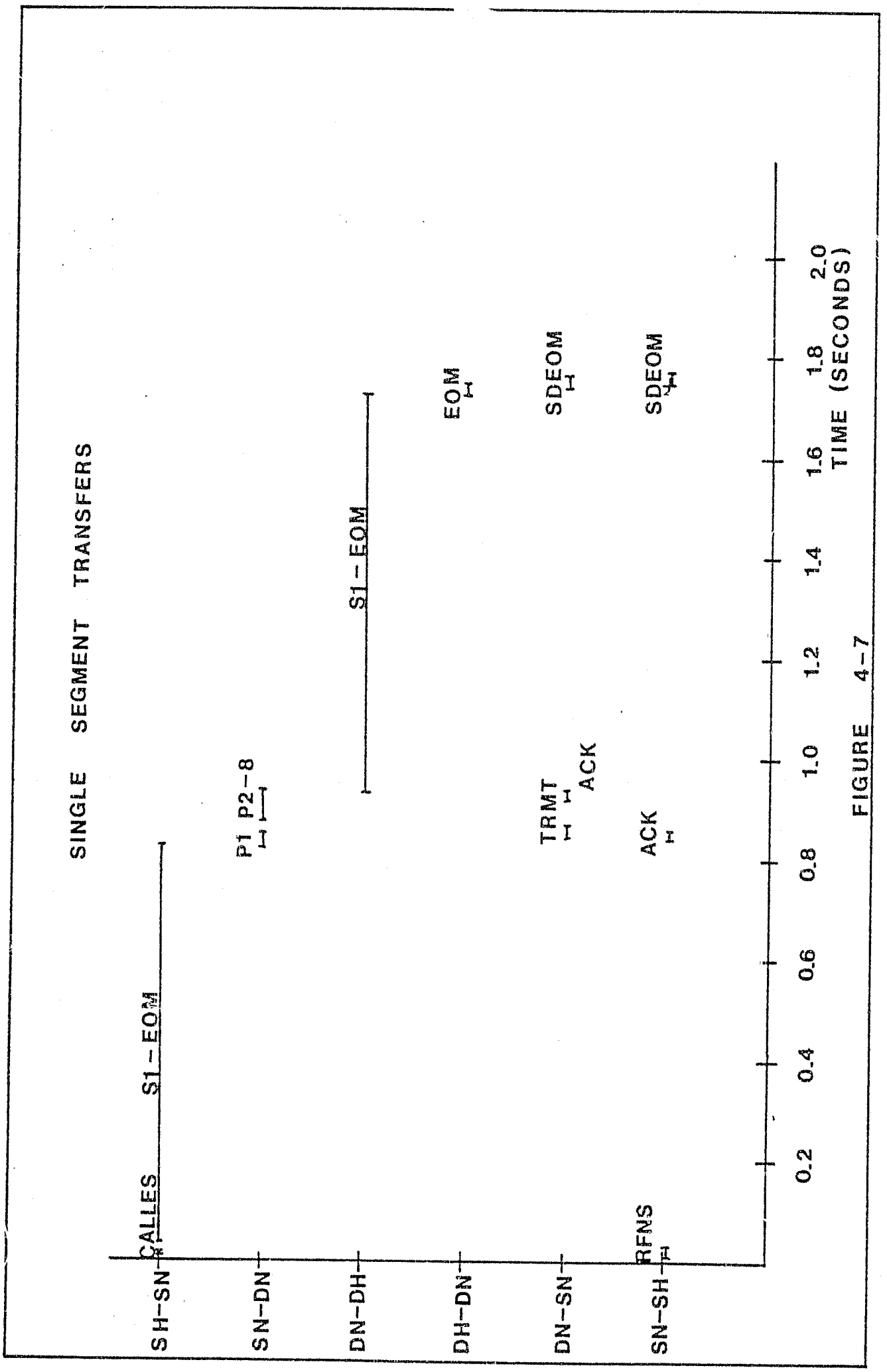

SINGLE SEGMENT TRANSFERS
SH-SN
SN-DN
DN-DH
DH-DN
DN-SN
SN-SH
CALLES
S1 - EOM
P1 P2-8
S1 - EOM
EOM
SDEOM
SDEOM
TRMT
ACK
ACK
RFNS
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
TIME (SECONDS)

FIGURE 4-7

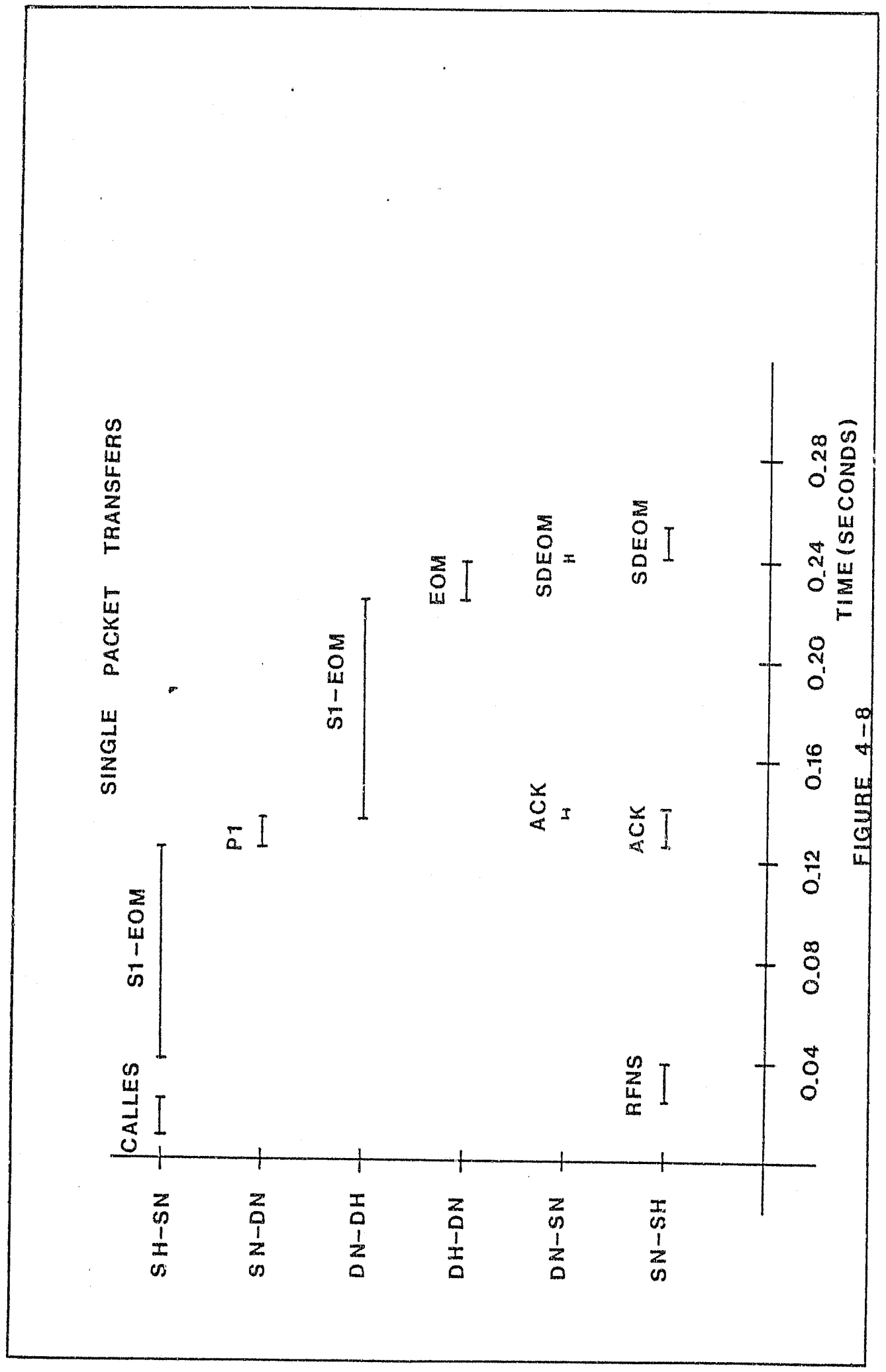

SINGLE PACKET TRANSFERS
CALLES
S1-EOM
P1
S1-EOM
ACK
ACK
RFNS
EOM
SDEOM
SDEOM
SH-SN
SN-DN
DN-DH
DH-DN
DN-SN
SN-SH
0.04
0.08
0.12
0.16
0.20
0.24
0.28
TIME (SECONDS)

FIGURE 4-8

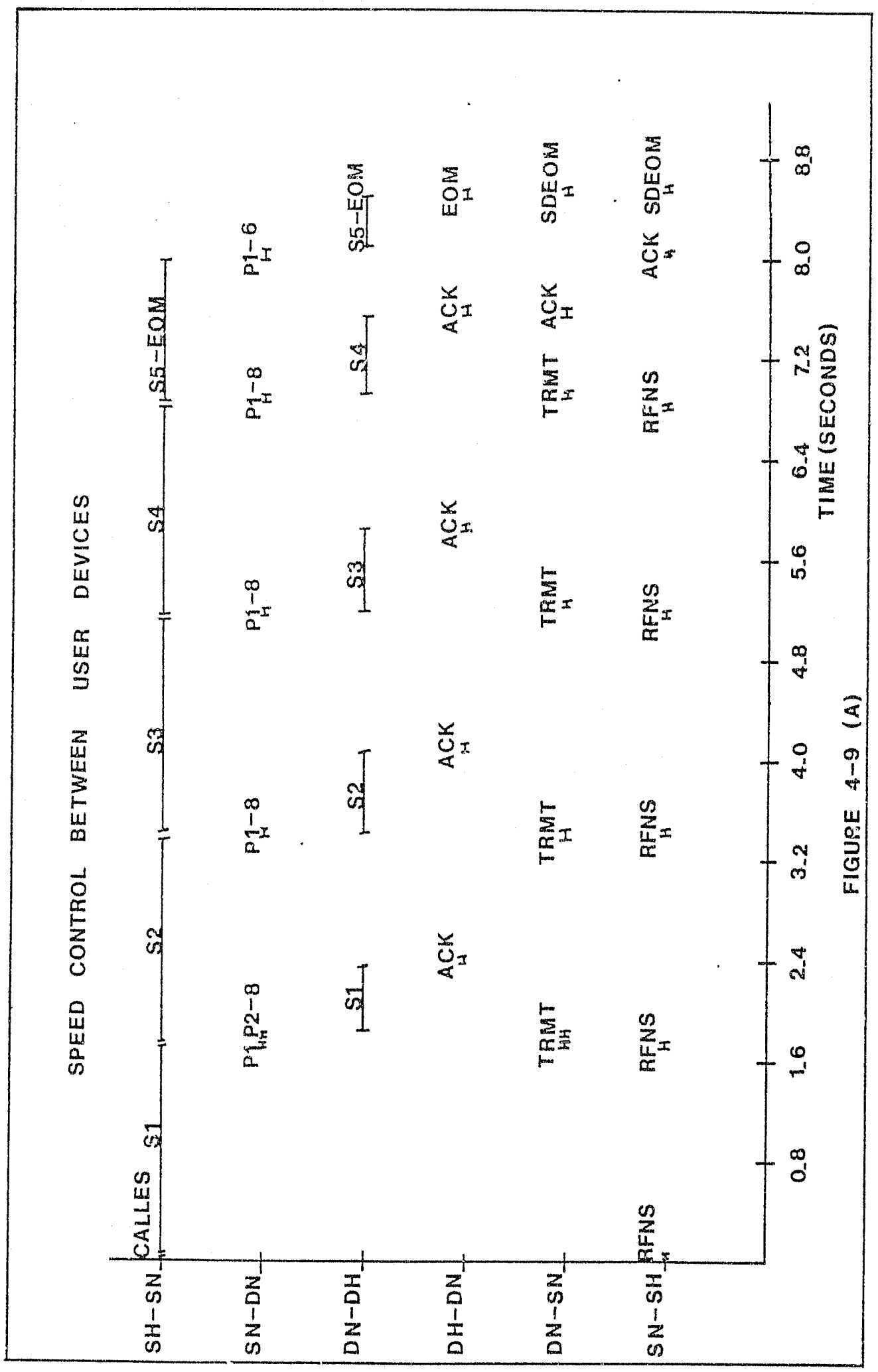

SPEED CONTROL BETWEEN USER DEVICES
SH-SN
SN-DN
DN-DH
DH-DN
DN-SN
SN-SH
CALLES
S1
S2
S3
S4
S5-EOM
P1-8
P1,P2-8
P1-6
ACK
EOM
TRMT
SDEOM
RFNS
0.8
1.6
2.4
3.2
4.0
4.8
5.6
6.4
7.2
8.0
8.8
TIME (SECONDS)

FIGURE 4-9 (A)

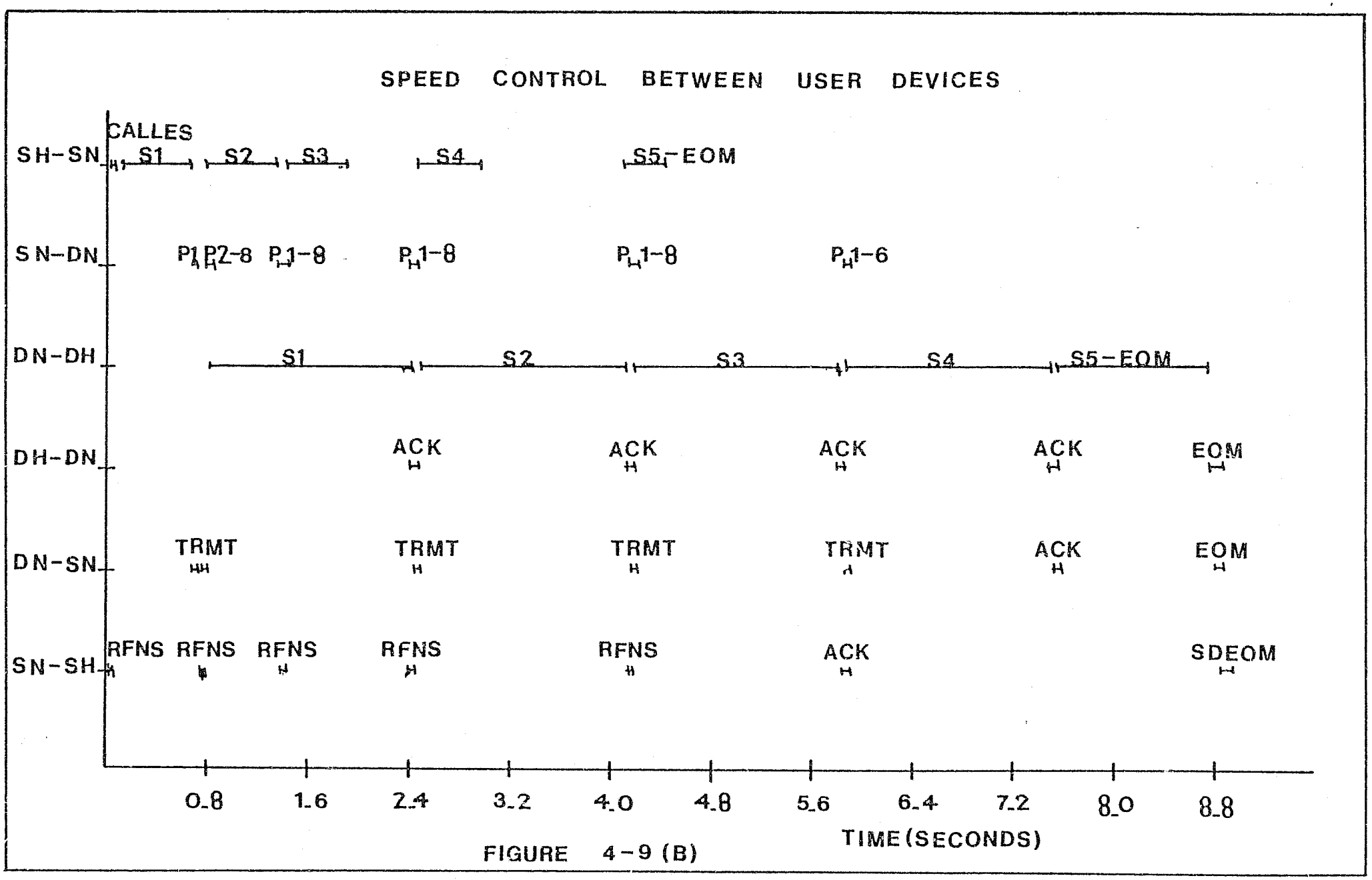

SPEED CONTROL BETWEEN USER DEVICES
CALLES
SH-SN
S1
S2
S3
S4
S5-EOM
SN-DN
P1P2-8
P1-8
P1-8
P1-8
P1-6
DN-DH
S1
S2
S3
S4
S5-EOM
DH-DN
ACK
ACK
ACK
ACK
EOM
DN-SN
TRMT
TRMT
TRMT
TRMT
ACK
EOM
SN-SH
RFNS
RFNS
RFNS
RFNS
RFNS
ACK
SDEOM
0.8
1.6
2.4
3.2
4.0
4.8
5.6
6.4
7.2
8.0
8.8
TIME(SECONDS)

FIGURE 4-9 (B)

signal is necessary before a source node may transmit a set of packets to the destination node. Segment transmission along 'Source Host to Source Node' is only possible if a RFNS is received from the source node and the transmission of the previous segment along 'Source Host to Source Node' has finished. A RFNS is generated whenever a set of packets is sent along 'Source Node to Dest Node', unless the set of packets belongs to the last segment in the message.

Figure 4.6 shows the typical effect of store-and-forward buffering on the host machine connected to the network, a shifting of the entire segment in time due to the delays incurred in network transit.

Figure 4.8 shows that single packet traffic may achieve low transit delay times provided sufficient buffering is available at the interface processors and the packet switching section is not congested.

Figures 4.9(a) and 4.9(b) illustrate the speed control aspect of this protocol, wherein two users are connected to the network with links of different bit rates. Line rates used are 2400 BPS for the 'Source Host to Source Node', 'Source Node to Source Host' links, and 7200 BPS for 'Dest Node to Dest Host', 'Dest Host to Dest Node' links in Figure 4.10(a), whilst the line rates are reversed between the user links in Figure 4.10(b). The large gap in time between transmission of successive segments for the faster links occurs since the network does not process more than three segments at a time - one segment entering, one in transit, and one leaving the network.

4.2.9 The ICP Protocol as a Flow Control Measure

The ICP protocol is used by the packet switching network as an end-to-end flow control measure.

It is possible using this technique to restrict the total number of packets that the network may have to process, since only one segment per logical link may be in transit at any time. The maximum number of packets that are in transit through the network is given by the product of the total number of logical links serviced by the network and the number of packets per segment.

Although end-to-end control will not of itself prevent congestion from occurring, it does provide the network with a means for limiting the amount of data to be transferred, i.e., it is not possible for exceedingly large numbers of packets to gain access, thus resulting in almost complete degradation of network performance.

If the specifications of a network are known, e.g., topology, protocols, amount of buffering at nodes, channel capacities, the designer may simulate and determine approximately the number of packets that the network can handle, and whether in turn the network is capable of handling the anticipated traffic conditions.

Although the ICP protocol controls the flow of data between user and network, and provides a means of limiting the amount of data

stored and in transit, it has no control over the admission of packets into the network. This aspect is handled by the switching processor in the node.

Figures 4.6 to 4.9 therefore provide insight only into the user environment-network area for the traffic control procedures used. Simulations performed for fully operational networks in which the end-to-end and local control measures are employed are studied in Chapter 7.

The simulations as shown in Figures 4.6 to 4.9 were conducted for a network utilising both above flow control techniques in which the behaviour of a particular logical link was recorded; however, the traffic in the network was minimal, ensuring little delay for packets in network transit. For highly utilized networks in which packet delays are excessive, the effect on the user-network transmission phase is to increase the intersegment time delay gap.

4.2.10 Summary

This section has detailed the ICP protocol for enabling data transmission between user devices via the network. The format of the protocol, synchronization of network and user processes, and the pipeline effect of the protocol have been discussed. A graphical demonstration of the working of the protocol has been included. Its use, not only as a means for effecting data transmission, but also as a flow control measure, has been pointed out.

4.3 Storage Management

The ICP memory basically serves as a secondary storage medium for the network. e packet switching section requires faster turnover of buffers to achieve adequate throughput and response performance indices. Such use of store-and-forward buffers cannot occur if these buffers are directly utilized by outside devices. A buffer allocated to a 50 bit per second line will be unavailable to other network users for an unacceptable amount of time if used in a store-and-forward mode. The buffer will be tied up for five seconds if used to receive and store a data block of 250 bits. Reassembly buffering may also be blocked whilst packets arrive and are formatted into the original segment.

Thus store-and-forward buffering in the channels has been purposely separated from the interface buffering. The ICP memory is used to temporarily store data until the network is ready to accept and transmit the packets.

Sharing of buffers between users with different line rates will require sophisticated flow control measures if congestion is to be avoided. Rather, provide a temporary storage medium whereby the necessary conversion to a standard line rate can take place, i.e., to 64000 bits per second.

4.3.1 ICP Memory Partitions

The ICP storage has been divided into a number of partitions to prevent performance degradation (see Figure 4.10).

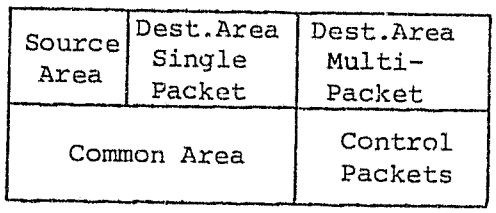


Figure 4.10 ICP Memory Areas

The source class is used to provide the necessary storage for entering data, the destination class is used as a reassembly area. The multipacket area is intended for long file transfers, the single packet area for interactive traffic. Provided sufficient single packet blocks are available, short messages will have a high probability of obtaining a buffer at the first attempt. Finally, a common area for use by all classes is provided. An area is also reserved for control data.

Ideally most efficient usage of storage would occur if all memory were available to whichever logical link section requires it. Such a technique might, however, arise in deadlock, e.g., if all of memory were to be completely occupied by source blocks, the ICP would not be able to receive packets from other ICP's,

Additional storage classes may be devised where necessary: a specific memory area for control packets is a necessity; further

different priority areas may be implemented corresponding to various user classes.

Various buffer allocation techniques have been devised. The most simple is to associate a fixed input and output area with each communication line. A more sophisticated approach is to use a pool of blocks, each block smaller than the size of the majority of messages. A block is assigned at the start of the transmission process. If the block fills with characters, another block from the pool is chained to it. The more complex techniques, although more efficient in buffer usage, require more sophisticated software for implementation.

4.3.2 Packet and Buffer Sizes

The host must divide its messages into segments not exceeding a specified size. The segments are further split into blocks by the ICP (see Figure 4.11).

The size of the packet to be transmitted may be chosen independently of the block size. The packet and block lengths are not necessarily identical. Packet sizes are determined by network response and throughput considerations. The block size is determined by the average incoming message length: smaller block sizes are more efficient since less memory is wasted for short messages, but the overhead increases due to the necessity to employ more block headers and chaining information.

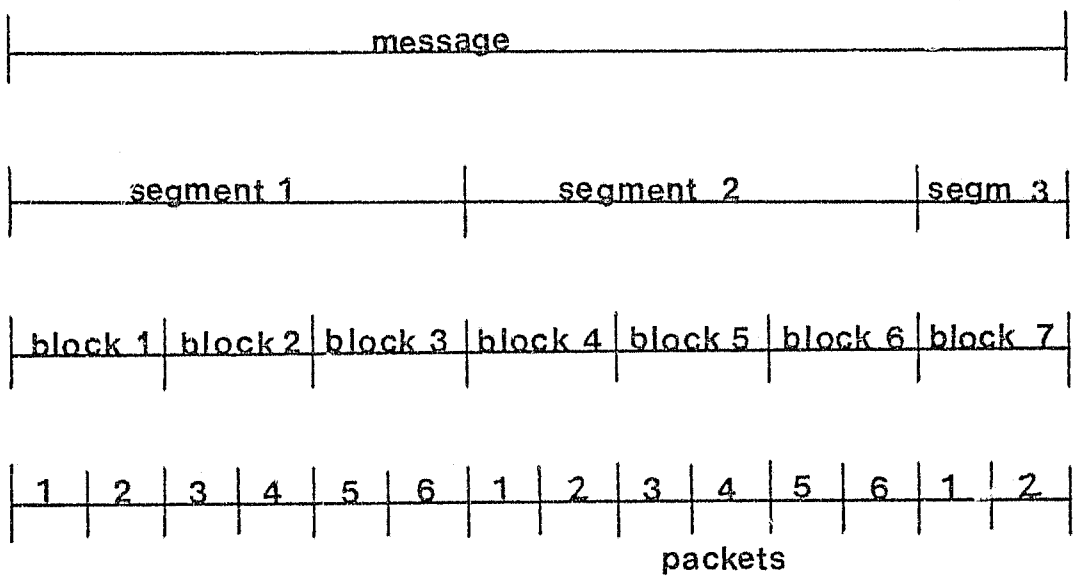


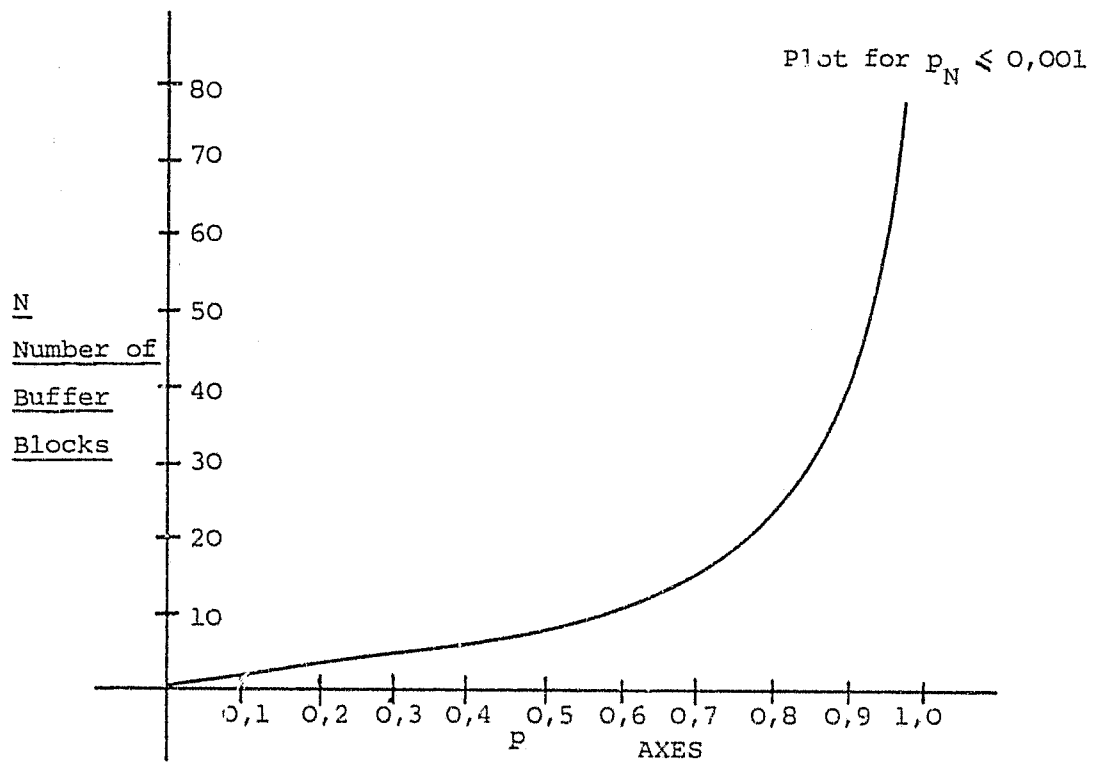
Figure 4.11 Length Concepts

Maximum data field lengths for packets employed in some of the current networks range from 16 to 1024 bytes.[14] In general, selection of parameters such as packet sizes, buffer size, resource allocation, etc., depend very much on the network performance goals and operation of the network mechanisms for transmission of data, e.g., a particular protocol may strongly influence the packet size choice. Unfortunately, the construction of networks with specified performance indices is not fully understood.

One approach, that followed here, is to produce a simulation facility so as to experiment and observe the behaviour of a network subject to parameter and traffic variations. The effect of varying packet and buffer sizes is observed.

$$p_N = \frac{(1-p)p^N}{1-p^{N+1}}$$

(MISCHA) (Ref. 4)



p_N is the probability that a buffer of size N blocks is filled and that messages are turned away. Criterion is $p_N \leq 0{,}001$

Figure 4.12 Buffer Size vs Line Utilization

Figure 4.12 shows the number of buffer blocks required with increasing line utilization. Assumed is an M/M/1 queue, a very crude approximat to the buffering processes taking place in the ICP.

4.3.3 Message Buffering

The ICP storage technique used allocates buffers as follows. The total required number of buffers must be allocated before transmission of the message segment, either between source host and source ICP, or between source ICP and destination ICP. If these are not available, they must be reserved. When the required number of buffers become available and are allocated to the relevant logical link, transmission ensues. If buffers cannot be allocated or reserved, the transmitter is informed. A transmission attempt must then be attempted at a later stage.

4.3.4 Double Buffering Concept

It will be recalled that double buffering is required for the transmission of messages consisting of two or more segments. Since it is assumed that a source interface processor has no way of determining the length of the message to be received, upon reception of a CALLES signal, the full quota of buffers is allocated, i.e., 2 x NSOURC, where NSOURC is the number of buffer blocks required to store a single segment. The packet, block and segment size relationships are given in Table 4.2. An ICP, upon receiving a CALLES signal, checks that a sufficient number of blocks is available, and ther allocates these blocks to the logical link concerned. The double buffering is maintained until an 'End of Message' code is encountered. This will occur for single segment messages and for the final segment of a number of segments transmitted. Unused buffers at the source ICP are thereupon released, the remaining buffer blocks being released as packets are transferred from the source ICP to the relevant channel

NSOURC: NUMBER OF BLOCKS PER SEGMENT
SGMENT: SEGMENT SIZE (MAXIMUM) IN BITS
BUFBLK: BUFFER BLOCK SIZE IN BITS
PSZNO: PACKET SIZE IN BITS
NOPCKS: NUMBER OF PACKETS PER SEGMENT
NOBUF: NUMBER OF BUFFERS FOR DST ICP

RELATIONSHIPS

$$\text{NSOURC} = \left\lceil \frac{\text{SGMENT}}{\text{BUFBLK}} \right\rceil$$

$$\text{NOBUF} = \left\lceil \frac{\text{NOPCKS} * \text{PSZNO}}{\text{BUFBLK}} \right\rceil$$

$$\text{NOPCKS} = \left\lceil \frac{\text{SGMENT}}{\text{PSZNO}} \right\rceil$$

NP: ABOVE NAMES USED CORRESPOND TO THOSE FOR SIMULATION

Table 4.2 Segment, Buffer and Packet Size Relationships

controller in the node (the channel controller must acknowledge reception of the packet).

The destination ICP, however, is in a position to allocate the exact number of buffer blocks required for transmission, whether multi-segment (2 x NSOURC blocks), single segment - multipacket, or single segment - single packet. The source ICP informs the destination ICP of these requirements (since it has at this stage stored the first/only segment) by sending the first packet in the segment to the destination interface processor. If the destination processor is able to allocate the required number of blocks, it informs the source ICP via a control packet that transmission may ensue. It is noted that single packet messages are sent to the destination ICP without first reserving the required number of blocks, since a single packet is a first packet by default. The scheme defined may be used for the transmission of both long and short messages. Figure 4.13 illustrates the storage allocation procedures for source and destination interface processors.

4.3.5 Reserving Buffers

If an ICP is not able to allocate buffer blocks, then the possibility of reserving blocks exists. Reserving buffers requires a certain amount of scheduling. The reservation of buffers is classified according to the type of packet requiring the buffer: source, multipacket and single packet types have been defined. These types may be given different priorities, fixed or dynamic. Both modes of priority allocation have been incorporated in the simulation.

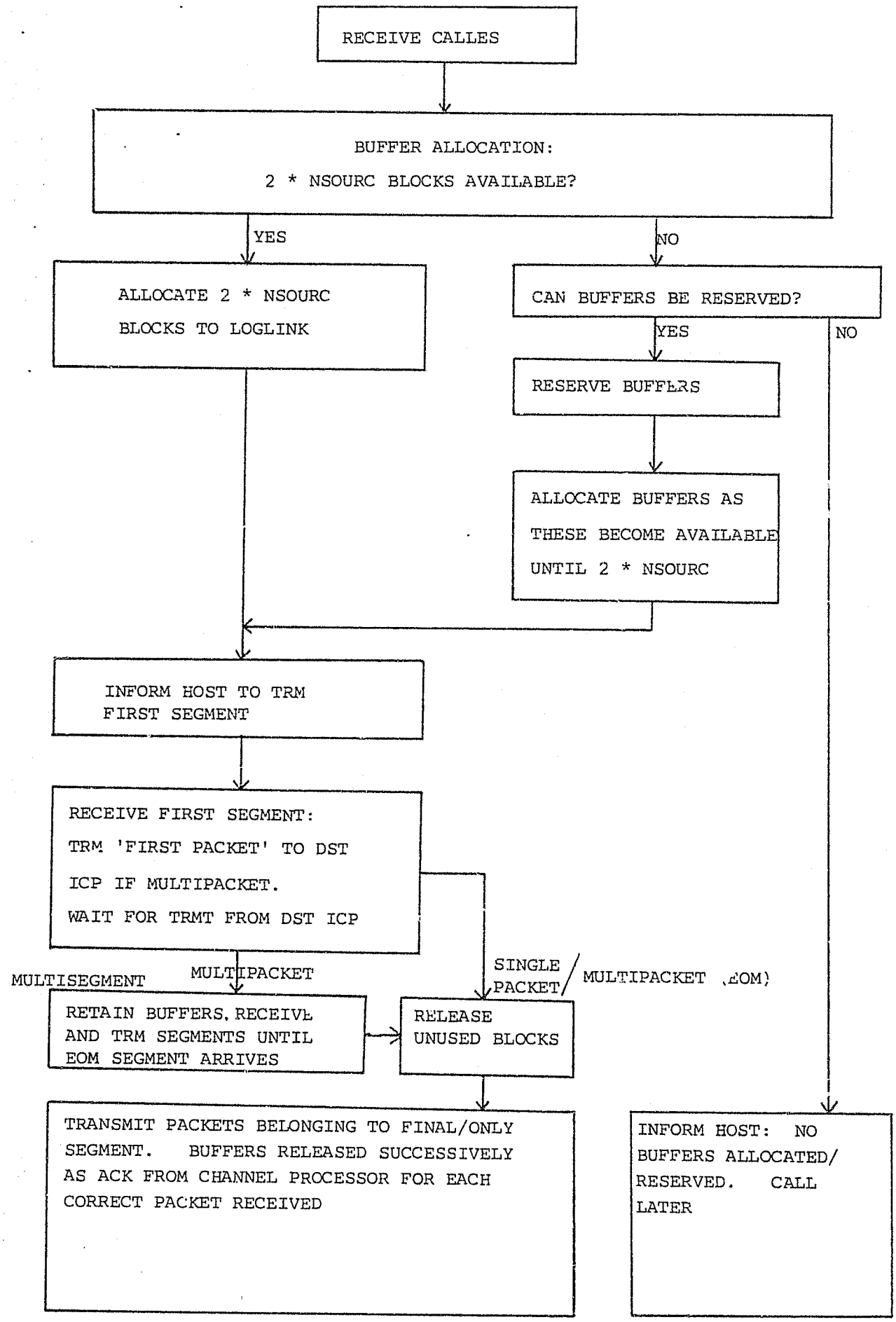


Figure 4.13(a) Source ICP Buffering Procedure

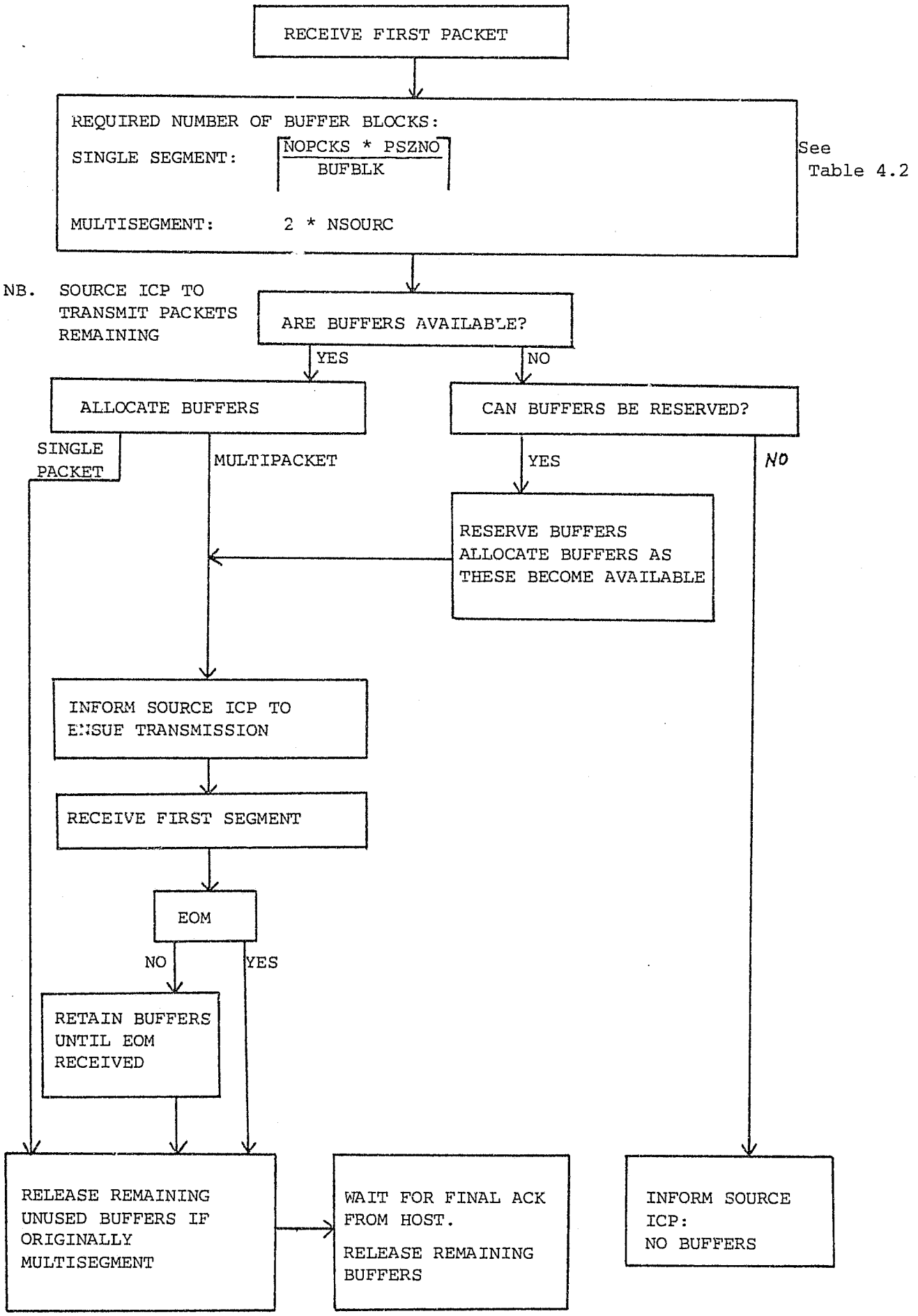


Figure 4.13(b) Destination ICP Buffering Procedure

Dynamic priorities are incremented with respect to time, the size of the increment depending on the type of packet.

All reservation requests are queued by the ICP 'waiting-state' algorithm according to their respective priority ratings. In order to prevent deadlock only the request at the head-of-queue is initially processed. When buffer blocks are freed by other links, their blocks are allocated to the request at the head-of-queue. If the blocks released are of a type different to that of the head-of-queue request, the remainder of the queue is examined in descending order to determine if any other requests may be allocated the released block. (Appendix E)

The identity of the head-of-queue request is retained by the algorithm and may not be pre-empted by other requests of higher priority (as might occur with dynamic priority assignment, resulting in possible deadlock).

As soon as the required number of blocks has been allocated, the request is deleted and the caller is informed that transmission may proceed. The next head-of-queue is subsequently processed.

In the case of blocks not being able to be allocated or reserved (the number of segments is limited), the relevant transmitting unit is informed that it must attempt to send its data at a later stage, after some random interval.

4.3.6 Summary

The ICP protocol has been purposely kept simple to eliminate the need for complex flow control and storage algorithms. The flow control procedure is conservative in that buffers must initially be reserved for all data messages of size larger than one packet, before data transmission may begin. Buffers are not, however, reserved for the packets in transit through the packet switching section of the network, i.e., in the channel controllers. Although the buffering at the source ICP's is handled somewhat inefficiently, this method ensures that data cannot be lost, since the situation of finding no buffers for storage midway through a transmission cannot occur.

4.4 Review

This chapter has introduced one of the main components of the communication node, the interface processor. The role and functions of the interface processor have been discussed. The logical link concept, the basis for further work, has been described. Attention was focussed on the details of a user-network protocol that allowed for data transmission and flow control between user devices and the network nodes. Simulation results have been included to demonstrate graphically the workings of the user-network protocol in coordinating data transmission from one end of the network to the other. Finally, the storage structure of the ICP was outlined.

CHAPTER 5

CHANNEL CONTROLLER

5.1 Introduction

This chapter considers the network communications processor or channel controller. The node-to-node protocol and the buffer management technique incorporated in the controller, are specifically examined. In addition simulations have been conducted to evaluate the performance of the controller.

The concept of a 'channel' consisting of two controllers, one attached to either end of a PCM link, is a basic building block of the network. Arbitrary network topologies may be constructed by linking nodes with channels. An example is shown in Figure 5.1. The capacity of the PCM link is 64000 bits per second, with error rates in the region of one in 10^7 to 10^8 bits.[3] The controllers are envisaged to be microcomputer-based.

The function of the channel controller is purely store-and-forward. The source controller accepts packets, provided buffers are available, and transmits these packets on a first-come-first-served basis to the destination controller. It is the responsibility of the channel to ensure that packets are transmitted correctly from one node to another. The controllers operate in full duplex mode. The channel concept is modular and may therefore be applied where network expansion is necessary.

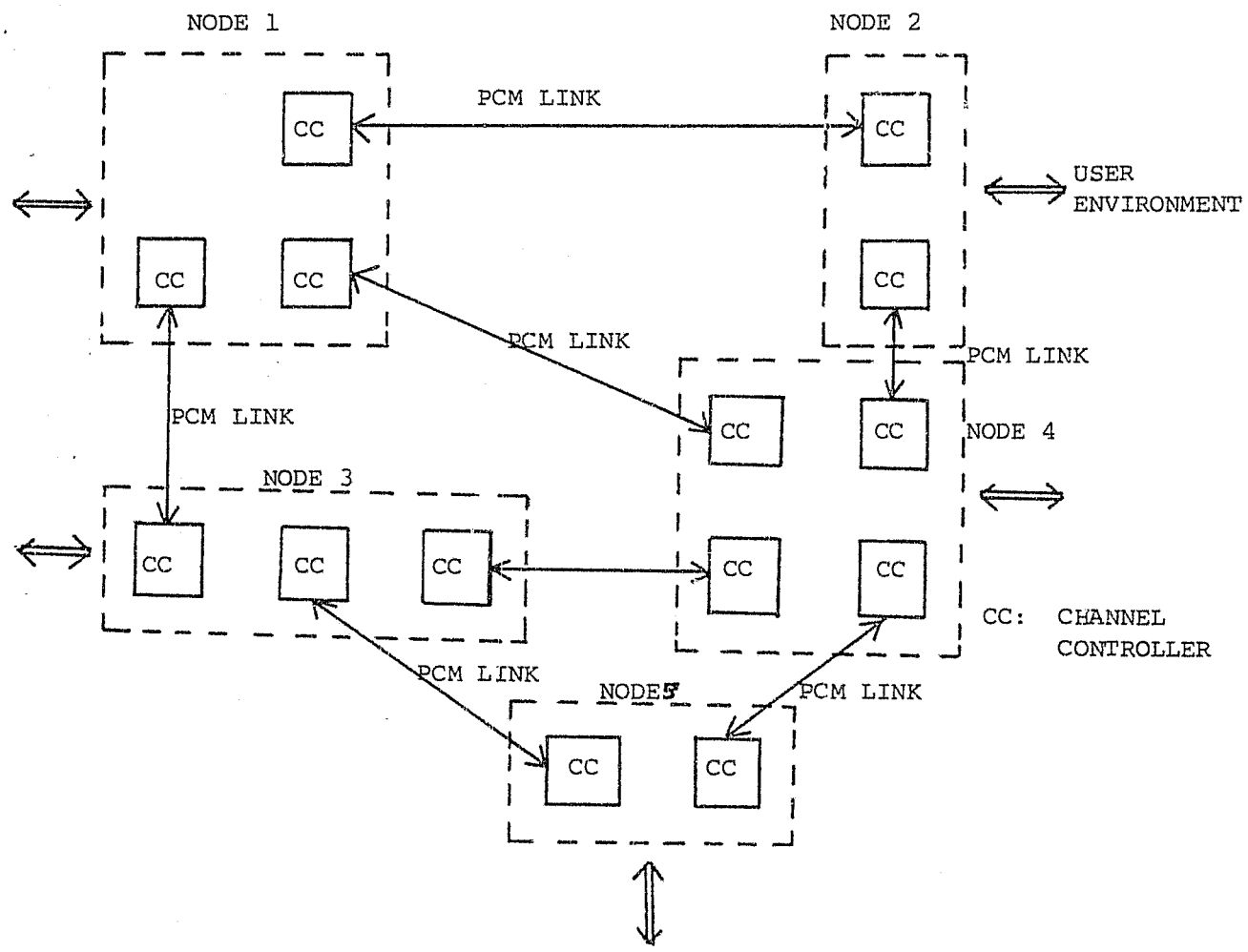


Figure 5.1 Network Example

The following sections consider the protocol enabling the exchange of packets between channel controllers to take place, as well as the technique for allocating buffers to packets enroute through the network.

5.2 Node-to-Node Protocol

A link control procedure must satisfy certain requirements if it is to serve as a reliable means of communication control. Synchronization must be established and maintained for proper delivery of

messages. Messages must be formatted and checked, and an acknowledgement procedure is necessary to indicate whether a packet has been correctly transmitted or not. Overhead in terms of control characters must be kept low. The line overhead depends critically on the characteristics of the messages. A large number of small messages involves more overhead than a small number of large messages to transfer the same number of data bits.

The protocol incorporated into the channel controller is independent of the code structure transmitted. It is bit-oriented and does not use control characters. It applies to serial-by-bit synchronous transmission between buffered stations. The protocol operates in full duplex mode and is employed in the point-to-point configuration. The above characteristics are shared with most other bit-oriented protocols. The protocol incorporated differs in that only incorrect packets are retransmitted. The Synchronous Data Link Control, for example, requires the retransmission of all transmitted packets after a faulty packet has been received at the destination controller.[27]

5.2.1 Link Throughput

The effective channel throughput depends not only on the link bit rate and packet overhead, but also on the ability to acknowledge packets efficiently.

When a packet is transmitted from a source to a destination station a copy of the packet is retained at the source until such time as an acknowledgement is received from the destination, indicating to the

source station that the packet has been correctly received at the destination. A buffer block is thus occupied at the source for the duration of the packet transmission and the acknowledgement delay time. The amount of buffering required is approximately given by the circuit rate times the expected acknowledgement delay.[18]

It is therefore imperative to keep the delay as short as possible, if memory requirements are to be kept within reasonable limits, yet allowing for a high throughput rate to be attained.

5.2.2 Outline of the Protocol

To provide a description of the protocol two channel controllers, A and B, connected via a link are assumed. Controller A is to transmit a packet to controller B.

A packet is sent from A to B, a copy being retained at A. Upon correct reception of the packet at B, an acknowledgement is transmitted from B to A, either via a data packet in the process of being sent from B to A (piggybacked) or by a control packet. When A receives the acknowledgement it releases the copy and the buffer is available for another packet. If the packet received at B is incorrect, then it is discarded. In this event, no acknowledgement is sent to A. At A, after the elapse of a time-out period, the packet is retransmitted.

5.2.3 Logical Path Concept

Although the protocol outlined above is correct, it has one problem. Once a packet has been transmitted by a source controller, then that

packet must be acknowledged before further packets can be transmitted. The technique is thus unsatisfactory from the throughput point of view. The problem can be overcome by making use of a number of logical paths (see Figure 5.2).

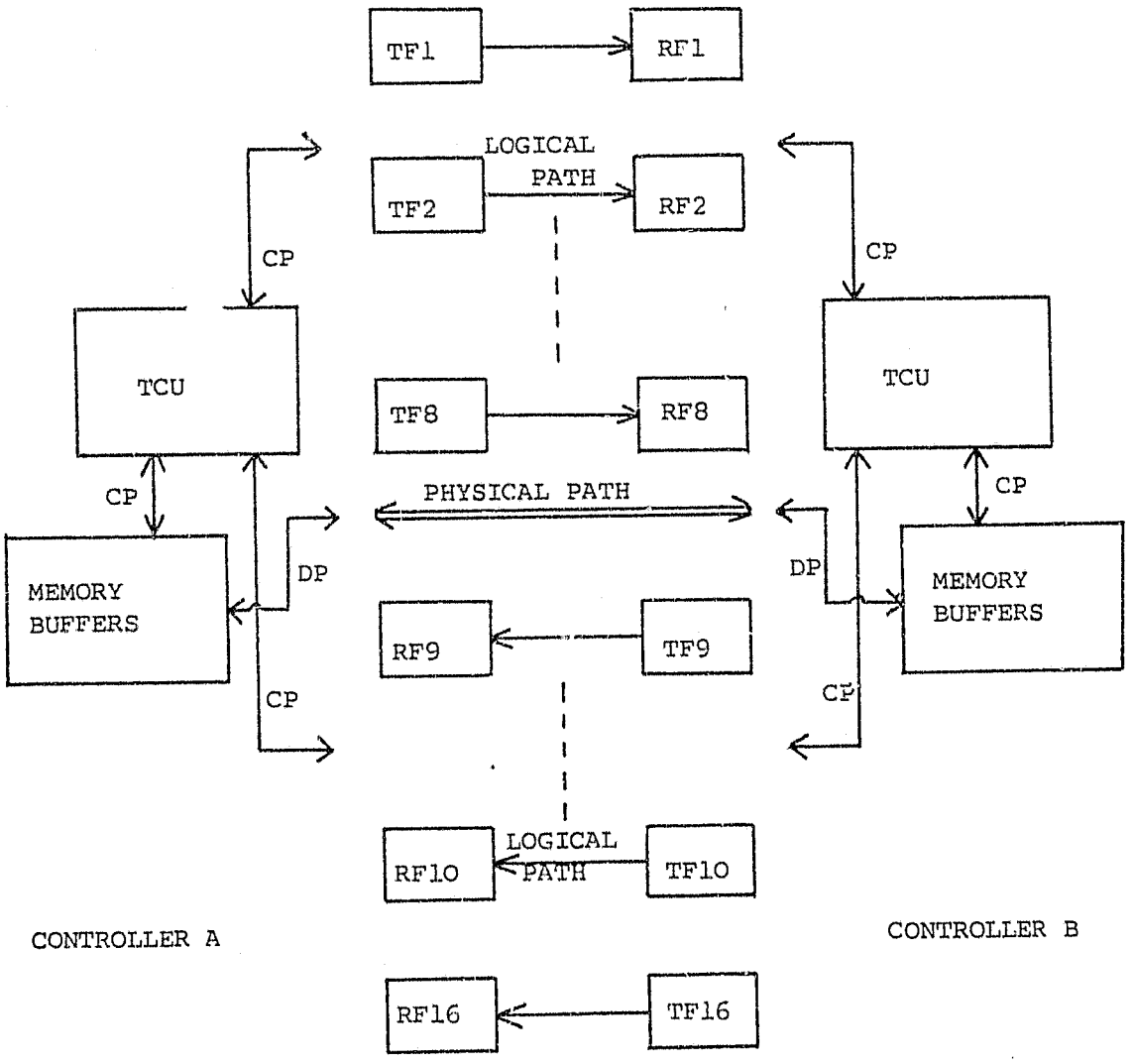


Figure 5.2 Schematic of Fields

A number of logical paths may be defined that share the physical PCM link, according to the protocol outlined above. Each logical path operates independently of the other paths, sending a packet and then waiting for acknowledgement. The maximum number of packets that remain unacknowledged at any one time, is equal to the number of paths used. Simulation tests have indicated that eight logical paths per channel are sufficient.

For each channel, status information concerning each of the logical paths is retained at both the source and destination controllers. The block of data that describes the state of the logical path is termed a 'field'. The fields at the source and destination controllers are referred to as the transmitting and receiver fields respectively. All such fields are maintained and updated by control unit programmes, in each of the channel controllers.

Data packets stored in the controllers' buffers are sent to the destination controller provided the physical link is free and a transmitting field is unused. The control unit supervises the transmission making sure that each field obtains adequate usage of the physical path resource. The transmitter and receiver fields at the source and destination controllers respectively, must maintain synchronism. The transmitter field maintains status data concerning the packet in transmission. The receiver field has the function of informing the source controller whether a packet was accepted or not, and whether the packet has been rejected due to bit errors or due to insufficient buffering (at the destination controller).

The field structures are shown in Figure 5.3.

FIELD NUMBER	ACTIVE FLAG	PACKET NUMBER	PARTITION NUMBER	SYNCH FLAGS	TIME-OUT CONTROL	PARTITION CONTROL

TRANSMITTER FIELD

FIELD NUMBER	SYNCH FLAGS	COMMAND FLAG

RECEIVER FIELD

Figure 5.3 Field Structure

Each logical path is identified by a field number. The activity flag is used to indicate whether a field is in use or not. The packet number is used to identify the packet currently in transmission, or awaiting transmission along the logical path specified by the field number. The partition number and partition control, as well as the command flag, are used to aid the buffer management process (see Section 5.3). The time-out section controls the duration of the time-out interval, after which a packet is to be retransmitted. The synchronization flags control the flow of packets along a logical path.

5.2.4 Synchronization Flags

The synchronization flags are used to ensure that duplication of packets occurs should error conditions arise. The mechanism is in fact time-independent. If a time-out occurs due to excessive acknowledgement delay (e.g., an acknowledgement packet is incorrect), whilst the originally transferred packet has been accepted by the

destination controller, the retransmitted packet will be identified as a duplicate and not accepted.

The synchronisation mechanism is shown in Figure 5.4. Four combinations of bits are used, the right-most bit, a zero, indicating packet transmission, the same bit, a one, indicating acknowledgement of transmission. The transmitter and receiver fields may be initialized to either '00' and '11' respectively, or to '10' and '01' respectively. To ensure continued synchronization, control packets are periodically transmitted between the two controllers to ensure that the above condition is satisfied. This condition is only guaranteed for non-active fields.

Assume that a packet requires to be transferred from controller A to controller B and that the transmitting field at A and the receiving field at B are initialized to '00' and '11' respectively. Correct packet communication requires that B receives the packet frame with flags set to '00'. An acknowledgement with flags set to '01' will be returned to A provided the packet is correct, otherwise the previous values of '11' will be returned. A awaits the return of the acknowledgement flags set to '01', and once received will release the buffer.

If a time-out occurs before the required flags are received, the packet will be retransmitted. For the following transmission A will set its flags to '10', and B will return a value of '11'. The process

starts anew with values of '00' and '01' for the packet transmission thereafter. If an incorrect acknowledgement occurs whilst the packet has been accepted, a retransmission will take place once the time-out perio' has elapsed. The retransmitted packet will be identified as a duplicate, since the destination controller recognizes the transmitter flag settings to be the same as those for the previously received packet.

Flag settings for both receiver and transmitter fields must alternate with successive packet transmissions. It is noted that each logical path operates independently of the other paths in this respect.

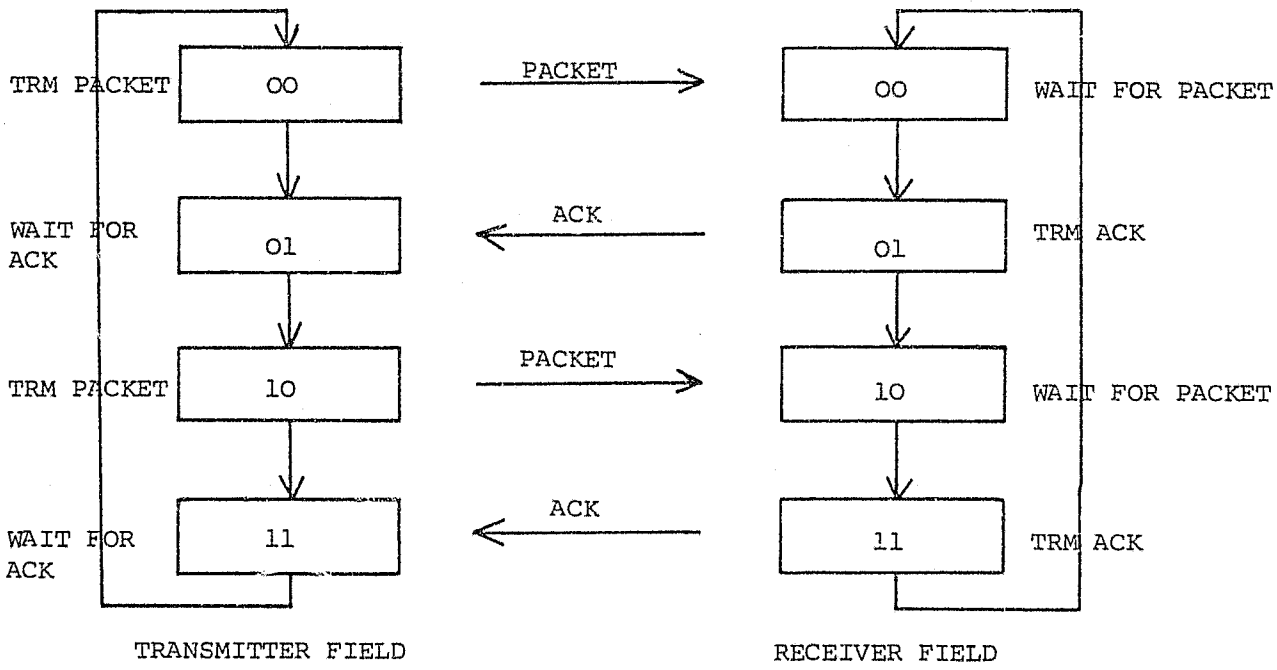


Figure 5.4 Synchronization Flags

5.2.5 Packet Frame Format

The packet format is shown in Figure 5.5. A packet, together with its flags and control information, is referred to as a frame. The header is used to contain network addressing and control information. It is used for routing by intermediate nodes, and for flow control and packet reassembly by source and destination nodes. The text is transparent to the network code used. The flags, serving as frame delimiters, establish and maintain synchronization. The flag format is that of SDLC where bit-stuffing is used to ensure that the flag is the only such data field appearing in a frame.

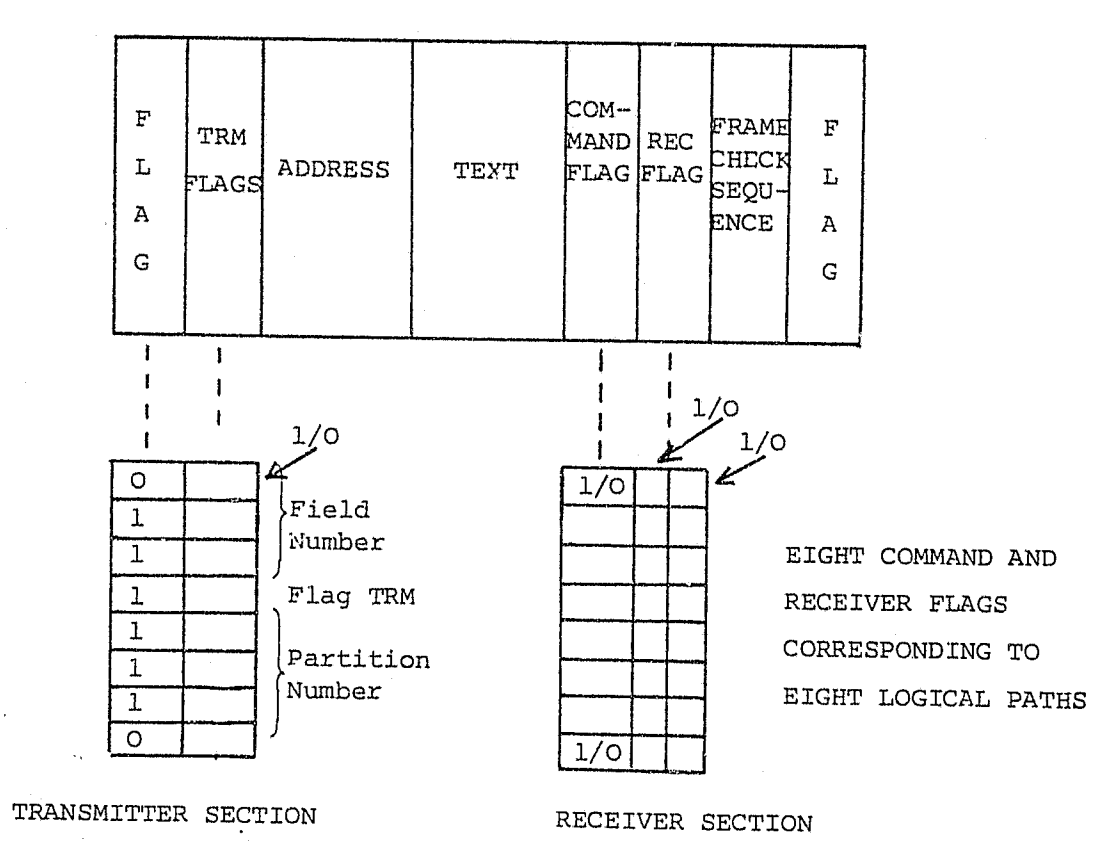


Figure 5.5 Frame Format

The transmitter, receiver, and command sections are the important features of this protocol. The field number identifies to which logical path the packet belongs. Eight paths are possible with the frame format depicted in Figure 5.5. The single bit transmitting flag has the synchronization function as described in the previous sub-section. In addition, sixteen possible partition numbers are included for buffer management purposes (see Section 5.3).

A channel controller transmitting to its destination controller will always include the status of its receiver fields, with each packet sent. Such a feature has two advantages. It allows for simultaneous acknowledgement of any or all fields, and it ensures that only incorrect packets are retransmitted. This technique thus allows for efficient acknowledgement, piggy-backed on returning data packets. In the event of no returning data packets, a control packet containing the receiver flags is sent.

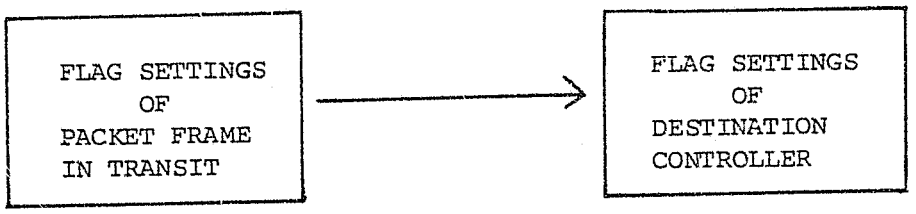
The command bit is used by the receiver to indicate whether a packet has been rejected due to insufficient buffering or not (see next section).

A standard generator polynomial may be used for the frame to check for errors, e.g., CCITT recommendation.

V.41 for 16-bit fields: $g(x) = x^{16} + x^{12} + x^{5} + 1$

5.2.6 Valid Flag Settings

The valid combinations of synchronization and command flag settings are shown in Figure 5.6. Any other combinations are incorrect and a re-initialization process is required.



TRANSMITTER FIELD		RECEIVER FIELD	
		11	RECEIVE
	00		
	10	01	PACKET
PART A			
	00	01	DUPLICATE
	10	11	PACKET

RECEIVER FIELD		TRANSMITTER FIELD	
COMMAND	SYNCHRONIZATION		
0	01	00	RELEASE BUFFER
0	11	00	NO-OPERATION
1	11	00	NO-OPERATION
1	01	00	RETRANSMISSION
PART B			
0	11	10	RELEASE BUFFER
0	01	10	NO-OPERATION
1	01	10	NO-OPERATION
1	11	10	RETRANSMISSION

Figure 5.6 Flag Settings

Part A is used to enable the destination station to determine whether it already has a copy of the packet or not. Part B is used for acknowledgement of packets.

A 'no-operation' indicates that:

i) a packet has not yet arrived at the destination controller,

ii) the packet has arrived but been rejected,

iii) the packet has arrived, been accepted, but its acknowledgement has not been received.

A 'retransmission' means that buffering was not available for the packet and that the source controller must retransmit the packet after a suitable delay period (in the simulation approximately 40 milliseconds).

A 'release buffer' indicates that the packet was accepted at the destination, so that its buffer may be made available to another packet.

5.2.7 Summary

A possible protocol has been outlined, only in-so-far as it affects network traffic flow. The principal feature of this protocol is its use of bit positions to synchronize frame transmissions. Such provides for the retransmission of only incorrect packets and enables a number of logical paths to be superimposed on a physical channel. Delays on one logical path do not affect link throughput significantly since other logical paths are not blocked in the process.

5.3 Storage Allocation

The amount of buffering in the channel controllers is limited; as such, network operation is very much dependent on the flow control measures employed. (Bulk storage mechanisms are excluded mainly for reasons of reliability and the added complexity to the nodal architecture.) As no techniques for deleting packets at very congested network regions have been considered, due to the complexity of the error recovery procedures required to continue network operation after deletion, greater emphasis must be placed on techniques for preventing degradation from occurring. The role of the storage allocation technique in this respect is now examined.

5.3.1 Store-and-Forward Buffer Blocking

Degradation in throughput occurs if buffers are not available to packets in transit through the network. If all of memory were made available to any one of a number of different groups of packets, e.g., packets to a specific neighbouring node, it is possible that a situation could arise whereby packets from one group use most of available memory. Such can arise when the ratio of the service rate to that of the arrival rate is low for one group, this group thus effectively blocking the transmission of other groups by occupying most of the buffers. Deadlock may even arise if two or more groups require each others resources for transit, whilst retaining their own.

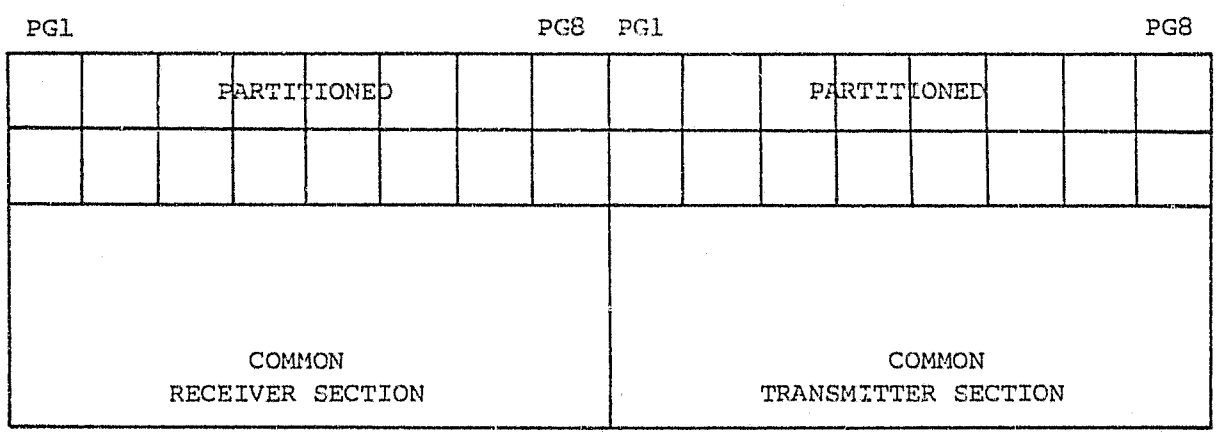
Due to the limited processing power of the computing elements involved, no mechanism exists for temporarily releasing a resource, i.e., the buffer, and later regaining it. Once a resource is accessed,

it will only be released when the user (a packet in this case) has finished with it.

Thus a packet in transfer from node A to node B will occupy a buffer in node A until such time as it is able to seize a buffer at node B, the release at A subsequently occurring upon reception of an acknowledgement signal.

5.3.2 Memory Partitions

To overcome the above problem the memory is partitioned intc a number of segments. A schematic is shown in Figure 5.7.



PGi: PARTITION GROUP i

Figure 5.7 Memory Division

The memory is split into a receiver and a transmitter section. Each section is in turn divided into a partitioned and common area. A partitioned memory area is allocated to a specific group of packets,

whilst the common memory is available to all groups.

From a resource sharing point of view, better sharing of resources occurs if all storage is available to all groups, rather than to provide each group with its own storage modules,
A compromise has therefore been made, each group having sole access to a number of memory blocks and the rest of memory being made accessible to any group requiring it. This means that heavy demands by one group may be satisfied, but the dominant group will not be able to block others from acquiring the necessary resource for network transit.

It is in fact shown in Kleinrock[21] that the response time for a channel comprising two resources (receiver and transmitter memory) will not be significantly larger than if the two resources were merged into a single resource, assuming constant channel capacity. The increase is in the order of two milliseconds.

In the simulation program eight partitions or groups are provided for. Each partition consists of two buffer blocks. Any particular group of packets will thus not be blocked since memory partitions in the transmitter section at a source node will have corresponding partitions in the receiver section of a destination.

5.3.3 Memory Partitioning According to Topology

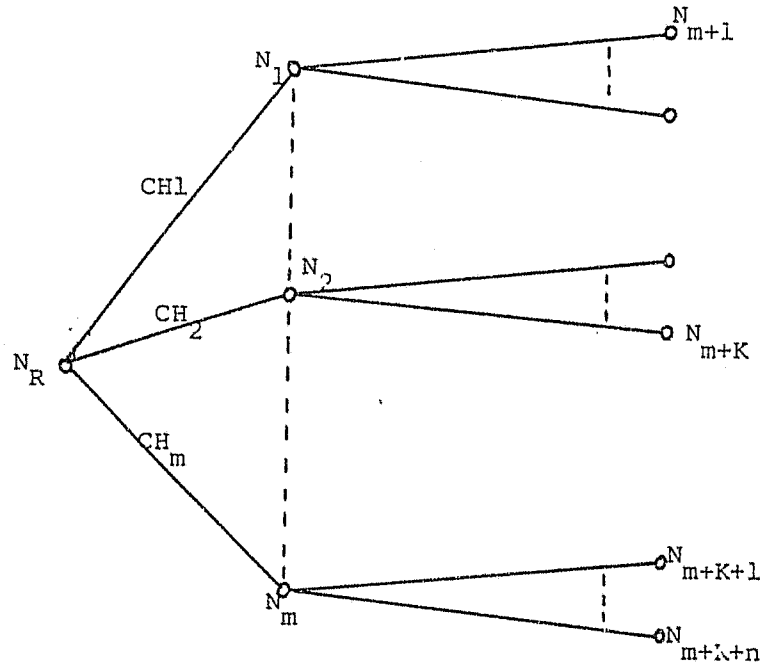
The grouping of packets occurs with reference to the topology of the network. As packets move through the network their grouping will change from node to node. In this regard control packets, e.g.,

acknowledgements, 'Ready for Next Segment' messages, are not distinguished from data packets. These packets are usually best differentiated by providing better service times for control packets based on priority ratings.

The process of partitioning is illustrated in Figures 5.8 and 5.9. A reference node in any network may be considered in terms of the tree structure shown in Figure 5.8. The reference node is connected to a number of neighbours, and these in turn are connected to their own particular neighbours (if any). For convenience, the first is referred to as a 'direct connection', and the second as an 'indirect connection'. The partitioning takes place on the basis that group packet flow between any nodes must never be impeded due to lack of buffer resources caused by other groups. Thus all directly and indirectly connected nodes must at all times be able to transmit data to the reference node. This implies that directly connected nodes may never form a barrier between an indirect and reference node.

A number of buffers will be specifically made available for data transferred from any one node in a network to another. Generally, use will be made of the buffer pool (common) since the number of partitioned buffers is small - two at the source and two at the destination, per partition.

Figure 5.9 gives an illustration of the partitioning for channel m. The partitioning process is done for all channels of a reference node as follows:



N_R : Reference Node

N_1 to N_{m+K+n} : Nodes directly and indirectly connected to Reference Node

CH1 ... CH_m : m channels connected to Reference Node

Figure 5.8 Memory Partitioning: Topology

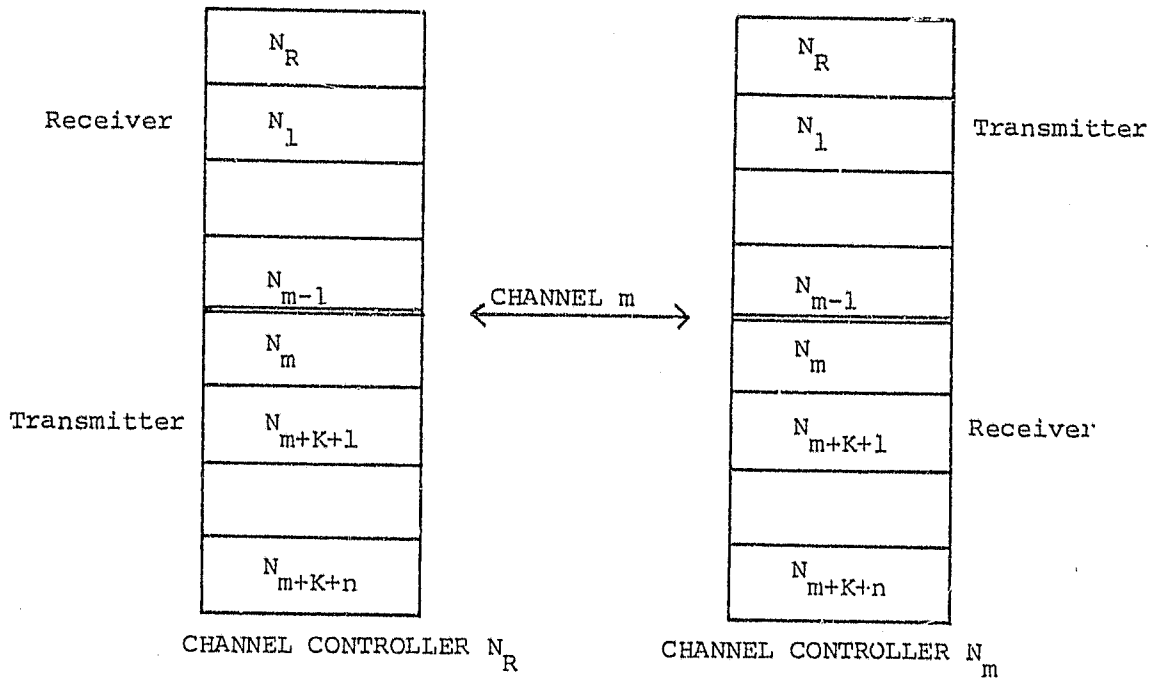


Figure 5.9 Memory Partitioning: Channel m

concerning the receiver section, the reference node itself, plus all directly connected nodes, excluding the node linked to the channel under consideration, are all allocated a partition;

concerning the transmitter section, all indirectly connected nodes via the referenced channel are allocated a partition.

It is noted that by obeying the above rule at the m'th node for the same channel (m), corresponding partitions to those for node R are produced. The memory sections of two channel controllers connected to a common channel are aligned.

5.3.4 The Packet Frame Format and Memory Partitioning

The command bit mentioned in the previous section is used as a flow control measure to ensure that a source station does not transmit packets associated with occupied destination partitions and full common storage, needlessly. The relationship between fields, logical paths, command flags, and partitions may now become clear. In order that the channel controller may transmit data at a high rate without recourse to extensive software based decisions, a technique has been formulated whereby a physical line may be shared by several logical paths. Due to the delay encountered by geographically separated communication processes, it is necessary to share a physical path between a number of such processes to achieve high line utilization. It is noted that the processes referred to above are not those of network users who are allocated channels on demand for their messages, but those of channel controllers who must share these channels amongst users efficiently.

The status of each logical path is retained via a field (synchronization flags, packet identity, partitioning data, etc.), each path responsible for its own transmission, and granted access to the physical line by the transmission control unit. The partition of a packet allocated to a field is determined by a Markovian routing matrix based on present and destination nodal addresses. If a packet is rejected due to insufficient buffering at the destination, the command bit will be set. The transmitting controller will subsequently become aware of this when it receives a packet from its destination; by mapping the command bit onto the partition information stored in the source field, the controller can determine which partitions are blocked at the destination, and transmit subsequent packets accordingly. The blocked packet is removed from the field to make way for differently partitioned packets. After a time-out period set by the partition timer, a retransmission of the packet is attempted.

The switching processor responsible for coordinating the activities of the nodal processors, will ensure that no packets of a blocked partition will be sent to an NCP with its partition command flag set.

5.4 Performance of the Link

A channel has been simulated to obtain figures for the average response time. The throughput of the channel is not too important in this regard; an approximate measure of throughput is given by the product of the average arrival rate and the blocking probability due to no buffering being available for an arrival. The throughput is

considered in more detail when dealing with the entire network.

In Figure 5.10 is shown a schematic of the channel tested. Data packets are generated on both sides of the channel at the rate of $\lambda 1$ packets per second. If no buffering is available the packet is simply rejected. Once buffered, the packet is transmitted to the destination controller according to the link control procedure as described in Sections 5.2 and 5.3.

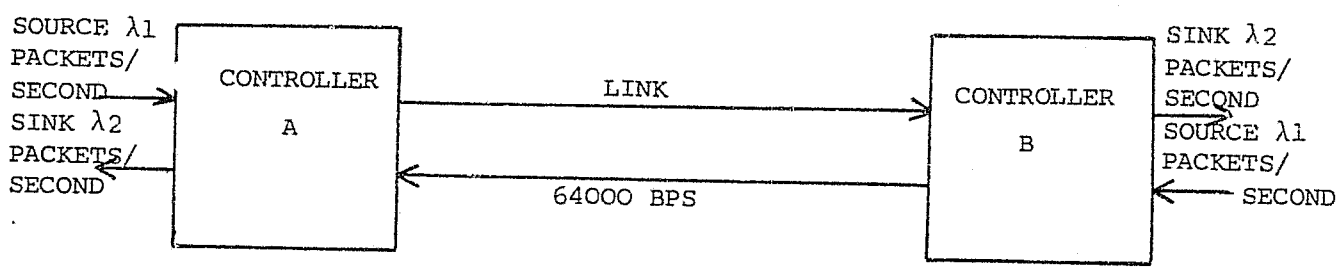


Figure 5.10 Channel Simulation

Associated with each channel controller is a mechanism for extracting packets from the channel at the rate of $\lambda 2$ packets per second. The order in which packets are withdrawn from the link is arbitrary. The response time is defined to be the time from when a packet enters the source controller until the time it arrives at the destination controller. Thus only the amount of buffering available at the destination is of importance, and not the order in which packets are sunk. At any rate, the order of packet extraction from a channel that has been incorporated into a network, is a function of the traffic patterns in a node. These patterns are rather complex and consequently

it is difficult to give a distribution for the average waiting time at a destination controller. This point is examined in more detail in Chapter 6, where a nodal configuration is considered.

The purpose of the set of simulation results is to obtain some idea of the behaviour of a channel when subjected to a number of controlled traffic inputs. There exists a large number of combinations of different parameters for which to test the channel but only two have been considered. The parameters studied are the response of the channel and the amount of buffering required subject to different channel utilization factors. It is felt that these two parameters are sufficient to gain an understanding, from the flow control point of view, of channel performance when the channel is placed into a network.

5.4.1 Parameters for the Channel Simulation

A description of parameter settings for the simulation follows.

The number of fields per controller has been established at eight. As the simulation results will demonstrate, the eight fields are sufficient for a 64000 BPS line rate, utilization factor 0,7, provided enough buffering is available at the source and destination controllers; no blocking of packets at the input to the channel was found to occur in this case.

A single set of partitions has been used, comprising two buffers at the transmitter and two at the receiver side of a channel controller. Two buffer blocks are provided so that, in the event of blocking due to

other packet groups occurring, a double buffering feature is available. Whilst one buffer's contents are sent from data bus to controller, or V V, the second buffer's contents are transmitted across the channel link, the buffer roles reversed in the next phase.

Line error distributions are assumed to be geometric, with the mean set at 10^5 bits per error.

A buffer block has been taken to be equal to the maximum packet size, the size in this case limited to 1000 bits. Although the storage size is varied, transmitter and receiver storages are kept at equal sizes. In the graphs to be shown, the unit referred to is a 'buffer', i.e., a block of 1000 bits. These units refer only to a transmitter or receiver section; the total amount of buffering required per controller is therefore twice the number of buffering units shown, plus four buffers for partitions.

The packet length has been assumed to follow a geometric distribution, whilst the arrival rate is a Poisson distribution. Packet sizes may be adjusted by the mean value parameter, but in all cases truncation occurs at 1000 bits for packets of length greater than the buffer block sizes. The rate of packet extraction has also been assumed to follow a Poisson distribution.

The retransmission interval for incorrect packets is 40 milliseconds and the partition timer period 30 milliseconds.

The expected delay for a channel utilized at 0,7 with an average packet length of 300 bits is approximately 10 milliseconds. This figure is obtained from the Pollaczek-Kintchine formula for M/G/1 queues. (Measurements performed on the Arpa network give packet sizes of mean length 218 bits[21].) Four fields may therefore be transmitted before a retransmission is due, by which time an acknowledgement should have been received. For a heavily utilized destination controller, servicing packets at the average rate of 150 per second, an average of four buffers are freed every 30 milliseconds.
A retransmitted packet will therefore certainly obtain a buffer at the destination in this case, since only one partition has been assumed. A blocked partition thus implies a complete cessation of transmission.

Required is to obtain a time-out period such that a retransmitted packet has a high probability of obtaining a destination buffer.
A detailed investigation of this facet is felt to be outside the scope of network flow control, however.

Preliminary calculations indicate that the order of delay, as determined by the above figures, is tens of milliseconds. Since a channel controller has essentially only link control functions to perform, the delay due to message processing has been neglected. It is thus assumed that processing overhead will have little effect on the figures presented for response times.

5.4.2 Channel Performance

Figures 5.11 to 5.14 show the relationship between response and buffer amount for three channel utilization factors. These include the mean value and variance of the response. The simulation points are plotted for mean packet lengths of 300 bits and assume equal input and output rates to the channel via external sources and sinks. The curves connecting the simulation points are intended to show the general response trend.

5.4.3 Channel Response

The response at low channel utilization (p = 0,3) is depicted in Figure 5.11. It is of interest to note that the response peaks to a value of 35 milliseconds for a buffer value of eight, before decreasing to 9 milliseconds as the storage is increased. This characteristic occurs due to the large number of retransmissions required, since the buffering at the destination is insufficient. Retransmissions affect the response to a large extent (the time-out period is 40 milliseconds). As more storage is provided the queueing delay for packets becomes the predominant factor. The threshold point for p = 0,3 is sixteen buffers.

At a larger utilization factor of 0,7 (Figure 5.12) the same trend is exhibited as above, but the final state response is much larger, i.e., approximately 23 milliseconds.

When the amount of traffic on the link is increased, such that p = 0,82, the response time increases as more storage is added to the controllers. Figure 5.13 shows that no final value of delay has been reached at a 32 buffer value. This large response is again primarily a queueing delay.

A comparative view of the response trends of the previous three graphs is given in Figure 5.14. Only the mean response values are shown. It is clear that the number of buffers chosen should not be less than sixteen if one is to avoid operation of the link in a region where delays are large, even under low traffic conditions.

5.4.4 Percentage of Packets Rejected

Figure 5.18 has been included to demonstrate the effect of storage size on the percentage of offered traffic rejected. For each simulation run the number of packets offered to the channel, and the number subsequently rejected due to insufficient storage (at the source controller) were counted. The number of rejected packets is expressed as a percentage of the offered traffic.

These results have been included only to emphasize that sixteen buffers are adequate under the above stated conditions. Accordingly the rest of the simulation results are based on the value of sixteen buffers per transmitter or receiver section of a controller.

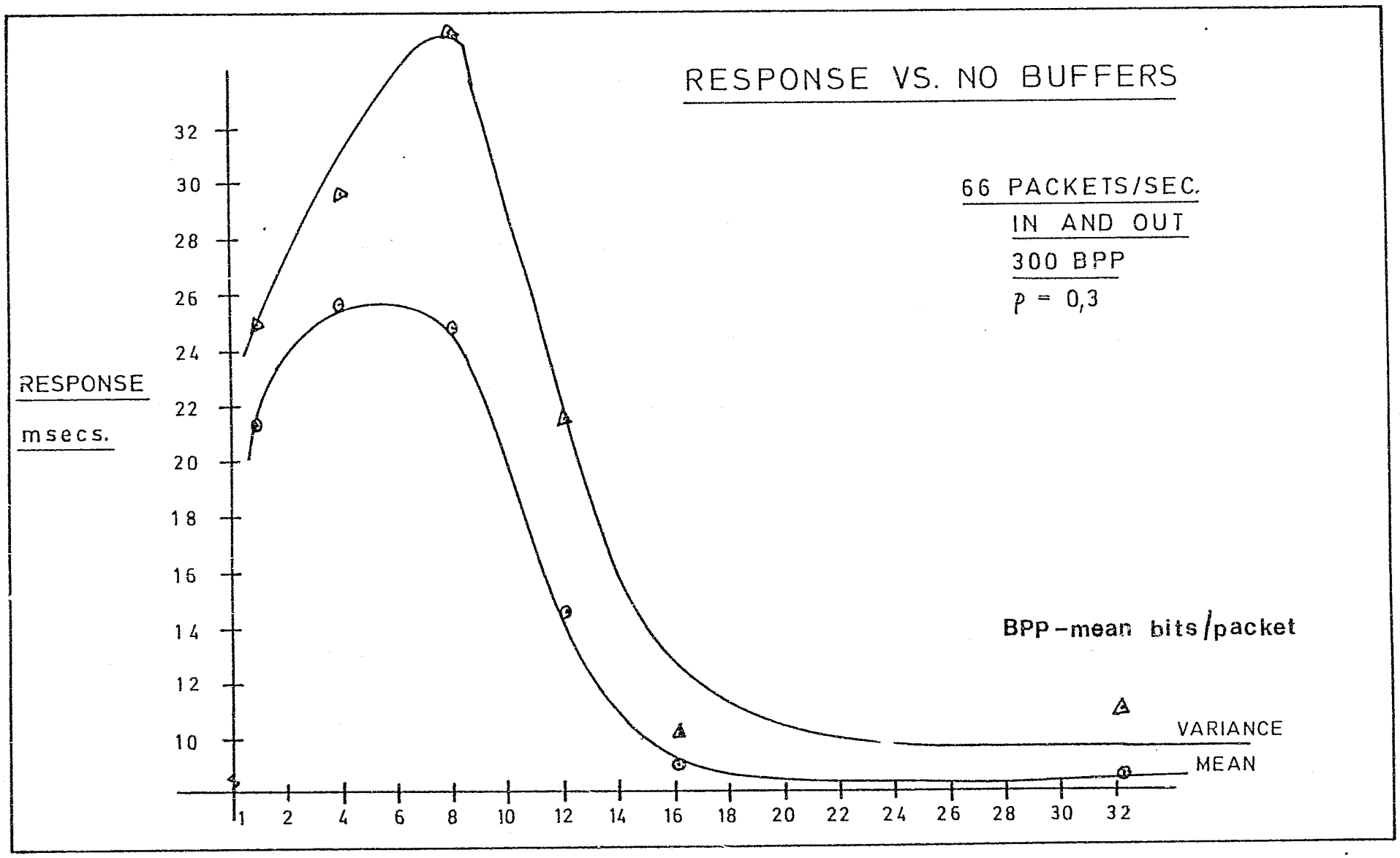


FIG. 5.11. NO. BUFFERS (+ 2 PARTITIONS)

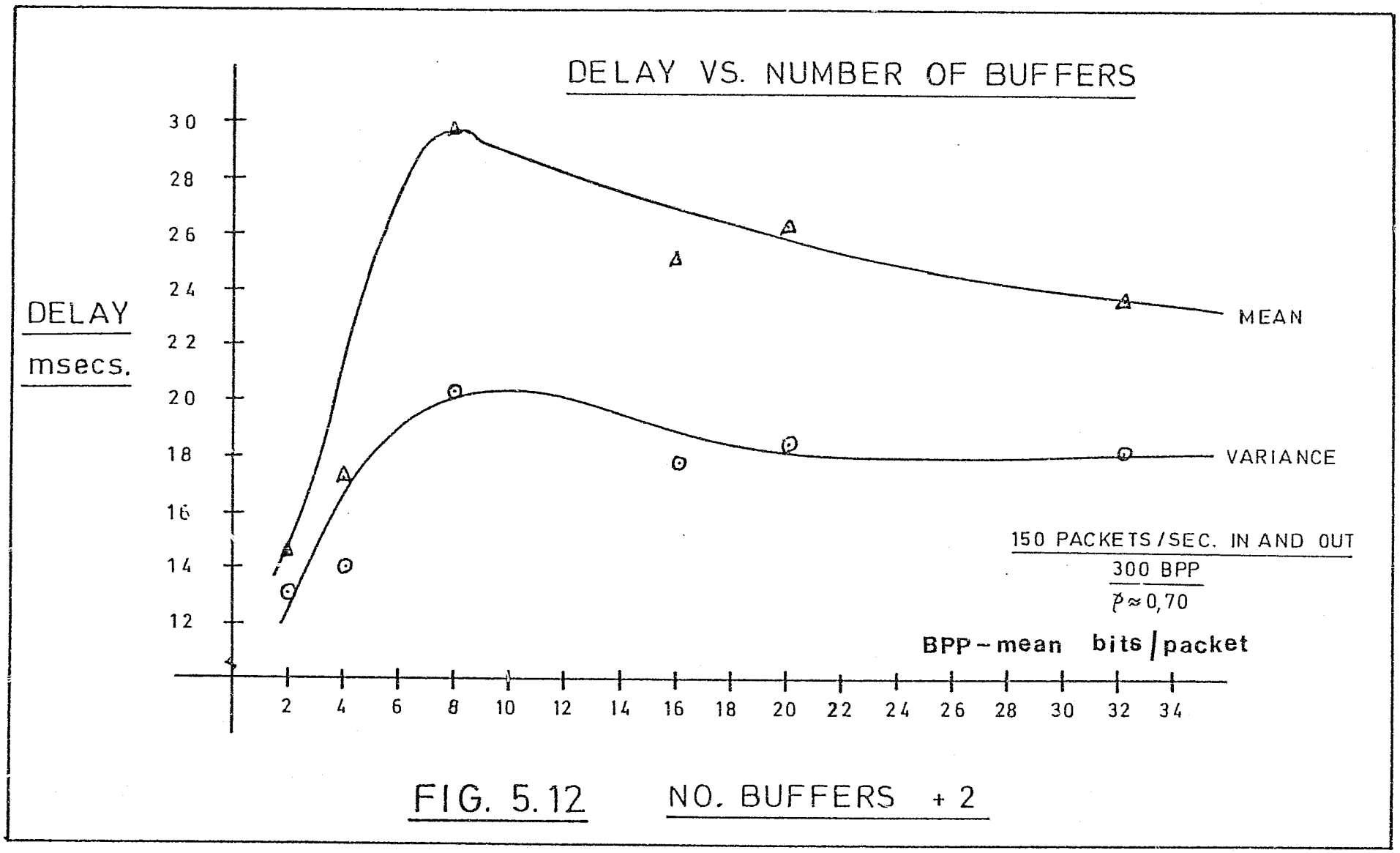


FIG. 5.12 NO. BUFFERS + 2

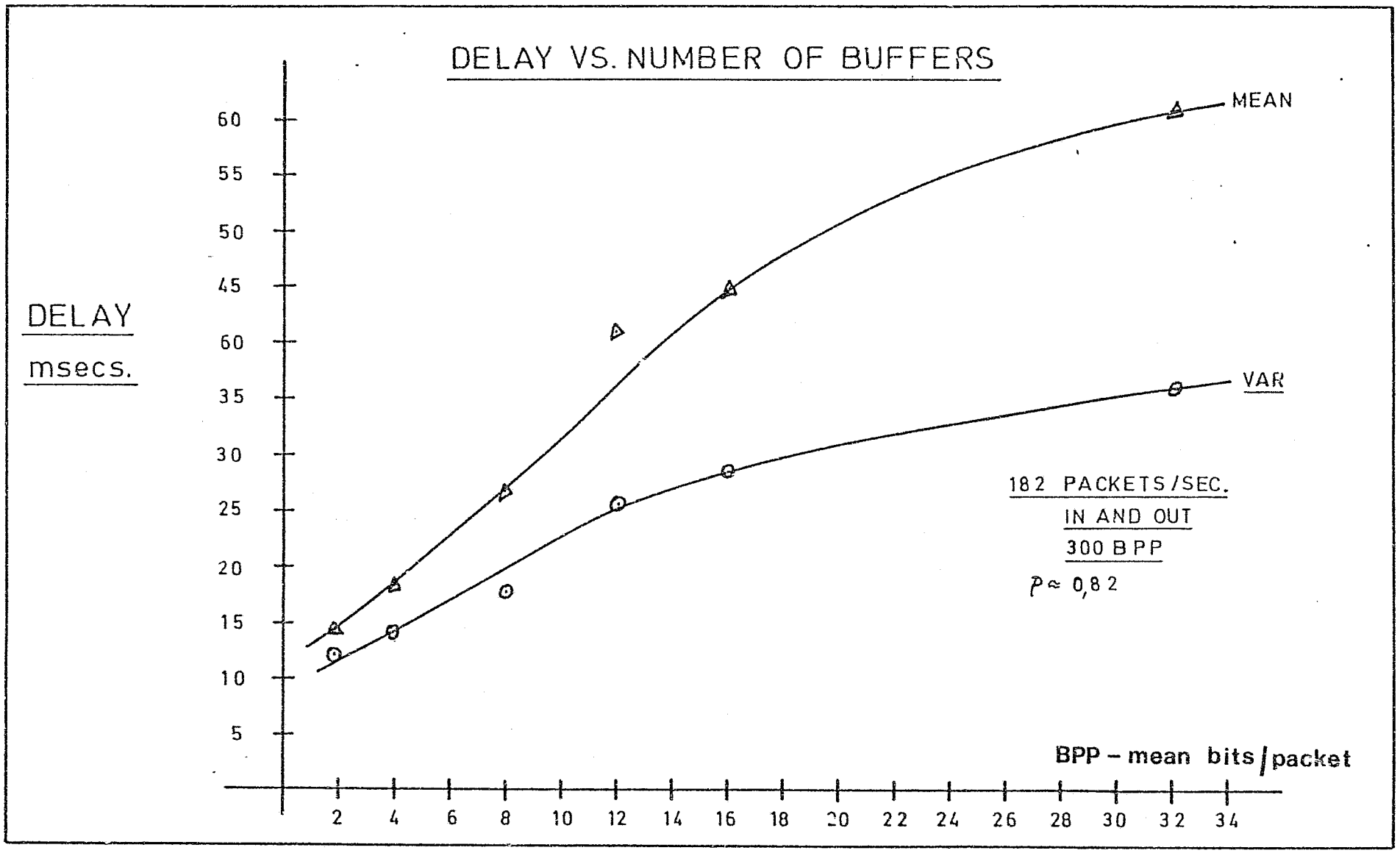


FIG. 5.13 NO. BUFFERS + 2

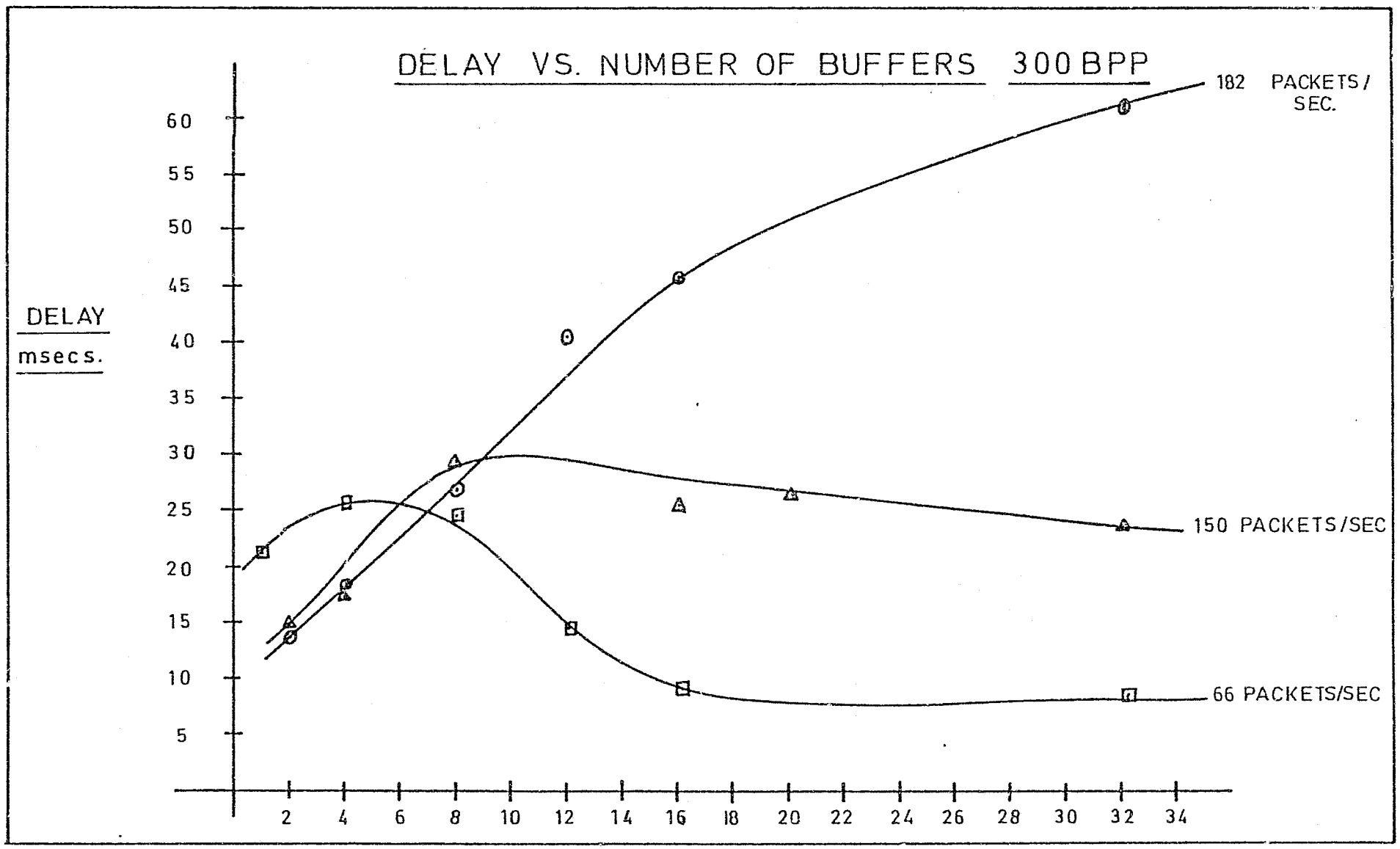

DELAY VS. NUMBER OF BUFFERS 300 BPP
DELAY
msecs.
60
55
50
45
40
35
30
25
20
15
10
5
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
182 PACKETS/ SEC.
P = 0,82
150 PACKETS/SEC
P = 0,7
66 PACKETS/SEC
P = 0,3
NO BUFFERS (+2)

FIG. 5.14

5.4.5 Comparison with Analytic

The increase in response for a corresponding increase in p is illustrated in Figure 5.15. The four simulation points are plotted together with a predicted response based on the Pollaczek-Kintchine equation for M/G/1 queues. Although the M/G/1 queue assumes infinite buffering and an infinite sink for serviced packets, the latter assumption is satisfied by minimizing the number of retransmitted packets (sufficient storage at the destination controller), whilst the former assumption is satisfied by providing sufficient storage at the source controller so that the probability of blocking for incoming packets is small.

It may be seen that the analytic approximation gives a reasonable estimate of the simulation results. The packet transit delay at line utilization factors greater than 0,7 increases extremely rapidly. A delay of 25 milliseconds at p = 0,70 rises to approximately 60 milliseconds at p = 0,85.

Figure 5.16 gives a more complete picture of the channel response-utilization characteristic. A number of simulations for different average packet lengths have been performed. These curves all show the rapid rate of change of response time as the amount of traffic on a channel exceeds the p = 0,70 point. The curves also demonstrate the applicability of the Pollaczek-Kintchine equation for predicting performance based on different mean packet lengths.

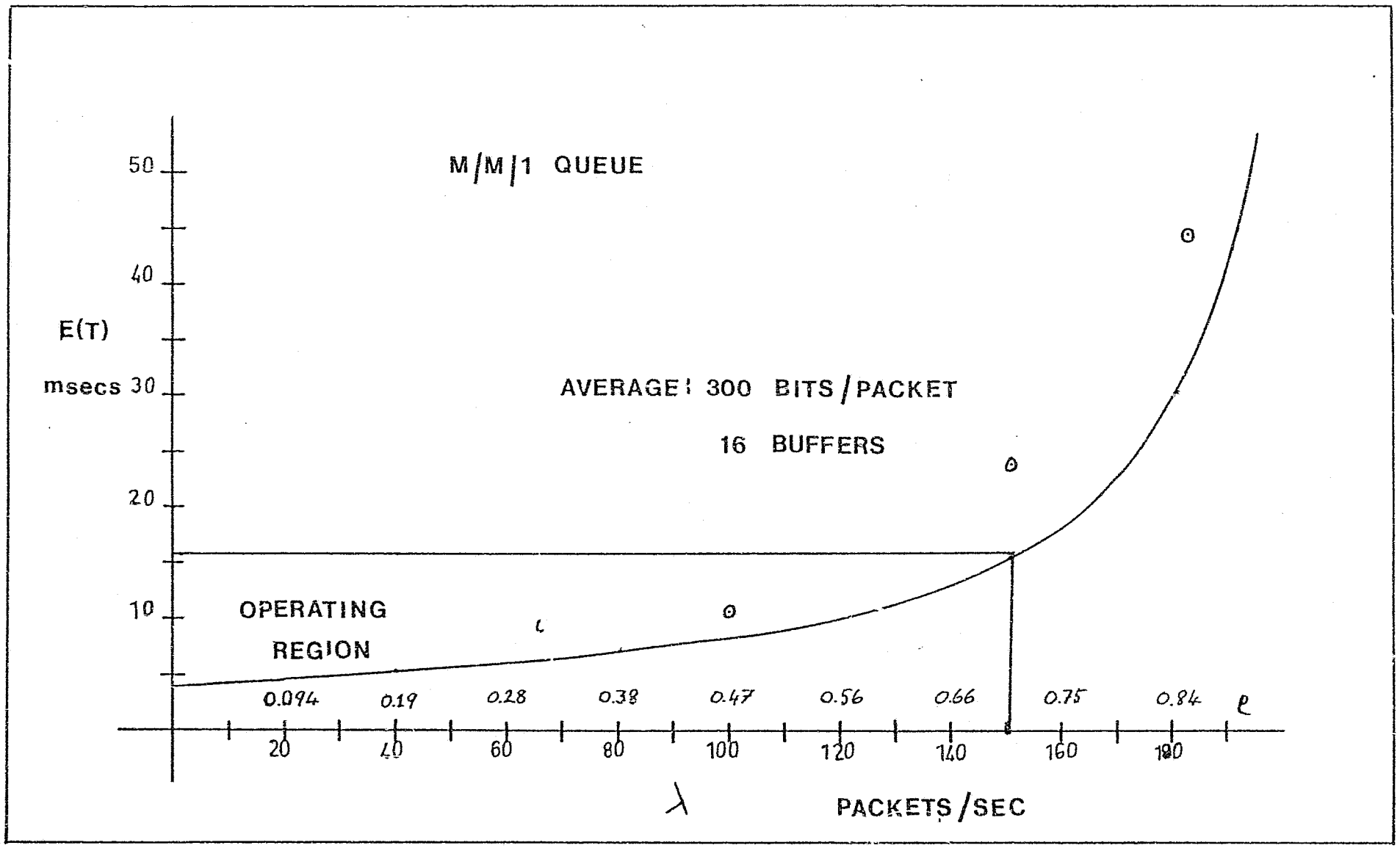

M/M/1 QUEUE
AVERAGE: 300 BITS/PACKET
16 BUFFERS
E(T)
msecs
50
40
30
20
10
OPERATING
REGION
0.094
0.19
0.28
0.38
0.47
0.56
0.66
0.75
0.84
ρ
20
40
60
80
100
120
140
160
180
λ
PACKETS/SEC

FIG 5.15

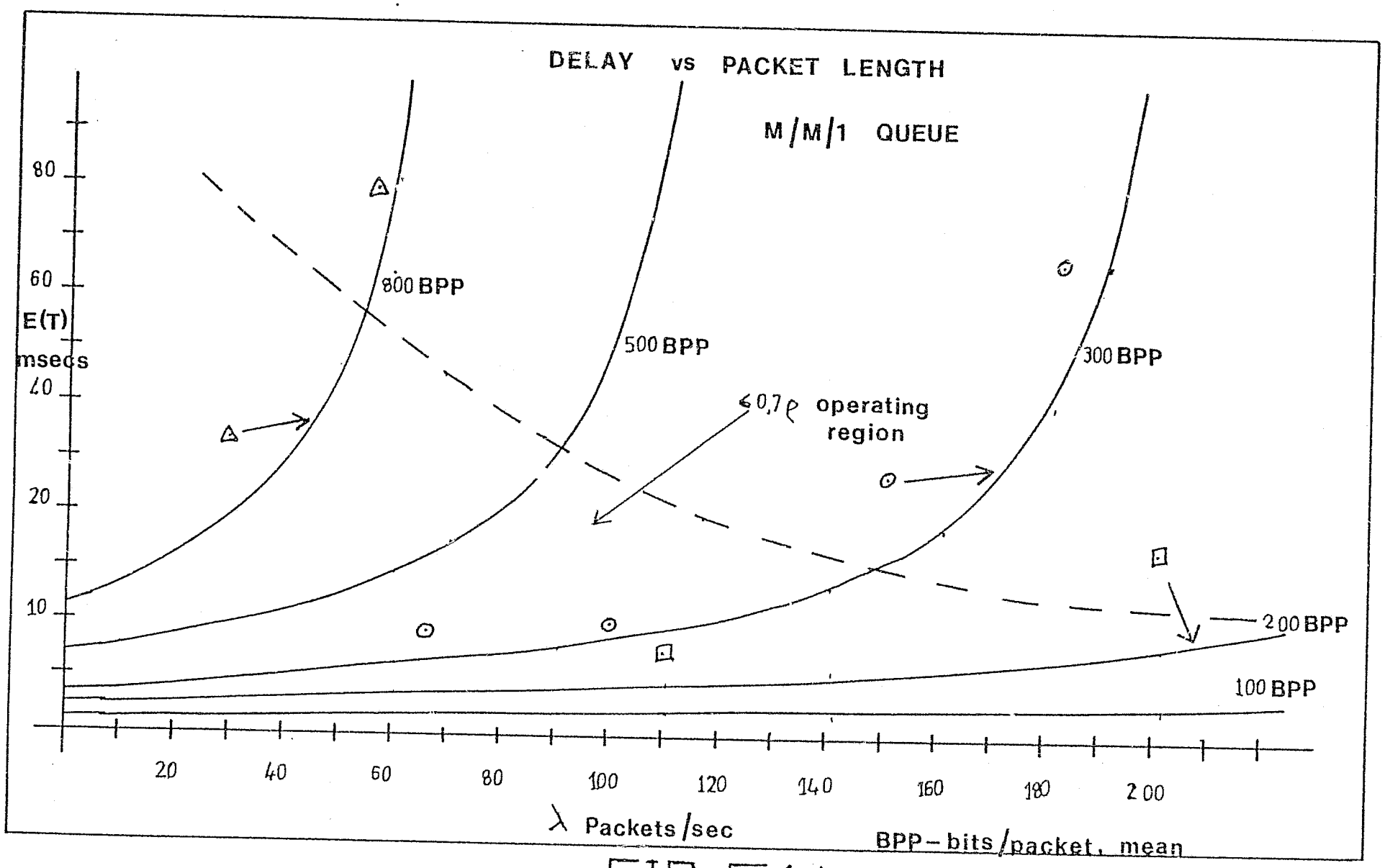

DELAY vs PACKET LENGTH
M/M/1 QUEUE
80
60
E(T)
msecs
40
20
10
800 BPP
500 BPP
300 BPP
200 BPP
100 BPP
0,7 ρ operating region
20
40
60
80
100
120
140
160
180
200
λ Packets/sec
BPP – bits/packet, mean

FIG 5.16

The utilization factors may be calculated from equation (5.1), where C is the line rate in bits per second, λ the average packet arrival rate, and $1/p'$ the average packet length.

$$p = \mu o = \frac{\lambda}{p'C} \tag{5.1}$$

(The Poisson arrival rate and geometric message length distribution are assumed.)

The Pollaczek-Kintchine equation for mean response is given in equation (5.2); see Schwarz.[4]

$$E(T) = \frac{1}{2C}\left|1 + \frac{2-p'}{p(1-\mu p')}\right| \tag{5.2}$$

5.4.6 Discussion on Unequal Input/Output Rates to Link

The results of these simulation runs indicate that the traffic utilization factors of a channel should not exceed some specified limit if excessive delays are to be avoided. The limit chosen is that of p = 0,7. For average lengths of 300 bits, the mean time to transmit packets from the source to the destination station is approximately 25 milliseconds. An 'operating region' has been defined in Figures 5.15 and 5.16 to illustrate this point.

The preceding simulations have been performed on the basis of equal input and output rates for a channel, i.e., $\lambda 1 = \lambda 2$ (as in Figure 5.10). This situation obviously occurs infrequently in practice. Accordingly, simulations for differing input and output packet rates

have been run. These are illustrated in Figure 5.17. Assumed is a 16 buffer channel controller transmitting packets of 300 bits average length. Output rates ($\lambda 2$) of 100, 150 and 180 packets per second have each been plotted for various input rates ($\lambda 1$). The standard deviations are shown in Table 5.1.

The results follow the trend of the curves displayed in Figures 5.11 to 5.16. Irrespective of the channel utilization, a rapid increase in response time occurs when the input rate exceeds the output rate. This result is not as obvious as it might sound, bearing in mind that the channel controllers must reject packets offered (from external sources to the channel) if no buffers are free. The explanation for this characteristic is again based on the retransmission of packets. A lower output rate at the destination controller means that insufficient buffering (at the destination) will consistently occur, leading to the retransmission of an abnormally high number of packets, thus causing the response time to increase rapidly. This explains in part why no further time has been spent on optimizing the time-out period. In Chapter 7 it will be shown that a retransmission interval of from 30 to 40 milliseconds is sufficient.

The results of both equal and differing input/output rates to channels should be borne in mind when incorporating channels into a network.

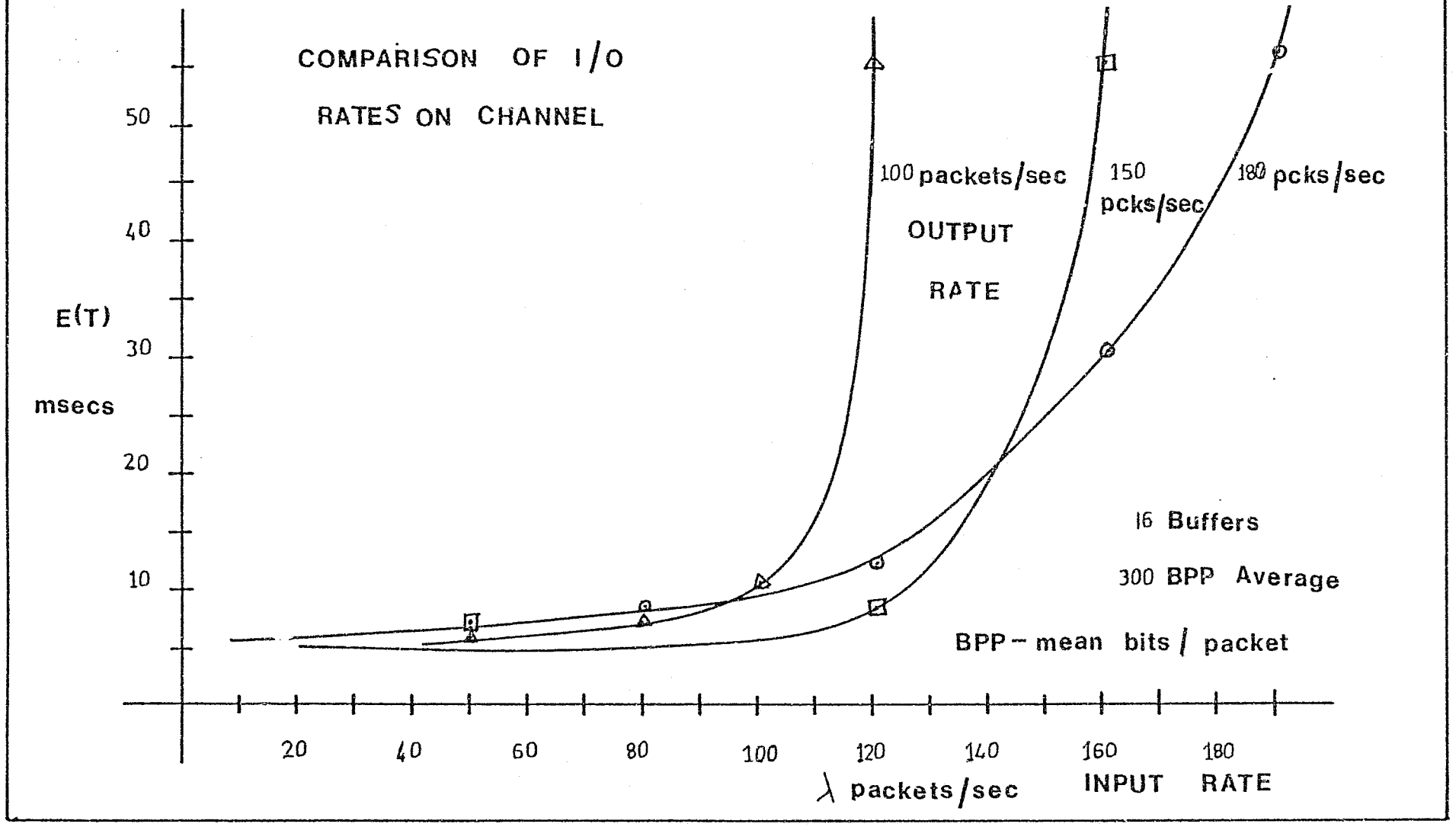

COMPARISON OF I/O
RATES ON CHANNEL
E(T)
msecs
50
40
30
20
10
100 packets/sec
150
pcks/sec
180 pcks/sec
OUTPUT
RATE
16 Buffers
300 BPP Average
BPP – mean bits / packet
20
40
60
80
100
120
140
160
180
λ packets/sec
INPUT RATE

FIG 5.17

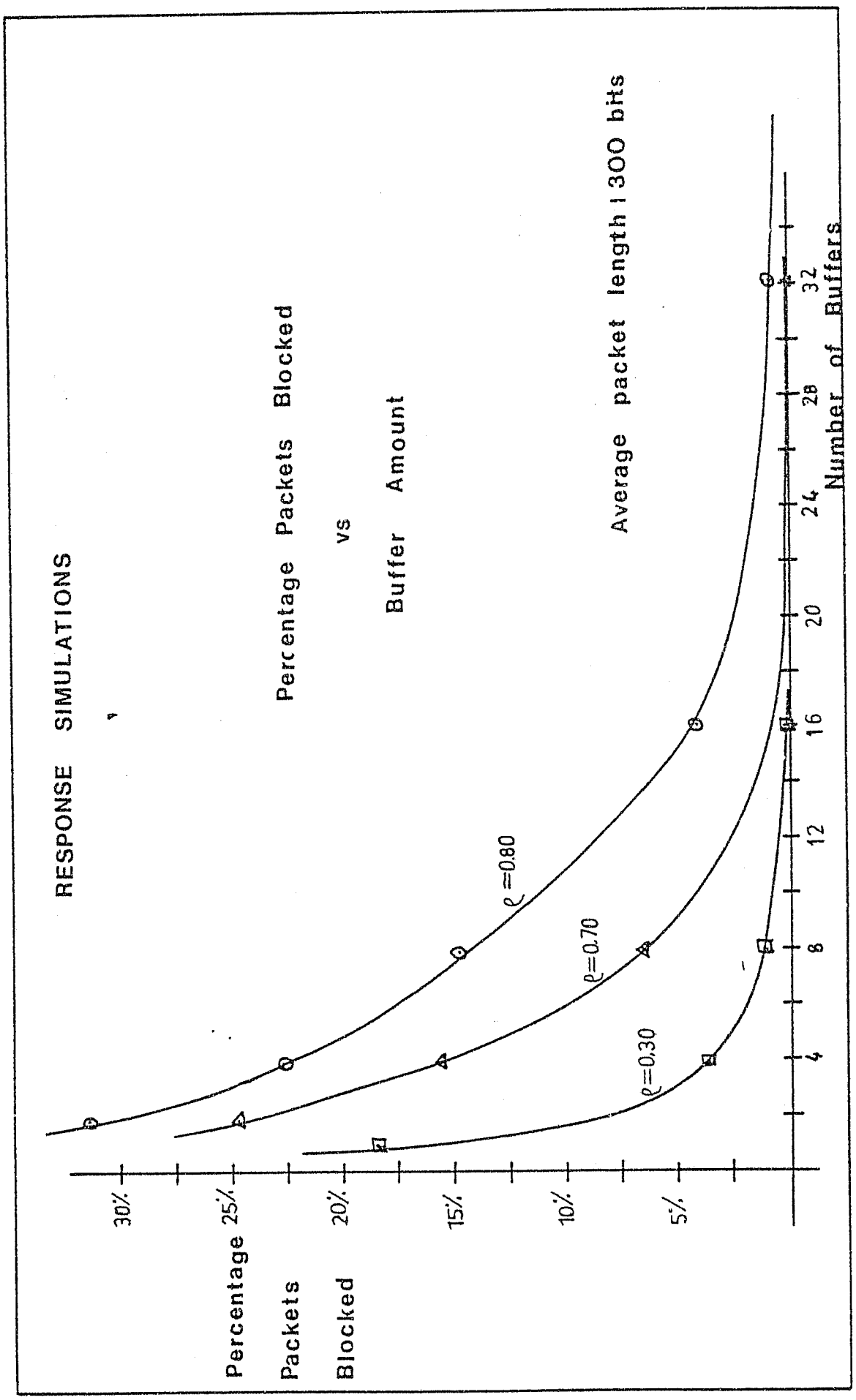
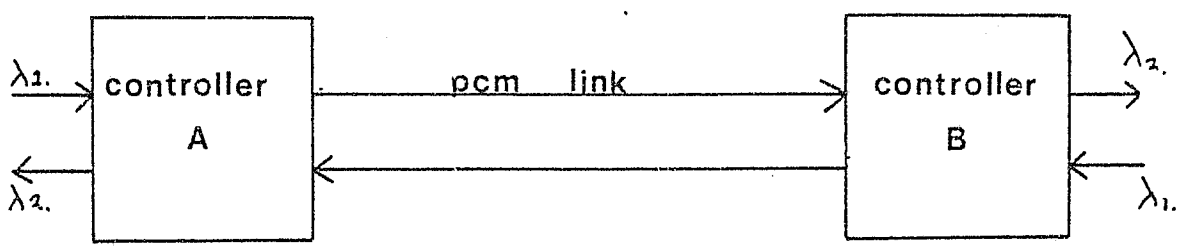

RESPONSE SIMULATIONS
Percentage Packets Blocked
vs
Buffer Amount
Average packet length 300 bits
Percentage Packets Blocked
30%
25%
20%
15%
10%
5%
ρ=0.80
ρ=0.70
ρ=0.30
4
8
12
16
20
24
28
32
Number of Buffers

FIG 5.18



Link rate is 64000 BPS, mean packet size 300 bits,

LINK CAPACITY = 213 PACKETS/SEC

Input Rate $\lambda 1$ Packets/sec	Output Service Rate $\lambda 2$ Packets/sec	Mean Delay Units 10^{-3} seconds	Standard Deviation Units 10^{-3} seconds
50	100	7,3	5,5
80		8,4	6,1
100		11,5	10,8
120		54,1	58,6
60	150	7,3	5,5
120		8,4	6,2
160		54,1	59,5
180		92,5	56,5
80	180	8,2	6,4
120		12,0	8,5
160		29,1	21,7
200		55,3	58,5

Table 5.1 Channel Response

5.4.7 Summary

A channel should not be operated for traffic utilization factors exceeding 0,7. The packets output from a channel should be given priority over the packets input to a channel. A solution to the channel utilization problem has already been formulated in Chapter 4, where the number of packets per logical link and, therefore, the number in the network is restricted.

The channel transit time of a packet is dependent on the processes governing the relative rates of input and output traffic to and from the channel. Although channel flow control techniques may regulate the amount of traffic accessing a channel, these techniques are insufficient in themselves to prevent performance degradations from taking place, since the output rate is determined by processes not under the immediate control of the channel (assuming no packets are to be lost). This implication is studied further in Chapter 6 when the nodal architecture is presented.

5.5 Review

The structure of the channel controller has been outlined in this Chapter. Particular attention was paid to line control procedure and buffer allocation aspects, regarding their effect on data flow between nodes. The problem of packet acknowledgement delay has led to the concept of regarding a transmission line as a resource to be shared amongst a number of logical communicating processes residing on either side of the link in the channel controllers. The storage area has

been partitioned so as to allow traffic from a number of neighbouring nodes to communicate with a reference node without the risk of blocking occurring. Subsequent simulation of the channel has shown that operation should be confined to a region such that the traffic utilization factors should not exceed 0,7 if performance degradation is to be avoided. Finally, it became apparent that performance of the channel is dependent on the ability to remove packets at a sufficiently high rate whereby destination buffering remains unblocked.

CHAPTER 6

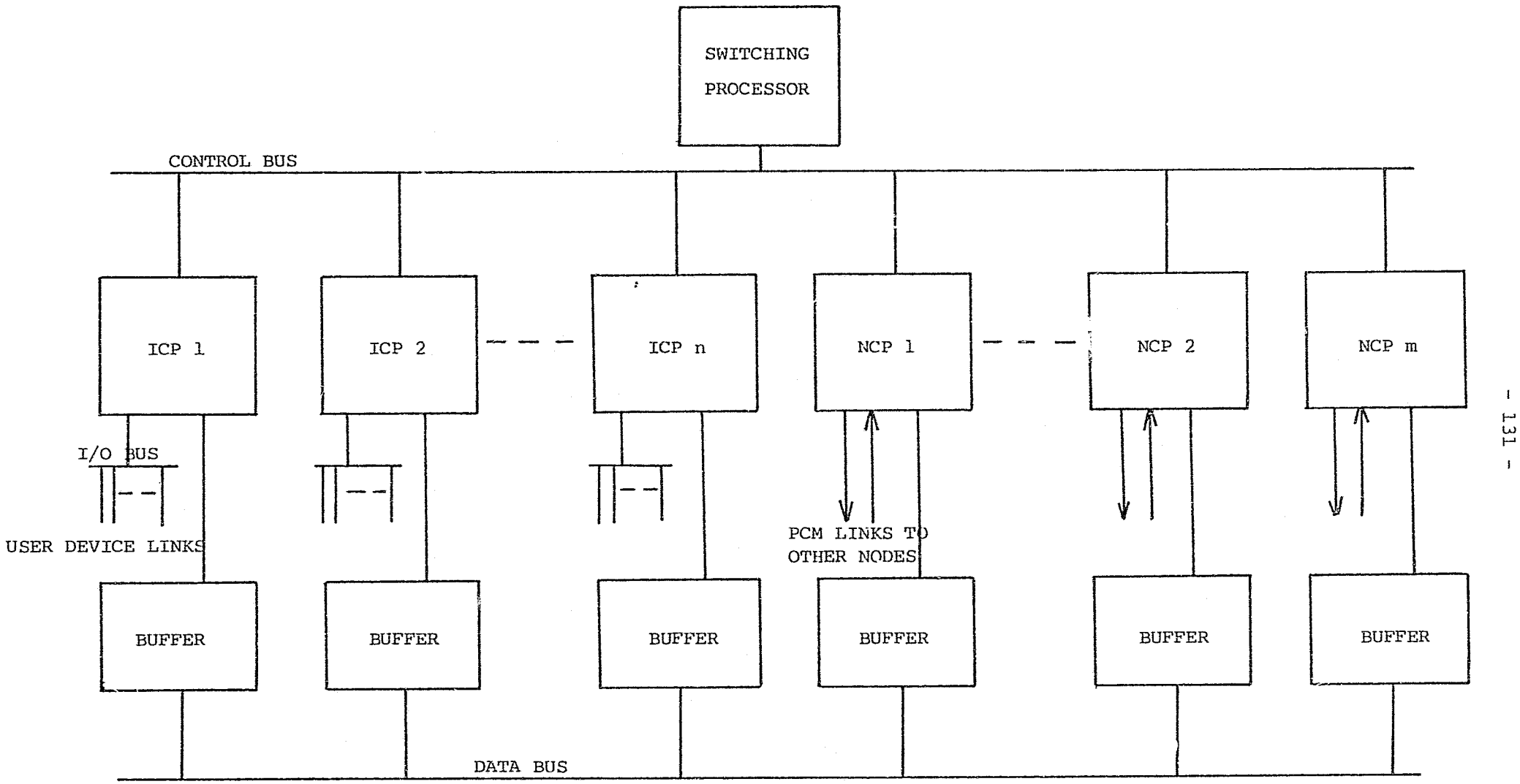
COMMUNICATIONS PROCESSOR ARCHITECTURE

6.1 Introduction

An outline of the proposed communication processor architecture, in terms of functionality of modules, is contained in this chapter. The aim is to provide insight into the technique by which packet interchanges within a node take place. Two types of processor, interface and network, have been described, each performing a specific function. A third module, a 'switching processor', is introduced to coordinate operation of the interface and network processors. The switching processor functions by sampling the states in the nodal modules, processing this information, and then issuing commands based on the current nodal states to control packet transmission between ICP's and NCP's. The chapter concentrates on the algorithms required in order that the switching processor may perform the above stated function.

6.2 Outline of the Node Architecture

The architecture proposed for a packet switching communication processor is shown in Figure 6.1. The architecture consists of three types of processor, the interface, network and switching modules. Further, a data bus and control bus serve to link together the modules. The two buses operate asynchronously, each serving a specific function. The data bus allows for packet transmission between ICP's and NCP's; communication between the switch processor and all other units, i.e., ICP's and NCP's, takes place via the control bus. Finally, interface



ICP: INTERFACE COMMUNICATION PROCESSOR

NCP: NETWORK COMMUNICATION PROCESSOR

Figure 6.1 Communication Processor Architecture

buffers connect the ICP's and NCP's to the data bus.

The ICP communicates with nodal user devices via an I/O bus or switching matrix, enabling it to interact with users over a number of serial links. Messages are formatted into packets, generally these packets will be transmitted to another processor, either interface if the destination host is attached to the same node as the source host, or network processor if the destination host is connected to a different node. It is also possible that source and destination processors make use of the same ICP. The ICP then simply serves as a buffering module, possibly providing code translation functions.

The network processors are used to transmit data between nodes. Each processor is connected to a set of full duplex 64 KBPS pulse code modulation-based links, and transmits packets according to the protocol as described in Chapter 5. Data passing through an intermediate node, must be transmitted from one NCP to another, the transfers taking place over the data bus.

The data bus thus serves as the medium via which data is shifted within the node. Whether a packet enters, leaves, or is in transit in the network, in each case it must make use of the data bus. The medium might equally well be a common memory buffer accessible to all nodal processors.

The data bus rate must be high in order not to become a bottleneck. To aid the processor in transmitting packets into - or receiving packets from the bus, a buffer interface unit is inserted between processor and bus. The buffer acts as a high speed translation unit. The buffer unit transmits a packet at a time to a similar destination unit. A processor requiring to transfer data to another module informs the buffering unit of the starting address and length of the packet to be sent, including the destination address. The I/O operation is therefore envisaged to be direct memory access. The buffer is divided into a transmitter and receiver section, each capable of holding a single packet. The receiver memory is essential, since a processor may not be in a position to receive a packet, e.g., the processor accessing memory and busy on an uninterruptable task.

The transmission of packets over the data bus is a controlled operation in which each processor is given a turn to access the bus. However, the processor does not know when this access will occur. To leave it free to continue its tasks the buffering unit, once it has been informed which packet to transmit, will take control of the transmission aspect when the processor's access turn is due.

Bus multiplexing is shown in Figure 6.2, where each buffering unit transmits data from its transmission section to a receiver section in another unit.

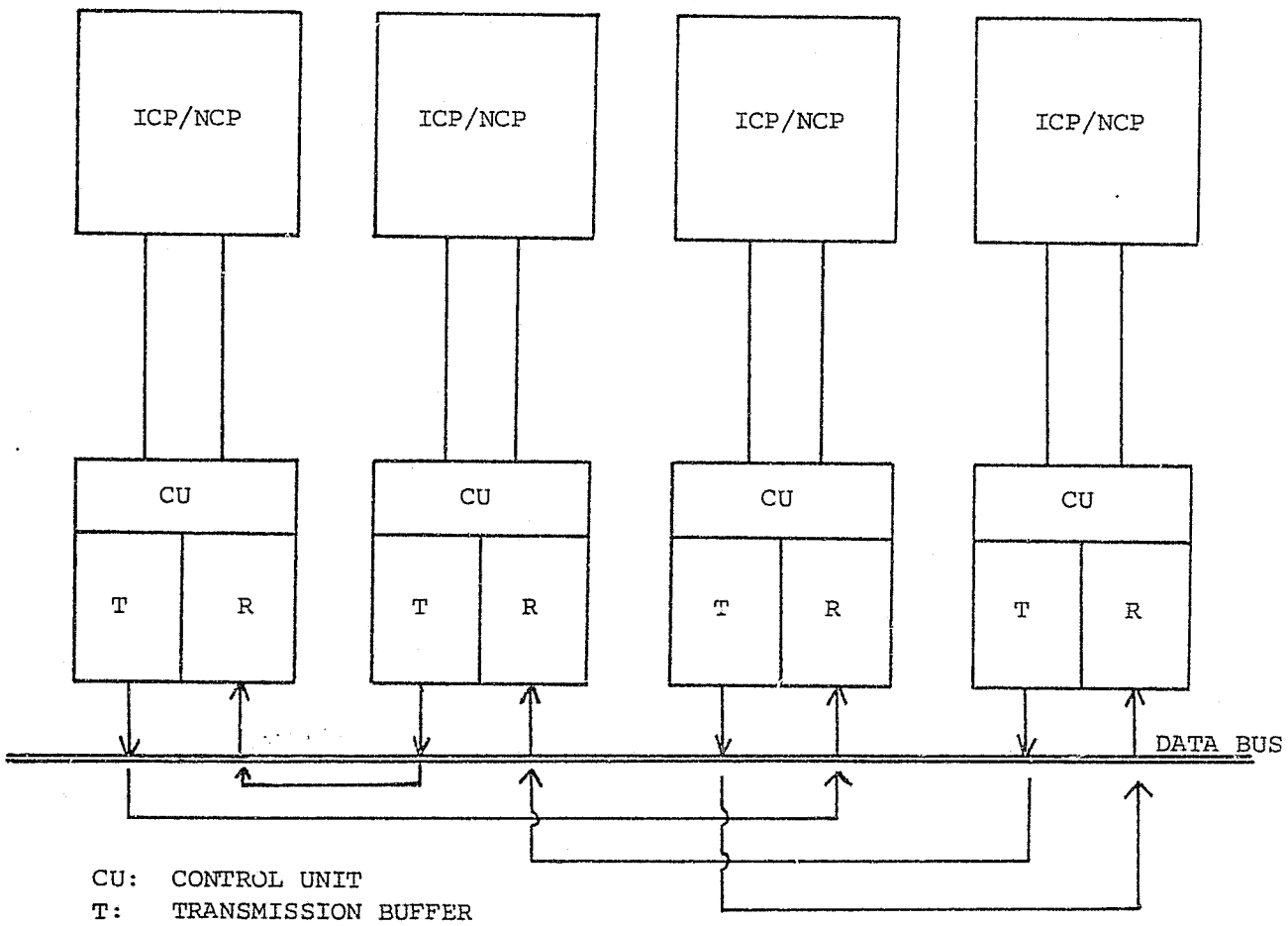


Figure 6.2 Bus Multiplexing

6.3 The Bus Access Problem

The accessing of the data bus must be given careful consideration, since an excessive number of conflicts by processors transmitting simultaneously will result in a significant lowering of the nodal throughput. Systems whereby processors may transmit at will, result in inefficient operation as is shown in the following.

In Schwarz[6] it is shown that a pure Aloha channel (one where processors transmit at will over a common channel) can attain a utilization factor of no more than 0,18, after which instability

results. For slotted Aloha, wherein packets may only be transmitted within time slots, the utilization can be increased to 0,368. Assumed is a fixed packet size and a Poisson transmission distribution.

Channel instability results when the channel throughput is so high that a significant number of collisions occur; these collisions require the retransmission of the packets involved which, in turn, increases the number of packets to be transferred across the bus. This, in turn, results in more collisions. One solution used is to provide the transmitter with a detection unit to determine whether a packet is in transit across the bus. Another is a slot reservation scheme where the utilization factor may theoretically be increased to 0,8.[6]

These schemes are not followed up here since they do not satisfy a criterion central to the nodal operation: that the local flow control mechanism be bound up with the bus access control technique. This will be subsequently explained.

6.4 The Function of the Switching Processor

It is necessary that some knowledge of the nodal processors' states be available so that processors can be made to coordinate their transmission activities to minimize the number of conflicts. By 'states' is meant:

* knowledge of buffer availability,
* number of packets to be transmitted,
* the destinations of the packets.

If the states of processors can be determined at specific intervals, it is possible to arrange orderly data bus accessing and to ensure availability of buffering at the destination processor. It is clear that the exchange of information and the resolution of conflicts amongst processors themselves, would be highly inefficient. Accordingly, a processor has been dedicated specifically for this task - henceforth referred to as the 'switching' processor.

The control bus is used to transfer status information and switch processor commands between the processing and control modules. Operation of the node is thus centralized and is dependent on the continued and correct functioning of the switch. A dual set-up may be necessary for reliability purposes.

The switching processor maintains the routing tables. A processor requiring to transmit a packet informs the switch of the packet's destination (logical process number and nodal address). The switch sends the destination processor address to the transmitting processor. Certain error detection and recovery techniques are executed by the switch; ideally it should be able to isolate defective units and arrange for continued nodal functioning without them. It should have some capability for serving in a network reconfiguration role if nodes or internodal links fail. Abnormal conditions such as congestion must be detected and steps taken to ensure that performance degradation does not result. Specific recovery procedures have not been implemented in the simulation, however. Simulations for traffic control have been

performed assuming well-behaved nodal modules.

Control of nodal operation takes place as envisaged in Figure 6.3. At regular intervals the switching processor obtains the nodal status information, processes it according to some defined algorithm, and then issues commands to the network and interface processors. The switch instructs as to which packets to transmit and when the transmission is to take place.

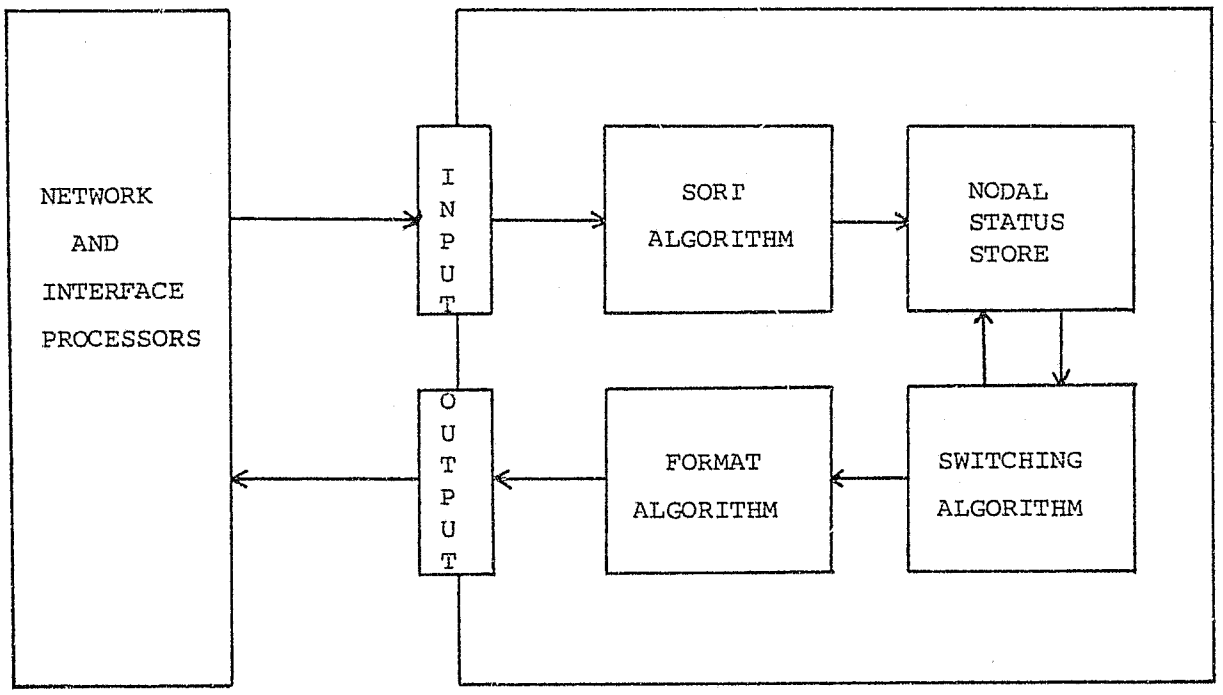


Figure 6.3 Switch Operation

The switching process consists of four parts. The sorting algorithm obtains the status information and sorts it into a form convenient for later processing. The data is subsequently stored together with data from previous intervals in the 'status store'.

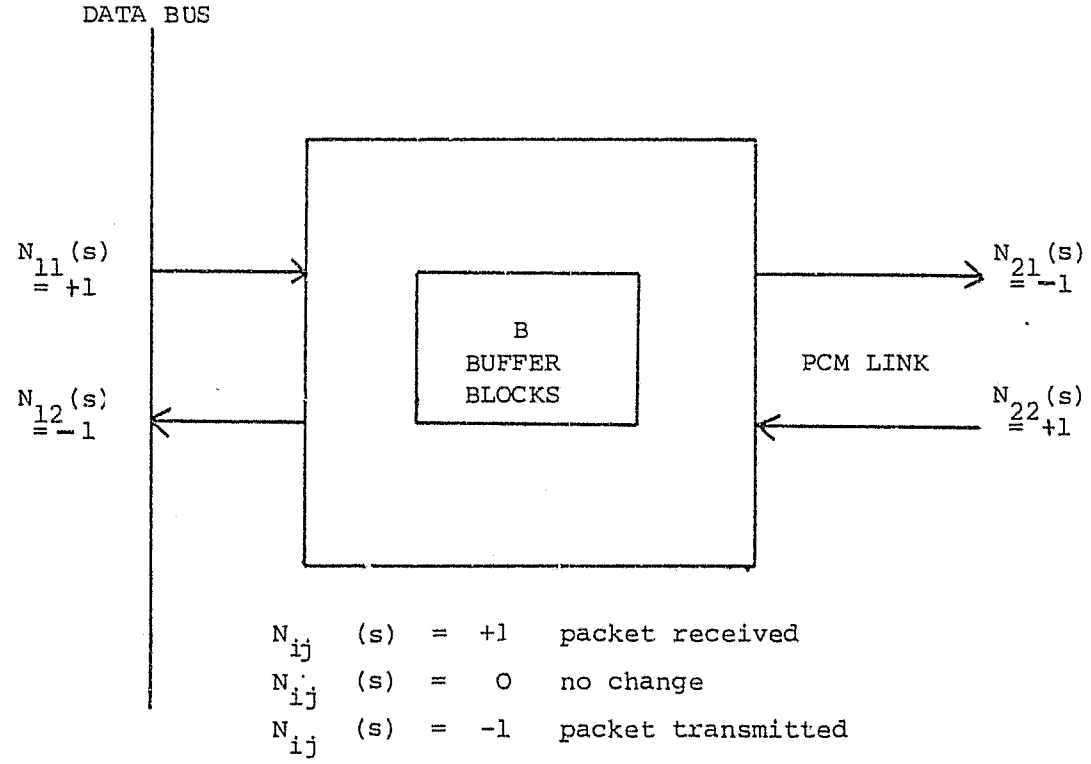
The switching algorithm processes this data so as to coordinate the activities of the NCP's and ICP's. It is noted that the inputs to the node are random, in the sense that message lengths and interarrival times are stochastic quantities. Finally, the format algorithm translates the switch algorithm's results into a form suitable for output to the nodal processors. This sequence of operations is repeated at regular intervals, providing the node with some form of adaptability to varying traffic conditions.

The entire algorithm must be executable within a certain time so that status information does not become so outdated that incorrect decisions are made. In addition, it must be executable by a unit with limited processing capability. The algorithm serves two well-defined functions. It must multiplex the data bus between ICP and NCP modules, and it must exercise control over data flow within the node.

6.5 Nodal Timing

The basic unit of time chosen for nodal operation is the 'cycle'. A cycle is defined to be the time required to transmit a maximum length packet over a 64KBPS channel. The network processors deal with such channels and ideally are required to transmit at the rate of one packet per cycle. In practice this rate can never be attained due to the queueing delays and acknowledgement requirements of the NCP operation.

The switching processor samples the nodal states approximately once per cycle (implying that the algorithm must be executable within this time). It is believed that sampling every cycle is sufficient for the switch to maintain 'control' over events, since during one cycle no more than one packet may be transmitted from any one neighbouring node to the node under consideration.



N_{ij} (s) = +1 packet received
N_{ij} (s) = 0 no change
N_{ij} (s) = -1 packet transmitted

Figure 6.4 NCP Buffering Status

Let the buffer status of an NCP be denoted by N(s) where (s) is the address of the processor. Then during a single cycle N(s) may

indicate a maximum change of

$$0 \leq \Delta N(s) \leq |2|$$

as shown in Figure 6.4.

It is assumed that each packet occupies one buffer. The unknown quantity, $N_{22}(s)$, over which the switch has no control since it has no knowledge of when packets may need to be received, can take on values of only

$$N_{22}(s) = 0 \quad \text{or} \quad N_{22}(s) = +1$$

during a single cycle. Since N(s) is therefore reduced to its most primitive form, the switch may observe each change of state in the NCP's and produce appropriate commands.

A large number of packets may be generated by ICP's during a cycle - the arrival of a number of long messages at the ICP. These packets do not, however, affect the response and throughput of the switching section of the network.

It is noted that the cycling operation of nodes is asynchronous.

The data bus rate may now be estimated. The requirement of the bussing structure is that each of the nodal processors be able to transmit packets at the instantaneous rate of 64KBPS - the interface

buffering units ensure the speed translation process. The data bus, used in a multiplexing fashion, then has a word rate of

$$\text{RATE} = |p \times 64|/q \qquad \text{K words/sec} \qquad (6.1)$$

where p is the number of processors that transmit during a single cycle, q is the word length of the bus in bits.

$$1 \leq p \leq m$$

where m is the number of processors attached to the data bus.

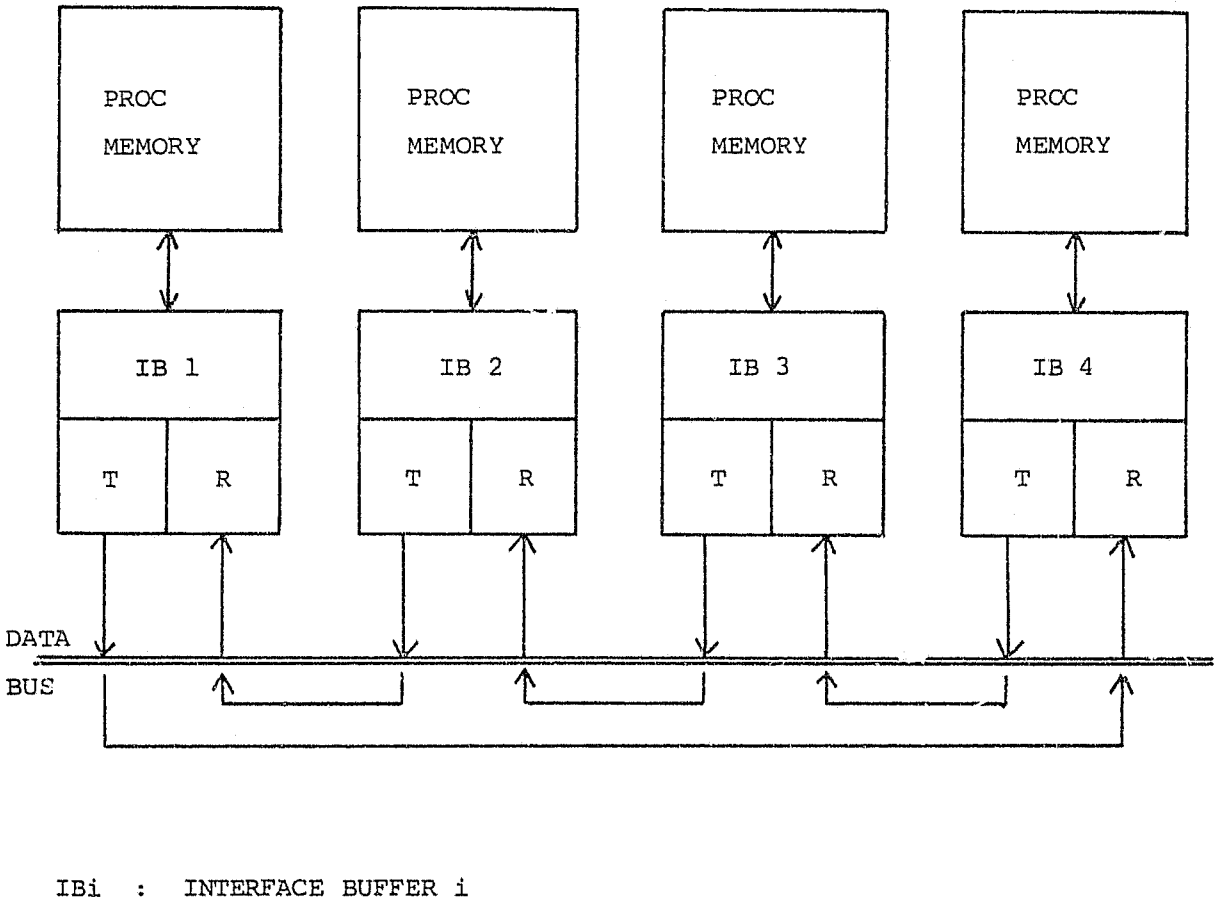
6.6 The Cycle Diagram

A 'cycle diagram' is produced by the switching processor to indicate which NCP or ICP is to transmit. To prevent packet loss, each processor is constrained to transmit and to receive only one packet per cycle. Due to the high bus speed, two modules transmitting successively to the same destination, would result in either the first packet being over-written, or the second to be lost. Figure 6.5 shows the transmission process.

Two rules are to be adhered to if conflicts on the bus are to be avoided:

(a) no processor transmits to more than one processor in a single cycle,

(b) no processor receives from more than one processor in a single cycle.



IBi : INTERFACE BUFFER i

T : TRANSMITTER

R : RECEIVER

Bus rate ≃ (4 x 64)/16 Kwords/sec

≃ 16 Kwords/sec

For a packet size of 1000 bits: cycle time is

(1000 bits/64000 bits per second)

= 15,6 milliseconds

Every 15,6 milliseconds four 1000 bit packets can be transmitted via the bus.

Figure 6.5 Multiplexing of Four Packets

Application of these rules leads to the cycle diagram, a graphic description of the multiplexing process in a node. Figure 6.6 shows the diagram, where the address of the transmitting processor is indicated along the Y-axis, the receiving processor along the X-axis. For example, X(#n1) indicates that processor P2 is to transmit packet n1 to processor P4 during cycle α. The diagonal of the cycle diagram is not used.

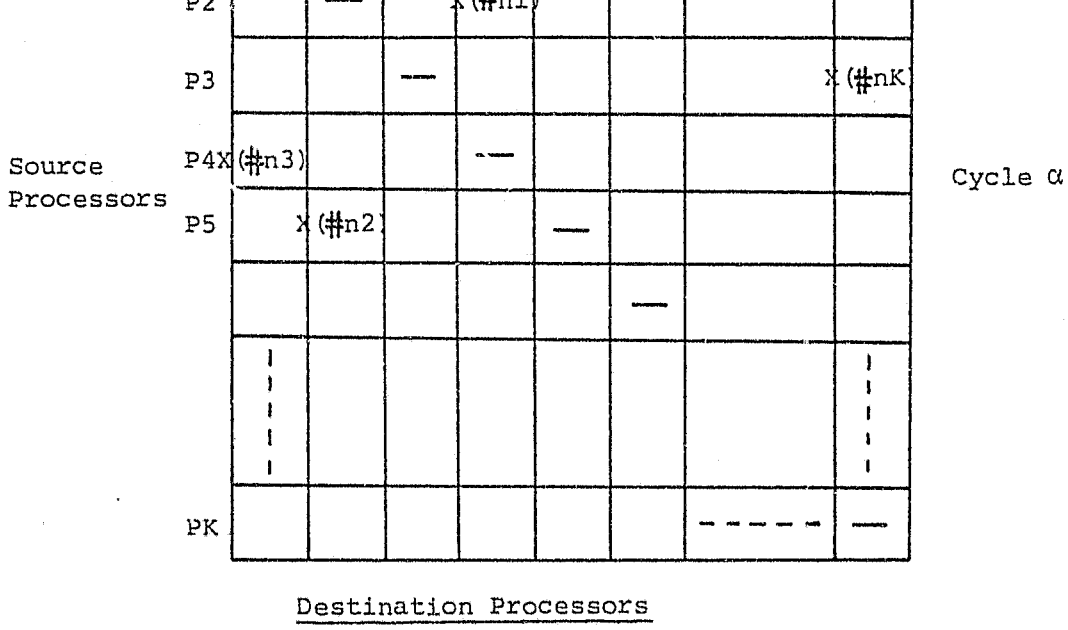


Figure 6.6 Cycle Diagram

The two rules given above are illustrated in Figure 6.7, where diagrams for cases (a) and (b) indicate the two situations to be avoided for efficient multiplexing.

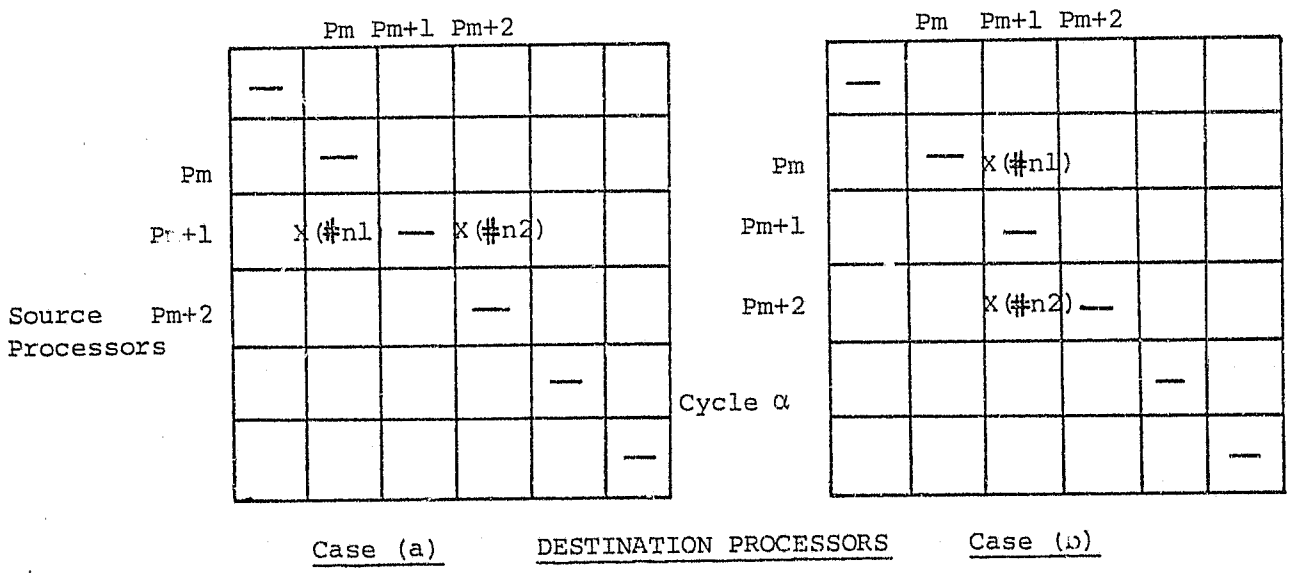


Figure 6.7 Bus Conflicts

The switching algorithm must produce a cycle diagram, whereby the intersection lines of any packet X(#ni) with coordinates (P_R, P_s), do not intersect the coordinates (P_X, P_Y) of any other packet X(#Nj) displayed in the diagram.

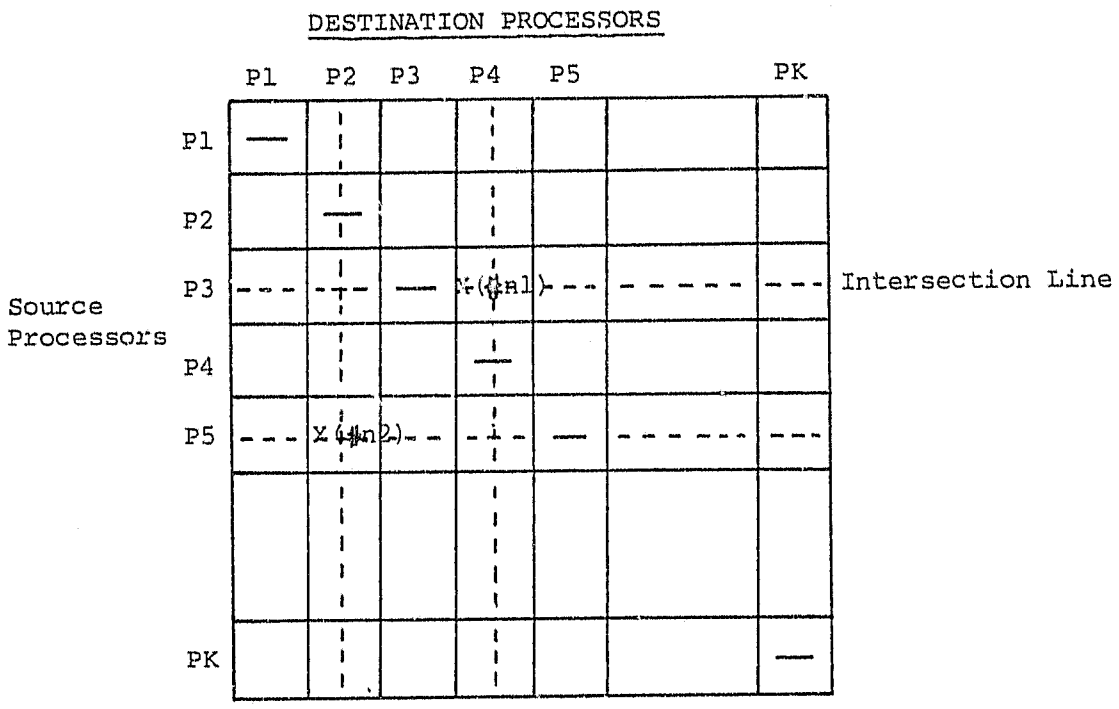


Figure 6.8 Switch Algorithm Criterion

In order to preserve the integrity of the nodal status stored in the switch, it was decided that the switch would poll the ICP's and NCP's. If the nodal processors were allowed to update the switch status table, problems of data integrity might arise. A processor updating the table, whilst the switching processor is generating a cycle diagram from the stored data, may result in the table giving an incorrect view of the nodal status. Such an incorrect table is difficult to rectify on-line, if the node is at the same time executing communication tasks. Updating the table by the NCP's and ICP's interrupting the switch, may result in the switch spending an excessive amount of time generating the cycle diagram. The polling technique has therefore been used.

A cycle is subdivided into three parts; the switch fetches the status data, processes it, and sends commands to the ICP's and NCP's. It is possible to generalize so that a number of cycle diagrams are produced utilizing only a single fetch and output operation (Figure 6.9). A multiple cycle operation is termed a 'period'. Implementation aspects might dictate that fewer fetch and output operations be used so as not to cause an excessive number of interrupts in ICP and NCP functioning.

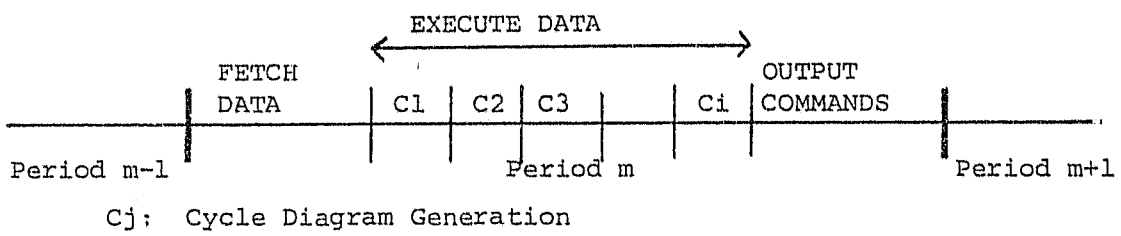


Figure 6.9 Cycle Operation

It is noted that packets are, in general, of variable length, although a maximum size is specified; also the execution period for the switching algorithm is dependent on the amount of data to be processed. The cycle diagram method is intended to establish upper-bounds for nodal resource service rates in order that implementation figures be determined for data bus rates, processor throughput and response, etc.

6.7 Description of the Sort Algorithm

This section discusses the method used to sort the polled processor data into a form suitable for processing by the switching algorithm.

6.7.1 Classification of the Status Data

The information content may be divided into packet description and network processor status groups. The latter consists of a tabulation of the number of unoccupied buffers in each of the NCP's, and includes a partition flag (indicating whether the NCP of a neighbouring node has become congested or not). These parameters are obtained every poll period. An example of a partition parameter:

|NCPi| |PARTITION j| |FLAG|

The partition function, j, for network processor, i, is allocated a particular address in store, the contents of which indicate whether the partition of a particular NCP is occupied or not.

The packet group has the following format:

PACKET IDENTIFIER

SOURCE ADDRESS

DESTINATION ADDRESS

TYPE

The packet identifier is a name associated with a particular packet in a particular processor, as allocated by the ICP or NCP. The source address corresponds to that of a module in the node. Packet identifications need thus be unique only for a particular ICP or NCP. The destination address corresponds to the logical process number of a source-destination host logical link. For example:

IDENTIFIER i

ICPj LOG LINK n

TYPE K

The switch will cause packet i to be transmitted via the data bus from ICPj to NCPq. It is assumed that NCPq controls the channel over which the packet must be sent to arrive at its destination node. The above is done by the switch making use of its address table. Identifiers may be implemented by modulo-n sequences, where n is the maximum number of packets that can be processed by a module. The addresses and packet types may be described by simple n-bit codes.

The above addressing method is used by the switch to control data bus multiplexing. The packet type and the status information is used by the switch for flow control. Packets may be differentiated according to a number of different criteria, discussion here being restricted to the types implemented in the simulation.

The types used are:

(a) network control packet

(b) data packet: (i) single packet

(ii) multipacket

(iii) multisegment

These types correspond to the packet groupings for the memory reservation algorithm presented in Chapter 4.

The above types serve mainly in an auxiliary manner, e.g., that control packets are given greater priority than data messages. An additional packet classification, as used by the local flow control method, has been implemented. These are:

(a) ICP - ICP

(b) ICP - NCP

(c) NCP - NCP

(d) NCP - ICP

Case (b) corresponds to all packets requiring transmission via the data bus from an ICP to an NCP, i.e., these packets are in the process of entering the network. Case (c) corresponds to packets in transit, case (d) for packets leaving the network. Case (a) is included, as the source and destination hosts may be connected to the same node. These types are stored in an associative manner with the packet identifier; they are determined during the execution phase of the sorting algorithm.

6.7.2 Sort Algorithm

To facilitate the handling of packet descriptor data, an elementary queueing structure has been implemented using pointers. The structure for the simulation of each node is shown in Figure 6.10.

As indicated in Chapter 2, packets are assigned priorities that increase with time. The switching algorithm attempts to transmit the higher priority packets before those of lower priority. Array Q is used to implement and maintain these priorities. Array QLIST maintains the additional information concerning each packet for transmission needs. The packets in Q are arranged in order of priority.

$$\text{PRIORITY } |Q(i)| \geq \text{PRIORITY } |Q(j)|$$

$$\text{where } i < j \qquad i = 1 \ldots\ldots \text{QUS}$$
$$j = 1 \ldots\ldots \text{QUS}$$

The maximum number of locations available in the array for status storage is defined as QUS. Packets belonging to different groups are shifted in the Q array according to the size of the priority increment allocated to each group in each cycle interval. Rather than shifting all the data describing each packet, only the priority data in the queue array is dynamic. Status data for each packet is stored in the first free location in QLIST. Pointers are used to link the data in QLIST to that in Q, a bidirectional link is necessary.

The source and destination addresses in QLIST refer to the module addresses in the node. Two types of modules are distinguished: ICP's and NCP's. The partition number refers to the partition of the

ARRAY QLIST

contents

SOURCE PROCESSOR ADDRESS

PROCESSOR TYPE

DESTINATION PROCESSOR ADDRESS

PROCESSOR TYPE

PACKET IDENTIFIER

PARTITION NUMBER

WEIGHTING FACTOR

POINTER VALUE TO Q

*addresses of network processors

ARRAY Q

contents

POINTER VALUE TO QLIST

PRIORITY VALUE

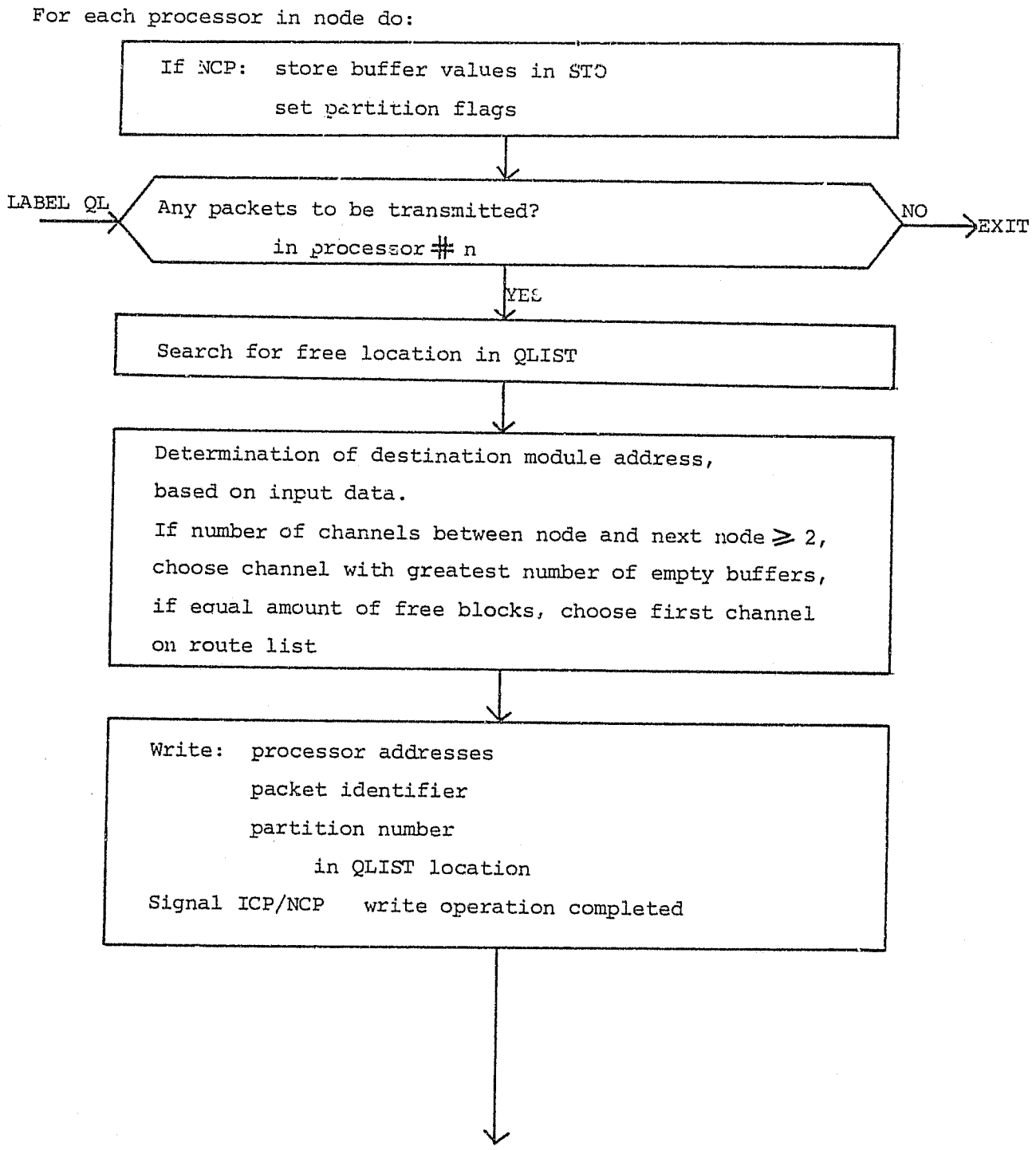
ARRAY PARTNP

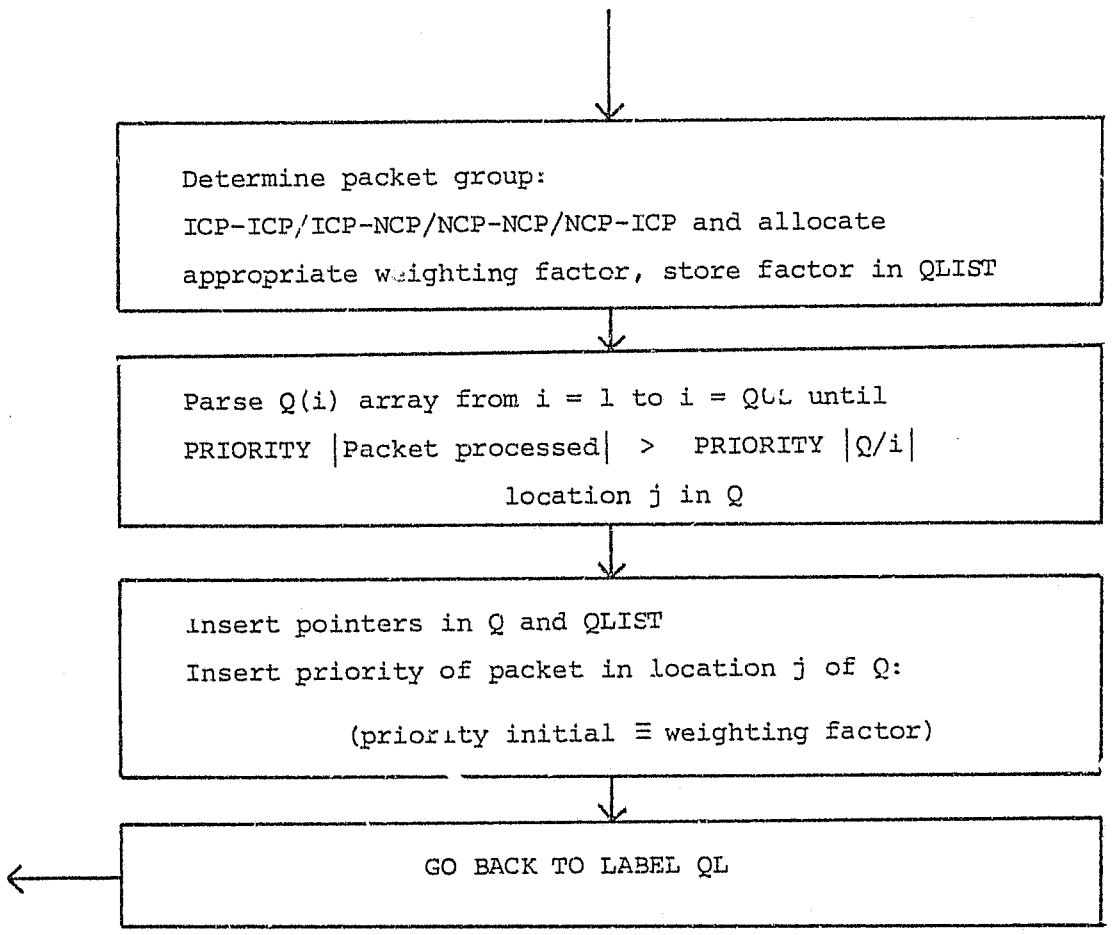
contents: addresses* + flag settings

ARRAY STO

contents: addresses* + storage status

Figure 6.10 Nodal Status Table


For each processor in node do:
If NCP: store buffer values in STO
set partition flags
LABEL QL
Any packets to be transmitted?
in processor # n
NO
EXIT
YES
Search for free location in QLIST
Determination of destination module address,
based on input data.
If number of channels between node and next node ≥ 2,
choose channel with greatest number of empty buffers,
if equal amount of free blocks, choose first channel
on route list
Write: processor addresses
packet identifier
partition number
in QLIST location
Signal ICP/NCP write operation completed



NB: Locations of Q/QLIST initialized to zero

Figure 6.11 Sort Algorithm

destination NCP to which the packet is allocated, as discussed in Chapter 5. The weighting factor is the increment unit assigned to each packet, as determined by its type. The priority value in Q is incremented in each cycle by an amount equal to the weighting factor stored in QLIST.

An outline of the SORT algorithm is given in Figure 6.11, the simulation code for which appears in Appendix B and is executed every poll period. The algorithm shown is very simple.

If an array overflow occurs during simulation, the simulation is terminated and an appropriate error message printed. Concerning possible action for nodal implementation, the SORT algorithm is terminated and no further processors are polled. The SWITCH and FORMAT algorithms proceed as normal. When packets are transmitted via the bus, their QLIST descriptions are deleted. Locations will, in general, be available when polling is initiated again on the following cycle, either starting from the first processor on the polling list, or from the processor for which storage overflow occurred.

6.8 Description of the Switch Algorithm

The function of the switching algorithm is to generate the cycle diagram from the data stored in arrays QLIST and Q. This section discusses the working of the algorithm.

6.8.1 Notation

In order that the cycle diagram may be produced each cycle, the switch requires a temporary workspace to store data whilst processing the contents of the QLIST and Q arrays. This space is denoted by SWITCH |I,J|, where

$I \triangleq$ maximum number of NCP's plus ICP's in node.

$J \triangleq 5$

The array description is shown in Figure 6.12. The contents of array SWITCH are in fact equivalent to the X and Y axes of the cycle diagram. The addresses (source and destination) refer to those of the processors in the node. The contents of this array state that the packet give. by the identifier will be transferred from the processor identified by the source address to that of the processor identified by the destination address, in the following period.

ARRAY SWITCH (I,J)	
contents	
$I \triangleq$	maximum number of locations
J = 1	SOURCE PROCESSOR ADDRESS
J = 2	PROCESSOR TYPE (ICP,NCP)
J = 3	DESTINATION PROCESSOR ADDRESS
J = 4	PROCESSOR TYPE (ICP,NCP)
J = 5	PACKET IDENTIFIER

Figure 6.12 Switch Workspace

It is necessary to clarify some of the terminology used. The term c|XXX| denotes 'contents of array XXX'. Since this algorithm is based on the ordering of the Q array contents, the QLIST array must be referenced by using the pointer in Q, this is indicated by

$$c|QLIST(Q(j):pointer)\ yyy|$$

where yyy is the descriptor of the contents in QLIST to be processed, and j is the subscript in Q for the queue position of the packet identified by the pointer to QLIST. Queueing is denoted by

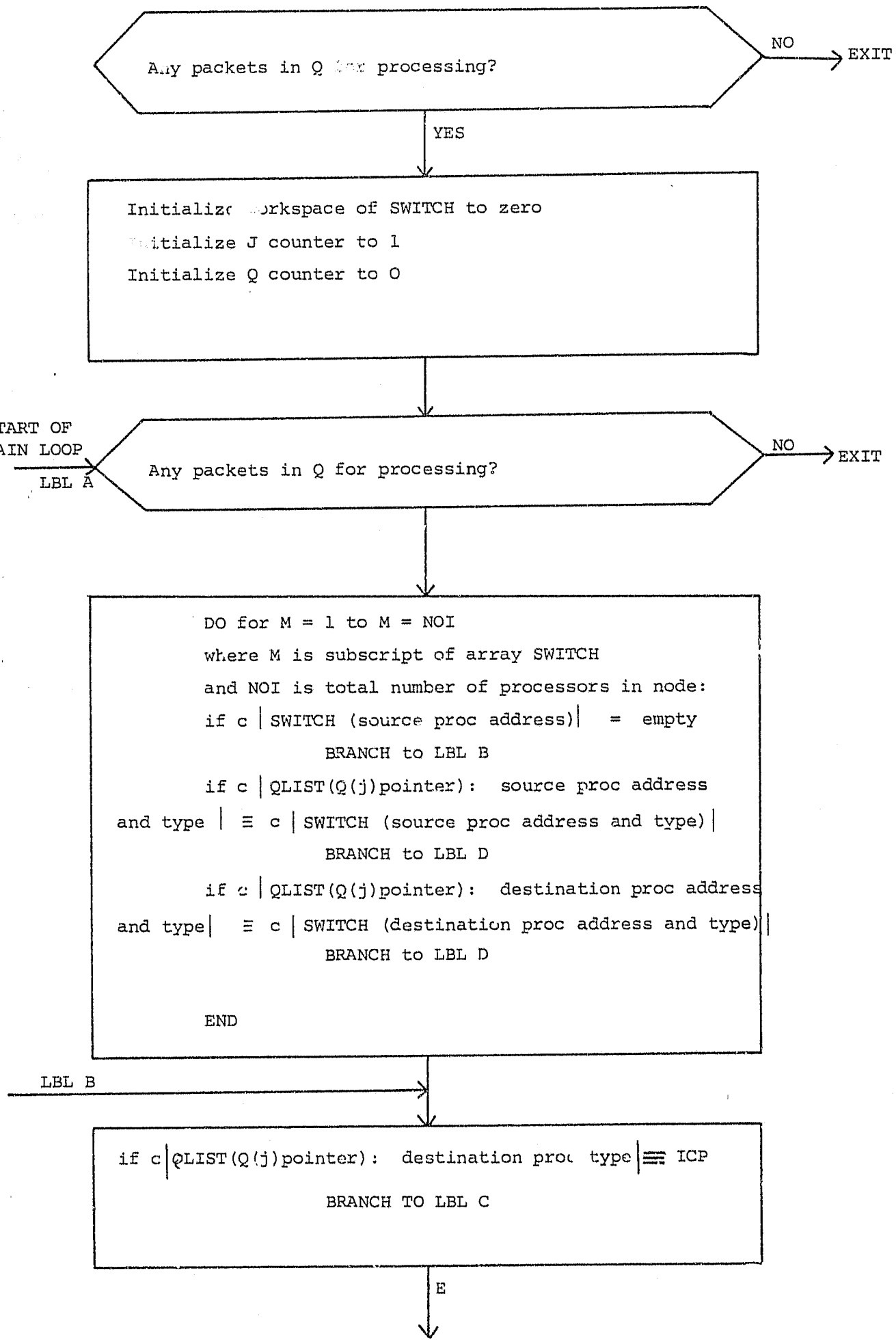
$$Q(j,i) \longrightarrow Q(j)$$

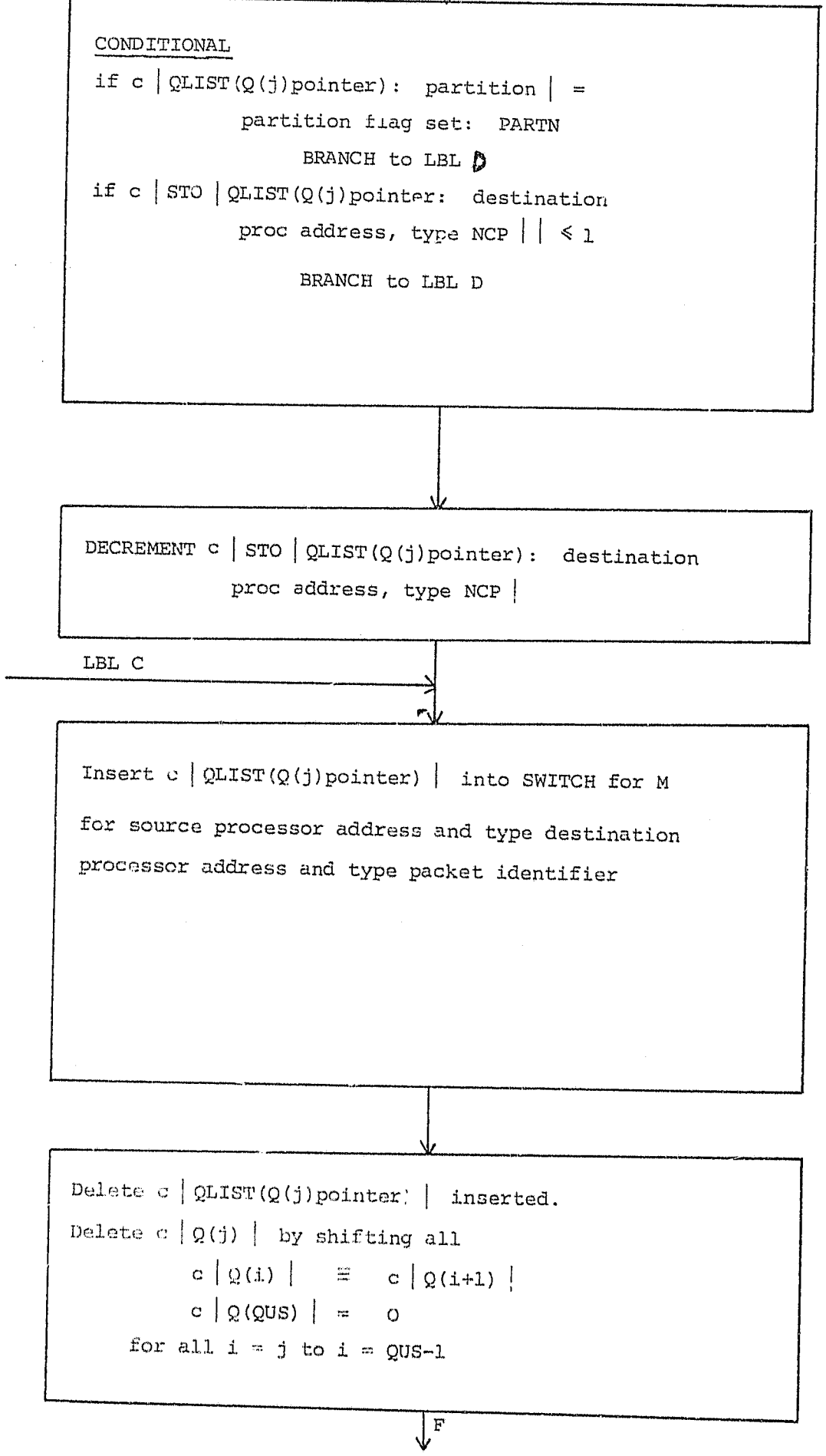
where j indicates the queue position, i = 1 the pointer value, or i = 2 the priority value.

6.3.2 Switch Algorithm

Once the polling phase has been completed, the switch is ready to generate a cycle diagram. Initially it is determined whether there are are in fact any packets to be transmitted; if none, the execution terminates (EXIT) and the switch waits for a one period duration before polling. If packets are to be transmitted the contents of the workspace SWITCH are initialized to zero, the queue position indicator j of Q(j) is set to one, and the counter QCOUNTER is set to zero.

The execution then enters the main loop, terminated by 'BRANCH TO LBL A'. One of the loop exit conditions occurs when no more data requires to be processed. The algorithm is shown in Figure 6.13.


Any packets in Q for processing?
NO
EXIT
YES
Initialize workspace of SWITCH to zero
Initialize J counter to 1
Initialize Q counter to 0
START OF MAIN LOOP
LBL A
Any packets in Q for processing?
NO
EXIT
DO for M = 1 to M = NOI
where M is subscript of array SWITCH
and NOI is total number of processors in node:
if c | SWITCH (source proc address) | = empty
BRANCH to LBL B
if c | QLIST(Q(j)pointer): source proc address and type | ≡ c | SWITCH (source proc address and type) |
BRANCH to LBL D
if c | QLIST(Q(j)pointer): destination proc address and type | ≡ c | SWITCH (destination proc address and type) |
BRANCH to LBL D
END
LBL B
if c | QLIST(Q(j)pointer): destination proc type | ≡ ICP
BRANCH TO LBL C
E


E
CONDITIONAL
if c | QLIST(Q(j)pointer): partition | =
partition flag set: PARTN
BRANCH to LBL D
if c | STO | QLIST(Q(j)pointer: destination
proc address, type NCP | | ≤ 1
BRANCH to LBL D
DECREMENT c | STO | QLIST(Q(j)pointer): destination
proc address, type NCP |
LBL C
Insert c | QLIST(Q(j)pointer) | into SWITCH for M
for source processor address and type destination
processor address and type packet identifier
Delete c | QLIST(Q(j)pointer) | inserted.
Delete c | Q(j) | by shifting all
c | Q(i) | ≡ c | Q(i+1) |
c | Q(QUS) | = 0
for all i = j to i = QUS-1
F

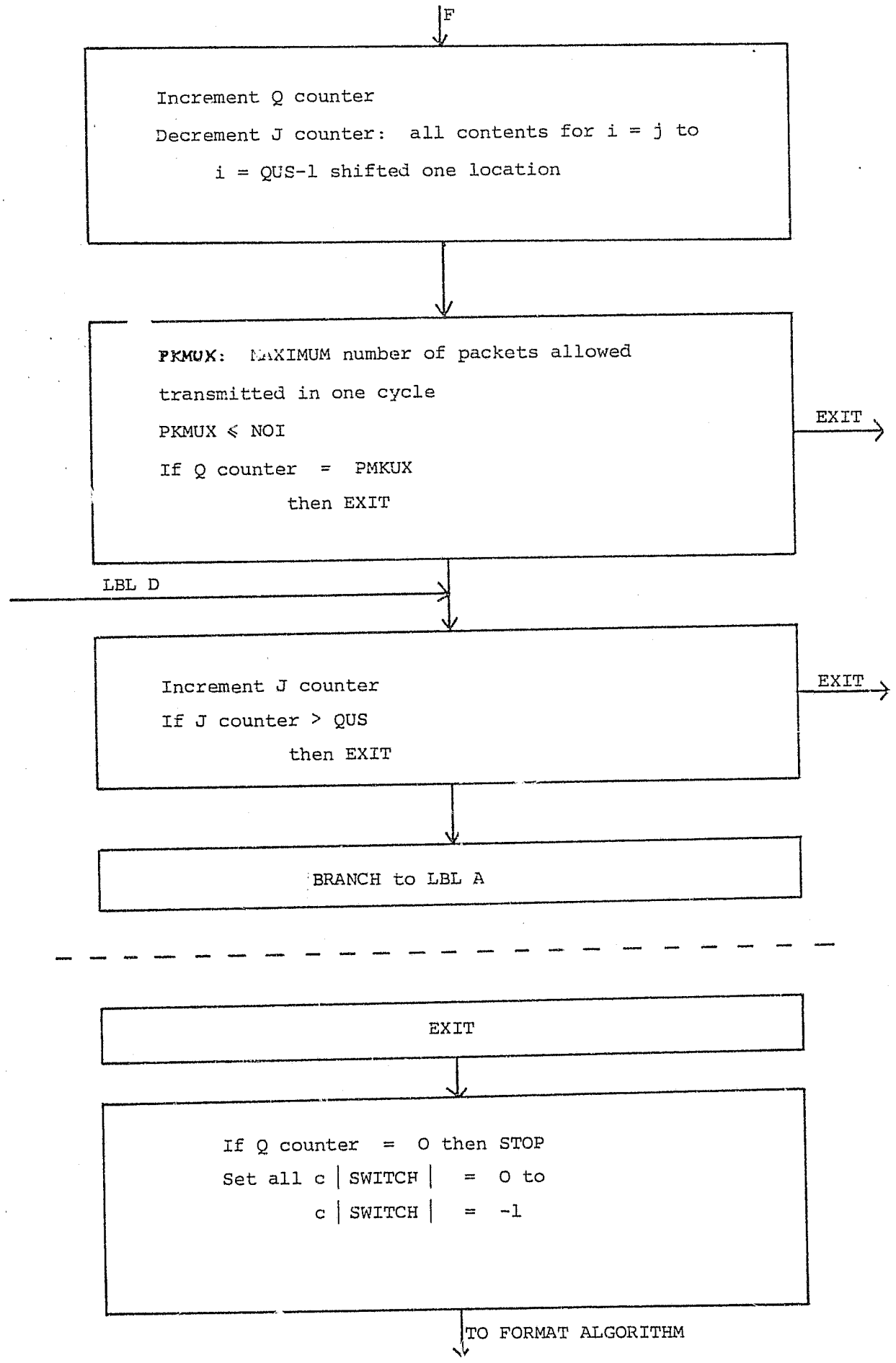


Figure 6.13(a) Switching Algorithm

The following holds for any packet in position j of Q. The first phase is concerned with satisfying the cycle diagram criteria: the source processor address for the packet in position j of Q, as given by QLIST, may not be the same as that of any of the source processor addresses currently stored in SWITCH. The same argument holds for destination processor addresses. If an empty location is found in SWITCH, that location is reserved until further conditions can be satisfied. In the event of these being satisfied the packet is allocated the location. Packets of higher priority will therefore be assured of a better chance of transmission success than those of lower priority. More important, since priorities are incremented with time, no packet can be delayed indefinitely. Finally, this subloop can never overflow since NOI is equivalent to the total number of ICP's and NCP's in the node.

The conditions mentioned above only apply to NCP's. So that congestion in the switching section of the network will not occur, packets are always transferred to ICP's irrespective of whether storage is available in the ICP. Temporary storage of such a blocked packet occurs in the buffer interface unit, enabling the ICP to issue a negative acknowledgement to the source ICP, if necessary. The conditions for the NCP to be met are:

(a) the relevant partition flag must not be set,

(b) buffering must be available in the NCP to accept the packet.

If the conditions are not able to be satisfied, the SWITCH locations

are relinquished, and it is attempted to transmit the packet in the next position in array Q. Otherwise the variable indicating store contents of the destination NCP concerned is decremented.

The data describing identifier, source, and destination processor addresses for a packet is inserted into the SWITCH locations, the corresponding QLIST contents are deleted, and the queue in array Q is shifted up one position. The J counter for queue position must be decremented since all data from the j'th position has been shifted once. At the end of the main loop the J counter is incremented, so that the next packet in order of priority can be processed, irrespective of whether the queue from position j has been shifted or not. If the j counter exceeds QUS, execution exits since end-of-queue has been reached. The QCOUNTER is subsequently incremented to indicate the total number of packets to be transmitted in the cycle, thus far.

The final exit condition occurs when QCOUNTER is compared with PKMUX, where PKMUX is a constant denoting the maximum number of packets allowed to be transmitted over the data bus in one cycle. PKMUX is in general dictated by bus bandwidth considerations.

Once the exit label is reached execution ceases if QCOUNTER is zero, the switch reverting to the polling stage some time later. Such an exit may arise if NCP's cannot meet the above conditions for accepting packets. All unused SWITCH locations are set to (-1) to

```
Do for j  =  1 to j  =  US

If c | Q(j) |  empty then EXIT

c | Q(j) |  =  c | Q(j) |  + c | QLIST(Q(j)pointer):
                                weighting factor |

          Set M to j

          Set N to j

LBL A     Decrement M

     if M  =  0 BRANCH to LBL B

     if c | Q(N) |  ≤  c | Q(M) |  BRANCH to LBL B

     EXCHANGE c  Q(N) |  with c | Q(M) |

     Adjust pointers in QLIST to new Q(N), Q(M)
                                locations

     Decrement N

     BRANCH to LBL A

LBL B   CONTINUE DO LOOP

     END
```

Figure 6.13(b) Priority Incrementation

distinguish between valid and invalid processor addresses. Finally, the switch algorithm may be executed several times in succession, generating a cycle diagram for every loop, if more than one cycle per period is required.

Figure 6.13(b) shows the manner of incrementing priorities; this code is not included in the main loop. It is executed every period. The priority of a packet is incremented by the weighting factor assigned in QLIST; the packet may jump ahead of other packets in the queue if its priority becomes greater than that of its neighbours.

6.9 Description of the Format Algorithm

Once the cycle diagram has been generated it is necessary for the switch to transfer the diagram results to the individual interface and network processors in the form of commands. These commands dictate when and which packet each processor is to transmit via the data bus.

It is remembered that control bus operations occur asynchronously to those on the data bus. This asynchronism allows packet transfers over the data bus to occur virtually continuously. Although previously events were described in terms of cycle terms, implementation-wise it is not very practicable; packets have varying lengths and the algorithm duration varies according to the amount of data to be processed. Synchronization flags have been used to control the transfer of commands from the switch to the ICP's and NCP's in a time-independent manner.

The format of the command is shown in Figure 6.14. It consists of four parts or fields. The packet to be transmitted is that defined by the packet identifier, and it must be sent to the processor listed in the field 'DEST PROC i'.

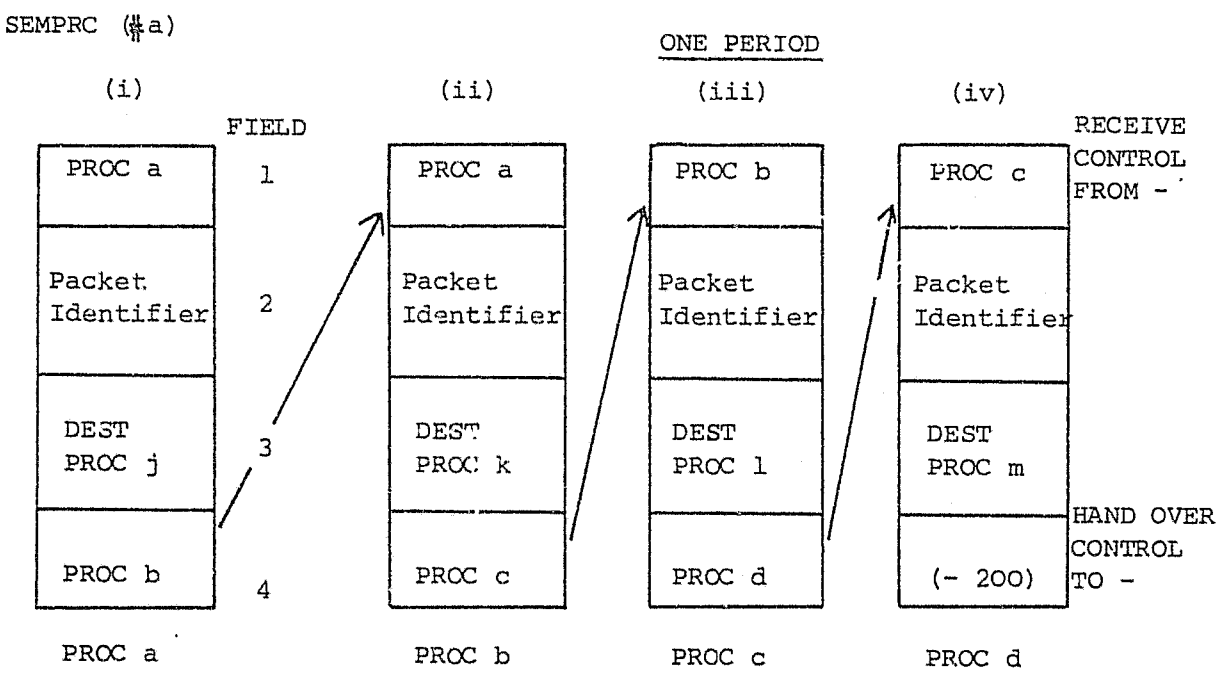


Figure 6.14 Formats Stored in Processors

6.9.1 Data Bus Access Control

The bus access technique used is essentially a form of hub polling, wherein control of the bus is passed from one module to another without recourse to a bus arbitration unit or a supervisor. The difference is that no polling of neighbouring modules occurs, but control is handed over directly to the processor specified in the fourth field of the command (see Figure 6.14). All transmitting processors are chained, bus access occurring only when needed. The transmitter contacts the

destination processor, sends its packet, and after an acknowledgement is received, hands bus control over to the next processor in the list.

The first field of the command is the module address from which a processor is to receive bus control, the fourth field dictates to which module it is to hand over control. The switching algorithm tries to ensure, where possible, that the processor to which the packet is to be sent, and the processor to which bus control is to be handed, coincide. This will reduce overhead in bus transfers.

It is necessary that the switch maintains some form of bus time-out. A faulty NCP or ICP in the chain may result in deadlock if bus control is not passed on to the next module. A diagnostic routine should then be initiated by the switch, to isolate the faulty module, and to re-initiate bus transfers. This level has not, however, been dealt with in the simulation.

The first field of the first processor in the chain to transmit will coincide with the processor's own address. This processor receives a signal from the switch to start the transmission procedure. Used is the signal SEMPRC (#a) where #a is the processor address. The last processor in the chain contains the code -200 (as used in the simulation) to signify that control is to be passed back to the switch.

6.9.2 Synchronization of Data and Switch Events

The synchronization of command transfers is shown in Figure 6.15.

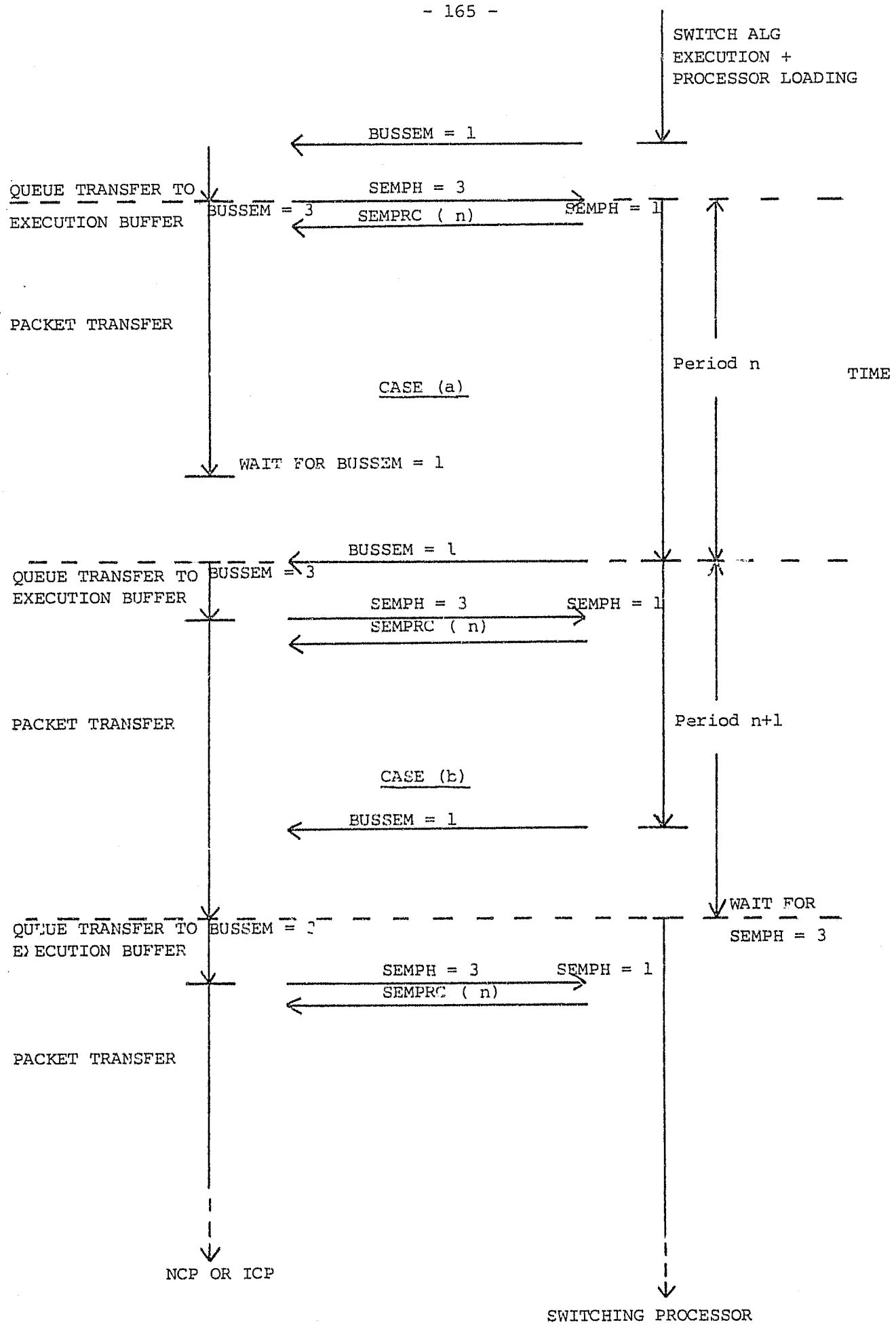


Figure 6.15 Packet and Command Transfer Synchronization

Synchronization is necessary to prevent the switching processor from overwriting a command that has been stored in the ICP or NCP, but not yet been executed.

Taking into account variable packet lengths, switch algorithm execution times, the term period may be redefined as the time for the initiation, execution, and termination of a chain of bus transfers; this definition is basically equivalent to the previously given one.

Basic nodal functioning may be considered to consist of two parts:

i) the sampling, processing and command output to the ICP's and NCP's by the switch,

ii) the transfer of packets over the data bus as controlled by the commands in each NCP and ICP.

These two functions may occur concurrently, i.e., whilst the switch is processing status data during period m for use in period m + 1, data bus transfers take place according to the commands produced by the switch in the previous period m - 1.

Buffers for storing commands consist of queueing and execution sections. The switch loads the command into the queueing buffer, it being subsequently transferred to the execution buffer by the ICP/NCP after the packet of the previous command has been sent. The command in the execution buffer controls the ICP/NCP packet transfer. Since the time of bus access by the ICP/NCP, and the time of command loading by the switch, cannot be determined in advance, double buffering serves

as a simple means of preventing overwriting of a command that has not yet been executed. In practice, simple interchange of a pair of pointers to indicate which is currently the execution and which the queueing buffer, is needed.

Synchronization is achieved as follows. The last processor in the chain must wait for a signal 'BUSSEM=1' from the switch, indicating that the algorithm execution and command loading has been terminated. The processor transfers the contents of the queue buffer to that of the execution buffer. This transfer phase will have been already completed in all other processors. Once accomplished, the signal 'SEMPH=3' is sent from processor to switch. Note that the above signal is only sent once the processor has transmitted its packet. The switch responds by issuing the signal SEMPRC (#m) to processor m initiating the next chain of bus transfers. While data bus transfers are proceeding, the switch samples and processes nodal status data. This operation is repeated successively.

If no packets are to be transmitted, no commands are sent, the sampling occurring every period as before, however.

Two situations, (a) and (b), in Figure 6.15 are discussed. If the data bus transfer period terminates before the switch has completed processing, as in (a), the transfer phase is forced to wait for the switch, i.e., the last processor in the chain must wait for the BUSSEM signal. In case (b), the BUSSEM signal has already been sent to the last processor before the transfer phase has been completed. The last

processor to transmit in the period will issue the SEMPH signal once it has sent its packet and transferred the contents of the queue buffer to the execution store. Thus, irrespective of which phase terminates first, both are synchronized and proceed as soon as the switch, after receiving SEMPH, has sent the SEMPRC (#n) signal to the first processor allowed bus access for the subsequent period.

6.9.3 Format Algorithm

The translation of switch algorithm results to command format is examined. Figure 6.16 illustrates the translation required between the SWITCH and PTICP/PTNCP array. The latter arrays are used in the simulation to store the commands destined for NCP's and ICP's. The packet identifier and destination processor address locations of array SWITCH are translated directly to fields 2 and 3 in PTICP/PTNCP. The source processor address in SWITCH is the address to which the command is transmitted. The format algorithm must determine the contents of fields 1 and 4 in arrays PTICP/PTNCP. Recall that field 1 specifies the processor from which a reference processor receives bus control, field 4 specifies which processor bus control is to be handed to.

The first part of the algorithm concerns the alignment of source and destination processor addresses (Figure 6.17). For such to occur, the destination processor address contents in array SWITCH (I, J) at location I = i, must be identical to the source processor address location contents of SWITCH at I = i + 1.

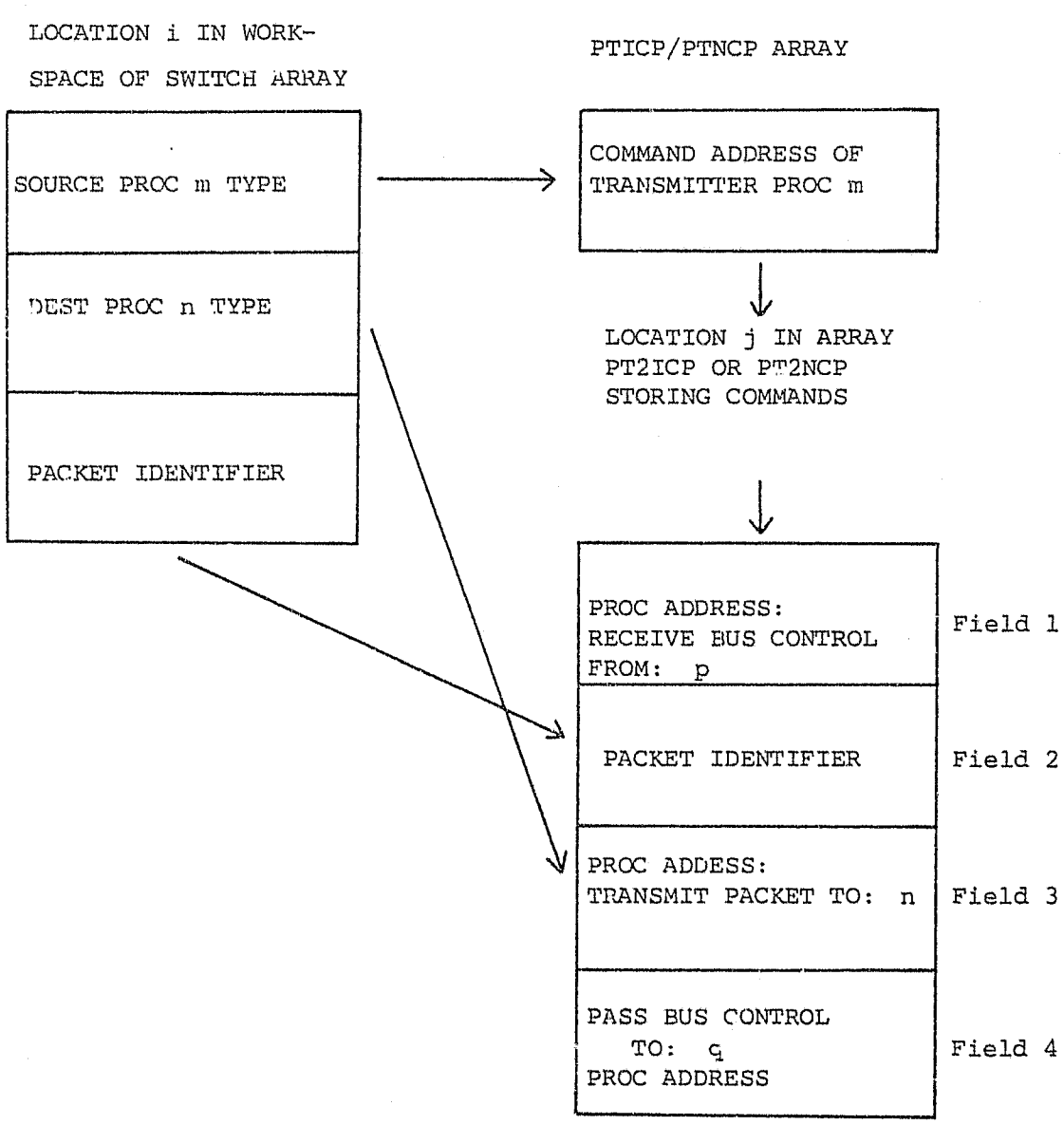


Figure 6.16 Translation of Switch Contents to Commands

The algorithm shown in Figure 6.18 aligns the two addresses where possible. For each index i in SWITCH (I,J), the contents of all other locations from I = i + 1 to NOI are examined, where NOI is the number of ICP's plus NCP's in the node. When it is found that a pair of source and destination addresses are identical, for example at I = j,

ARRAY SWITCH (m,5)

1	SOURCE PROC	SOURCE PROC	SOURCE PROC	SOURCE PROC
2	ADDRESS i	ADDRESS j	ADDRESS k	ADDRESS n
3	DEST PROC	DEST PROC	DEST PROC	DEST PROC
4	ADDRESS j	ADDRESS k	ADDRESS l	ADDRESS p
5	PACKET ID	PACKET ID	PACKET ID	PACKET ID
	m=1	m=2	m=3	m=4

Figure 6.17 Alignment of Source and Destination Processor Addresses in Workspace Switch

NOI : Number of workspace locations in SWITCH(I,J) ≡ total number of ICP's and NCP's in node

```
        DO for JV = 1 to JV = (NOI - 1):
        if c | SWITCH(JV+1,SPA) |  =  -1 EXIT LOOP
        KW  =  JV+1
        DO for JW = KW to JW = NOI:
        if c | SWITCH(JV,SPA) |  =  c | SWITCH(JW,DPA) |
                        BRANCH TO LBL A
        END
        BRANCH TO LBL B

LBL A       interchange c | SWITCH(JV+1, j = 1 to 5) | with
                        c | SWITCH(JW, j = 1 to 5) |

LBL B   CONTINUE
        END
        EXIT
```

Figure 6.18 Alignment Algorithm

SPA: SOURCE PROC ADDRESS

DPA: DESTINATION PROC ADDRESS

the contents of the SWITCH (I,J) at I = i + 1 are interchanged with those at I = j.

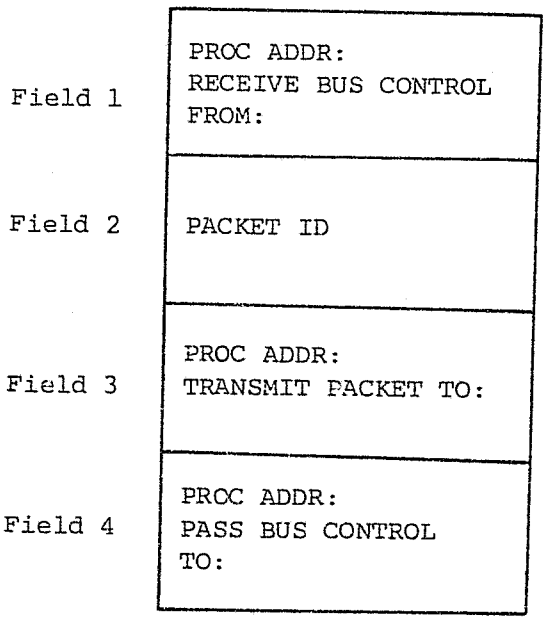
The second part of the algorithm formats the commands and is shown in Figure 6.19.

For each non-empty location I = 1 to NOI in array SWITCH (I,J) a command is formatted to be sent to the source processor identified by address J = 1 and J = 2. The loop is exited when all locations in SWITCH have been processed - the contents set to -1 indicate no packet transmission.

The variable SEMPRC is set to the address of the processor initiating bus transfers in the first cycle at the time the first command is generated.

The contents of field 1 of the command are set equal to the address of the processor itself - for the first processor in the chain to transmit. The variable VARIABLE 1 is set to the contents of the source processor address location in array SWITCH. It is used to contain the source address of the previous command for use in field 1 of the current command generated. The contents of field 1 ('receive bus control from -') are thus able to be specified by noting which processor was the last to be allowed bus access.

For simulation purposes, if the transmitter is an interface



SPA: Source processor address

DPA: Destination processor address

NOCYC: Number of current cycle in period generated

NOI: Number of workspace locations in array SWITCH

SEMPRC: Contents denote address of first processor in chain to transmit

```
DO for M = 1 to M = NOI:
     if c | SWITCH (M,SPA) | = -1      EXIT LOOP
counter = counter + 1 (initialized at sorting algorithm stage)
     if counter = 1 and NOCYC = 1
     set c | SEMPRC | to c | SWITCH (M,SPA) |
     set c | VARIABLE1 | to c | SWITCH (M,SPA) |

     if processor is ICP:
     set c | PTICP (trm proc addr, field 1) | to c | VARIABLE 1 |
     set c | PTICP (trm proc addr, field 2) | to c | SWITCH (M,PCK ADDR) |
     set c | PTICP (trm proc addr, field 3) | to c | SWITCH (M, DPH) |
BRANCH to LBL A
```

```
     if processor is NCP:
     set c | PTNCP (trm proc addr, field 1) | to c | VARIABLE1 |
     set c | PTNCP (trm proc addr, field 2) | to c | SWITCH (M,PCK ADDR) |
     set c | PTNCP (trm proc addr, field 3) | to c | SWITCH (M,DPA) |

LBL A  if counter = 1 and NOCYC = 1
BRANCH TO LBL C

|    if counter = 1
|    set c | PTICP (VARIABLE1, field 4) | of previous cycle
|         to c | SWITCH (M,SPA) | if ICP
|    set c | PTNCP (VARIABLE1, field 4) | of previous cycle
|         to c | SWITCH (M,SPA) | if NCP
|BRANCH TO LBL B

     set c | PTICP (VARIABLE1, field 4) | present cycle
          to c | SWITCH (M,SPA) | if ICP
     set c | PTNCP (VARIABLE1, field 4) | present cycle
          to c | SWITCH (M,SPA) | if NCP
LBL B c | VARIABLE1 | = c | SWITCH (M,SPA) |
LBL C
END
     if not FINAL CYCLE IN PERIOD RESTART SWITCH AND
          FORMAL ALGORITHMS AGAIN
END-OF-PERICD:  if ICP set c | trm proc addr, field 4) | = -200
                if NCP set c | trm proc addr, field 4) | = -200
```

Figure 6.19 Format Algorithm

processor the command is stored in array PTICP, if a network processor in array PTNCP.

The address specified in the fourth field of the command ('pass bus control to -') can only be known once the command for the next processor has been generated. This is achieved by setting the contents of field 4 for array PTICP/PTNCP as indexed by the content of VARIABL1, to the contents of the source processor address location in array SWITCH currently being processed. The arrays PTICP/PTNCP, indexed by VARIABLE1, will refer to the command most recently generated, excluding the current.

The variable VARIABLE1 containing the source address of the previously generated command, serves two purposes: its content is used to set field 1 of the current command; at the same time it is used as an index specifying where the previously generated command may be found in arrays PTICP/PTNCP, i.e., the command to be sent to the processor that is to hand over bus control to the processor currently of interest.

6.10 Preview

A functional outline of the communications processor has been given in this chapter. A technique has been evolved whereby a switching processor supervises the operation of the other modules in the node. The three primary functions and their algorithmic implementation, sort, switch and format, of the switching processor were dealt with.

It was shown that this technique provides a means for resolving bus access problems and for controlling data flow within the node, and between neighbouring nodes and the node itself.

CHAPTER 7

NETWORK PERFORMANCE

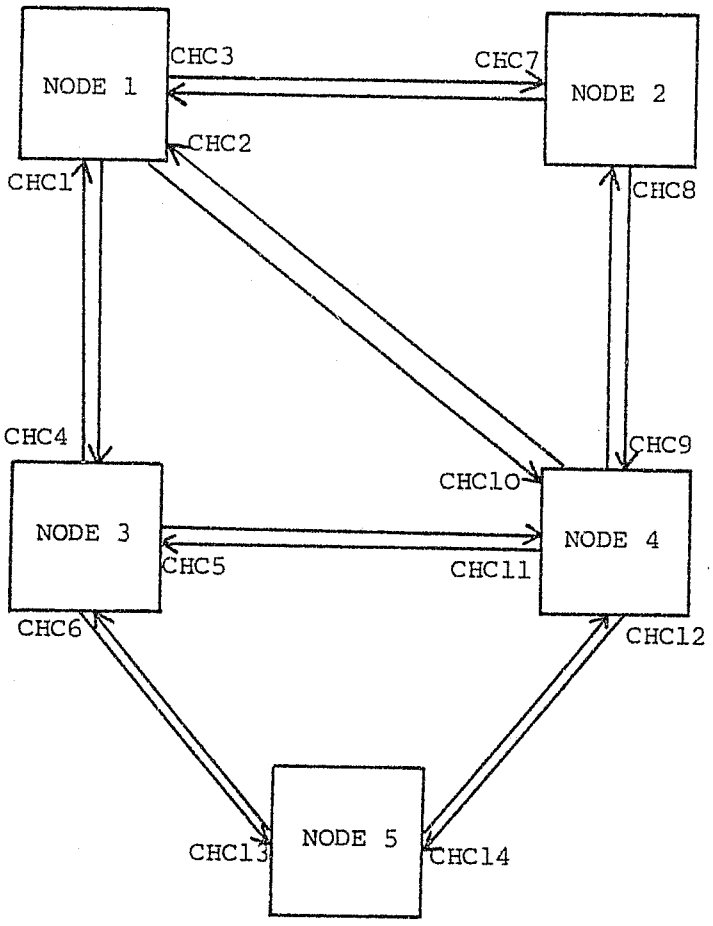
7.1 Introduction

Nodal operation, by virtue of its architectural features and protocols, must be tested as part of a network. No centralized facility exists in the network for monitoring and controlling data transmission. The implementation of the protocols and flow control techniques discussed in the preceding chapters is an attempt to enable a number of connected nodes to function as a system for packet transmission. Two network topologies have therefore been simulated to test the above idea and to determine the performance and characteristics of the proposed communication unit.

7.2 Discussion on the Topologies Simulated

The two network topologies simulated are illustrated in Figure 7.1, and will be referred to as the 'network simulation' and the 'nodal simulation'. The network simulation has been used to observe the characteristics of a decentralized system in which no master node exists to direct data flow in the network. The nodal simulation focusses attention on the behaviour of a nodal packet switch. The remaining nodes used in the nodal simulation are necessary to ensure implementation of complete protocols and flow control measures.

It is not particularly efficient to simulate a large network consisting of many nodes. Very long simulation runs would be



CHC: CHANNEL CONTROLLER

Figure 7.1(a) Network Simulation

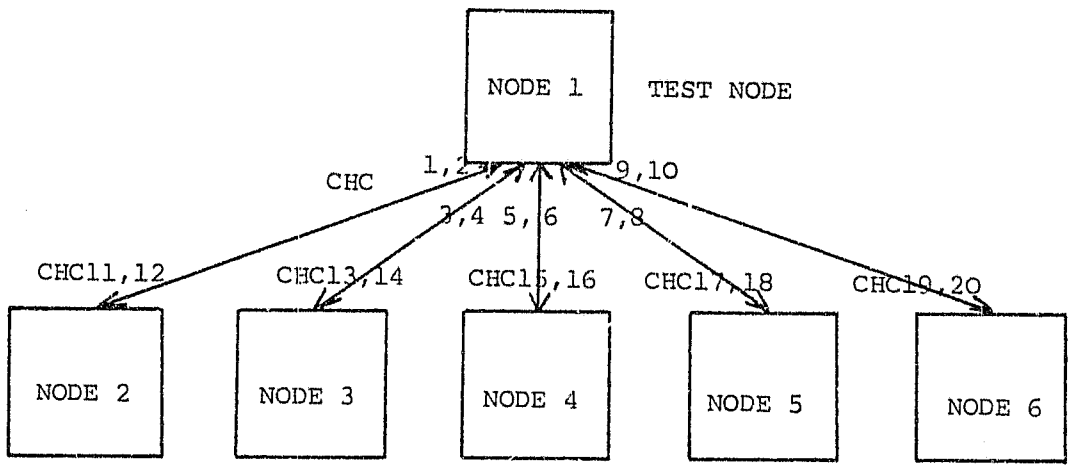


Figure 7.1(b) Nodal Simulation

needed before the model queueing structures arrived at a state of equilibria. The model shown in Figure 7.1(a) may be considered to form a subset of such a large network, forming an approximation to a hierarchically-based network consisting of nodal clusters. It is noted that the network simulation is an autonomous unit, with its own set of logical links.

7.2.1 Network Parameters

The network simulation consists of five nodes connected as in Figure 7.1(a). Each node has a connectivity of at least two. Fourteen channels are used to link the nodes; two network processors, one on either side of a channel, control internodal packet transfers. The capacity of the channels is 64 000 bits per second. Eight interface processors per node simulate the connection of hosts logical links.

The nodal simulation (Figure 7.1(b)) consists of a total of six nodes, twenty network and forty interface processors. The test node contains fifteen interface and ten network processors; each of the remaining five nodes, five interface and two network processors. A double link connects each of the five nodes to the test node. The channel capacity between two nodes is 128 000 bits per second. Each channel has its own pair of network processors controlling the link.

The route map for packets is fixed. In the nodal simulation a choice exists as to which of the two channels connecting node and

test node is to be used. The choice is decided by allocating the packet to the network processor with the shortest queue. If queues in both channel processors are equally long, the first on the route list transmits the packet.

Fifty logical links and fifty virtual circuits have been used for both simulation models.

Concerning the network simulation, each of the five nodes acts as source to ten logical links. Two of the ten links are sourced and sunk at the same node (also referred to as incest traffic). The remaining eight links are sunk in each of different nodes, two per node. This provides a reasonably equalized traffic distribution in terms of the number of links handled per node, but will cause differences in line and bus utilization as the topology is asymmetric.

Concerning the nodal simulation, the test node acts as source to fifteen links, three links terminating at each of the five remaining nodes. Nodes 2, 3 and 4 source five links each, one link sunk in each of the remaining five nodes; nodes 5 and 6 source ten links each, two links terminated at each of five remaining nodes. The traffic distribution is itself asymmetric.

For each logical link there exists a virtual circuit with identical source and sink node (remember that the virtual circuits are used to increase the traffic intensities in the network).

The maximum number of packets able to be multiplexed per cycle in each node of the network simulation is as follows:

node 1 and node 3 : eleven packets per cycle,

node 2 and node 5 : ten packets per cycle,

node 4 : twelve packets per cycle.

These figures correspond to those given for the number of interface and network processors in each node. Maximum throughput rates in each node can thus be attained. Similarly, for the nodal simulation:

node 1 (test node) : twenty-five packets per cycle,

nodes 2 - 6 : seven packets per cycle.

The following holds for both simulation models. The buffer block sizes used in the processors are equal to the maximum packet lengths and are 500 bits. Eight buffer blocks are allocated to logical links at the start of a message transmission, the maximum segment size being 4 000 bits. Each network processor is provided with sixteen buffer blocks per transmitter and per receiver section, plus two blocks per partition. Each interface processor contains enough buffers to ensure that no blocking at this level occurs - parameters to be tested concern the performance of the switching section of the network.

The message and packet lengths (for virtual circuits) and their interarrival times follow the Poisson distribution. The mean message length has been set at 400 bits. The message interarrival times will vary depending on the level of traffic injected, as holds

for packet lengths and generation rate. The overhead on packets has been set at 70 bits for internodal communication (added to the packet length). The error rates on channels has been assumed to follow a Poisson distribution, the mean error rate set at 10^5 bits per error.

For the switch processor in each node, one cycle per period has been used. This allows the switch to retain maximum control over nodal operation, as status data is sampled most frequently.

The input parameter listings for the network and nodal simulations are to be found in Appendices C and D respectively.

7.2.2 Assumptions made in the Simulation

The primary assumption made is that the time spent in processing data packets in the network and interface nodes is negligible compared to the time spent in queueing for resources.

The time taken for a packet to be transmitted across a channel may take from twenty to one hundred milliseconds, depending on the utilization of the link. The execution time of protocols by network processors, especially as many features may be implemented in hardware, is negligible compared to the queueing times of which the above figures are a reflection.

The processing of packets may take place concurrently in the

processors constituting the node. Whilst one packet is being transmitted across the bus by one module, the others can process packets until such time as they are allowed bus access. There is thus considerable overlap between the module activities and the packet transfers via the nodal bus. Again, it is assumed that use of the bus resource constitutes the dominant time factor.

In fact the simulation results to follow will show that the internodal transit times are dominant - of the order of tens of milliseconds.

7.2.3 Performance Indices Used in the Simulation

The performance indices used to characterize the behaviour of the network are the packet delay, the network throughput, and the bus and line utilization factors.

Three different delay variables have been made use of; definitions follow:

- the entry delay is defined to be the time taken for a packet to enter an interface processor to the time it leaves the processor, gaining entry to the network.
- the transit delay is defined to be the time taken for a packet to leave the interface processor, transit the network, and gain entry to the destination interface unit.
- the total delay is the sum of entry and transit delays.

Packet delay is the primary performance index used to define network behaviour. It is noted that the time taken for a message to traverse the line connecting host and network is not included; these delays depend on the capacities of the lines employed.

The traffic intensity in the network is measured via the line and nodal bus utilization factors. The utilization factor p is defined as the product of the mean packet arrival rate to the resource and the mean service time required. This quantity gives the fraction of time that the server is busy. It may also be defined as the ratio of the rate at which packets arrive at the resource to the capacity of the resource to service the packets.

Unlike the measurement of mean packet delay, which may be determined in a single simulation run, a mean value for packet throughput can only be found by repeating a number of simulation runs at different random number generator seeds for packet length and interarrival time. Such a mean throughput value has been found for certain simulation runs. A second figure, the 'recorded throughput' gives the number of packets transmitted for a single simulation run only. It is used primarily to obtain an idea of the total number of packets transmitted, averaged over the simulated time period that the model was run.

7.3 Performance of the Network

7.3.1 Notational Details

In order to observe the response and throughput it is necessary to control the rate of input of packets to the network. The logical links generate messages at the rate of one every second. This rate is kept constant for all simulations. Packet injection is done primarily by use of virtual circuits. Input rates range from 1,0 to 16,7 packets per second per circuit. The input rates displayed on the graphs are the total rates for fifty virtual circuits, thus a total input rate of 500 packets per second implies fifty circuits each transmitting at 10 packets per second.

It will be remembered that priority functions may be programmed into the switching algorithm executed in each node. Throughout most of the simulation tests to follow, tests will take place considering the following two cases. The priorities are time-dependent and are termed PRIORITY (ENTRY) and PRIORITY (TRANSIT). The former holds for packets input to the switching section of the network and for incest traffic, whilst the latter is valid for packets in transit at a node, or leaving the switching section.

7.3.2 Simulation Run-Time

Tests were conducted on the network simulation to determine the run time required for a five node topology model to reach a state of equilibrium. The basic unit employed is the cycle. Nodal bus rates were 640 KBPS, with a retransmission interval of

40 milliseconds. Further details are contained in Tables F.2 to E.12 of Appendix E.

The response has been plotted in Figures 7.2 and 7.3, throughput in Figures 7.4 and 7.5.

Consider the relation between the number of cycles and simulated time. The amount of time simulated at inputs of 50, 525, 625, 750 and 835 packets per second after 16 000 cycles is 46,0, 5,0, 4,5, 3,6, and 3,5 seconds respectively (Table E.2). The variation occurs since the simulation model is event driven. With fewer events taking place per unit time (as in the lower input levels), a longer time period may be simulated for a given number of cycles.

The response curves indicate that, above a certain input level, no equilibrium state can be reached. The curves for input rates up to 625 packets per second remain steady, varying by not more than a few milliseconds. The delays at the levels of 750 and 835 packets per second show no sign of settling down. That no steady state will be reached no matter the length of the simulation, may be inferred by considering the bus utilization factors (Tables E.3 to E.12). Consider the run with level 835 packets per second. A perusal of the line and bus factors shows that whilst the bus utilization remains relatively constant for each node, the line utilization increases with each increment of 2 000 cycles.

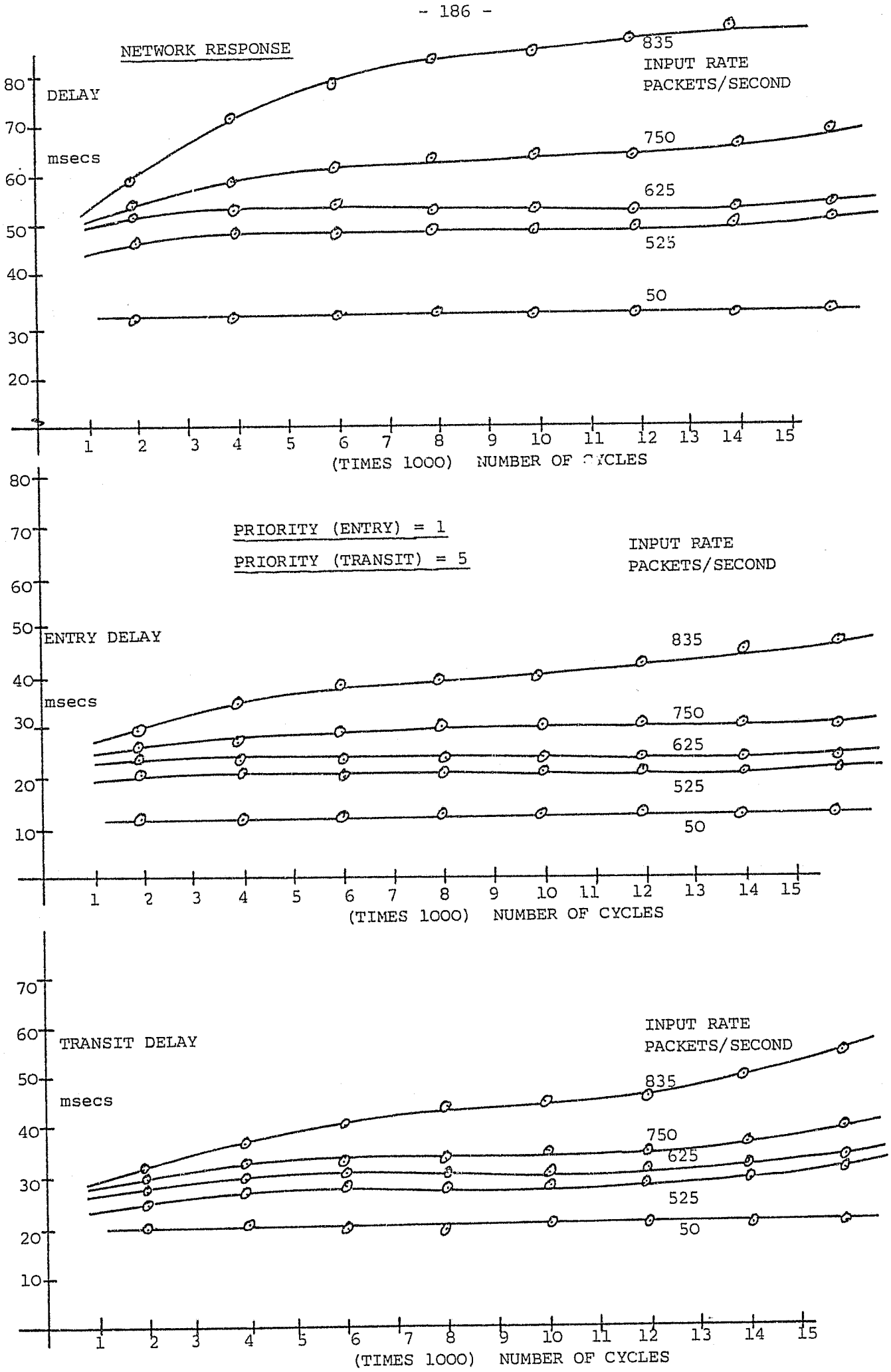

NETWORK RESPONSE
DELAY
msecs
80
70
60
50
40
30
20
835
INPUT RATE
PACKETS/SECOND
750
625
525
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
INPUT RATE
PACKETS/SECOND
ENTRY DELAY
msecs
80
70
60
50
40
30
20
10
835
750
625
525
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES
TRANSIT DELAY
msecs
70
60
50
40
30
20
10
INPUT RATE
PACKETS/SECOND
835
750
625
525
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES

Figure 7.2

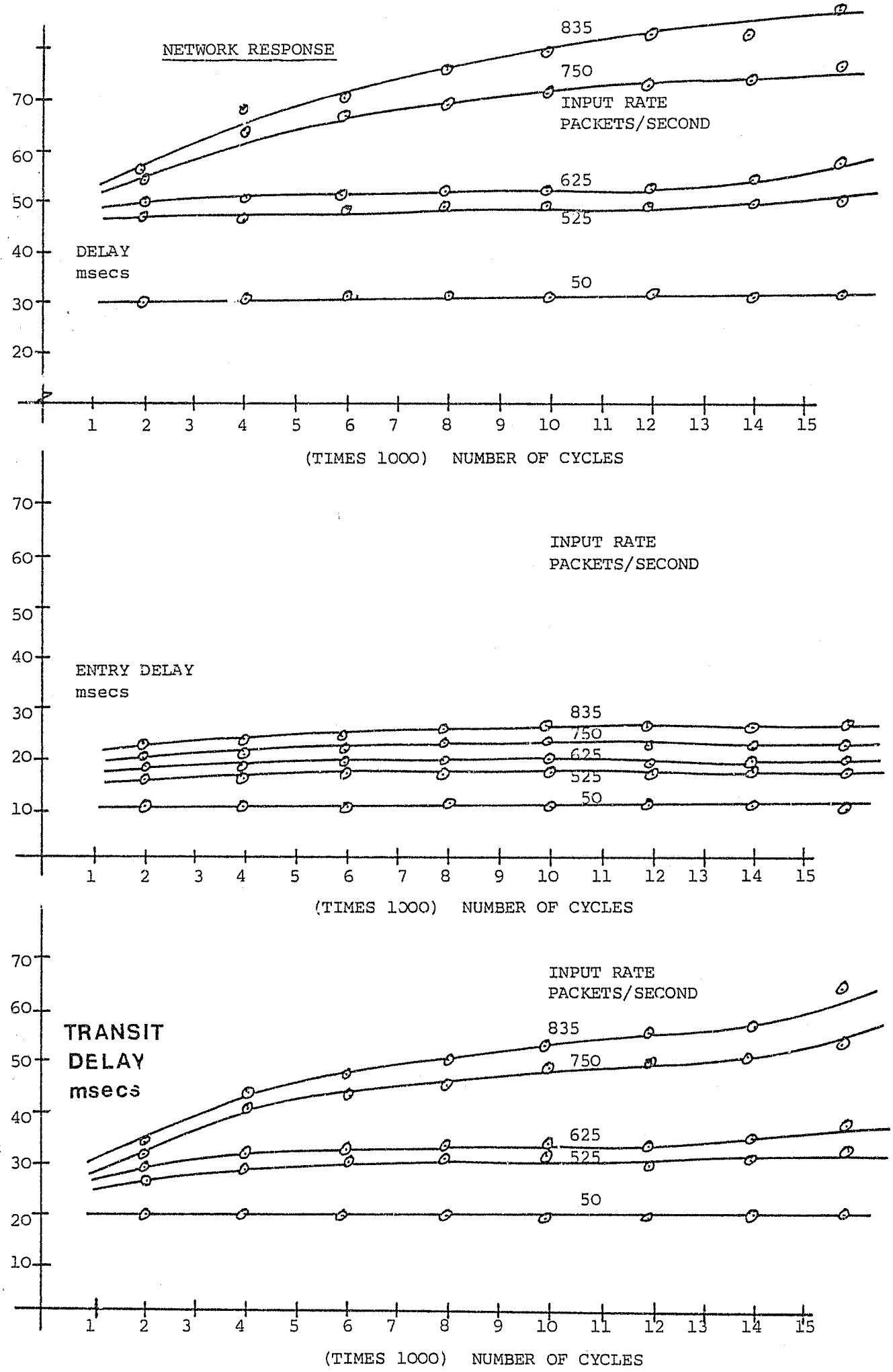
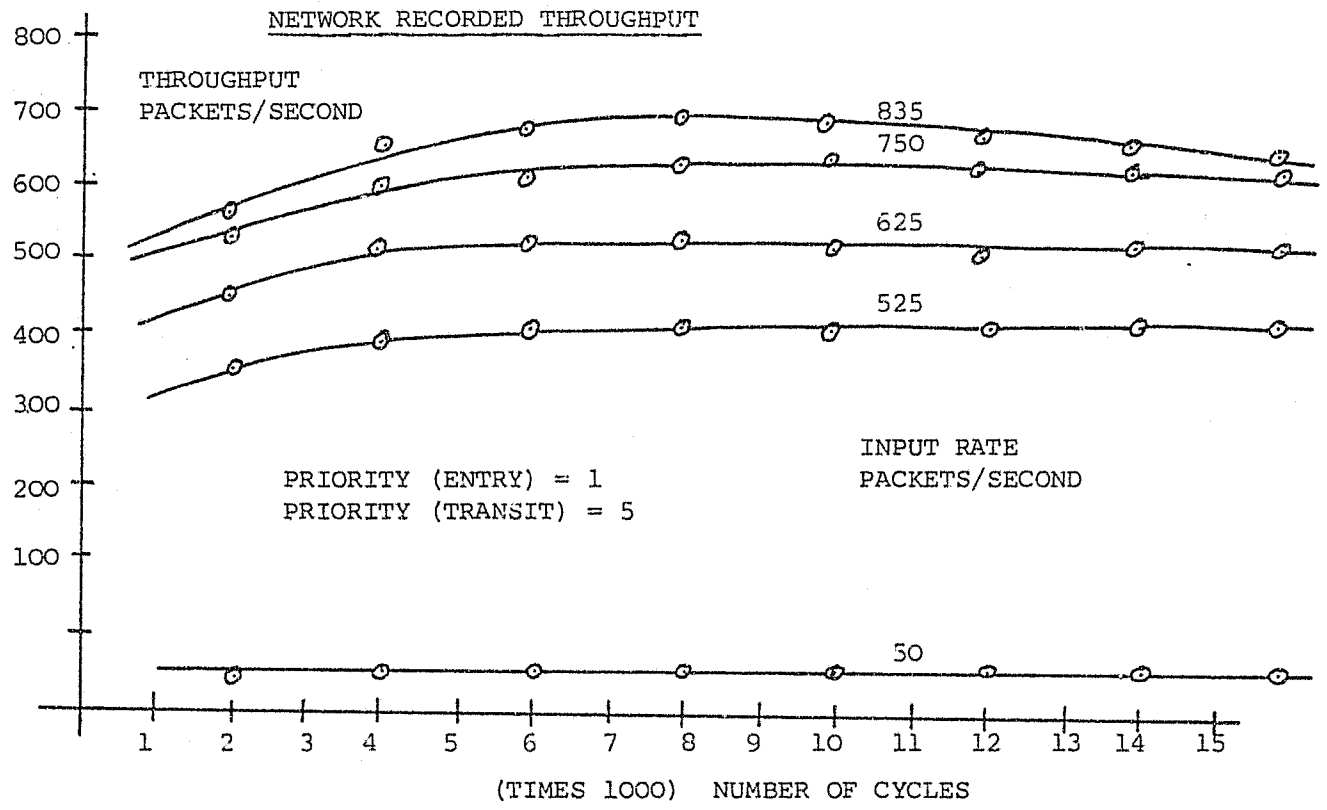

NETWORK RESPONSE
835
750
INPUT RATE
PACKETS/SECOND
625
525
50
DELAY
msecs
70
60
50
40
30
20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES
INPUT RATE
PACKETS/SECOND
ENTRY DELAY
msecs
70
60
50
40
30
20
10
835
750
625
525
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES
INPUT RATE
PACKETS/SECOND
TRANSIT
DELAY
msecs
70
60
50
40
30
20
10
835
750
625
525
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES

Figure 7.3


NETWORK RECORDED THROUGHPUT
THROUGHPUT
PACKETS/SECOND
800
700
600
500
400
300
200
100
835
750
625
525
50
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
INPUT RATE
PACKETS/SECOND
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES

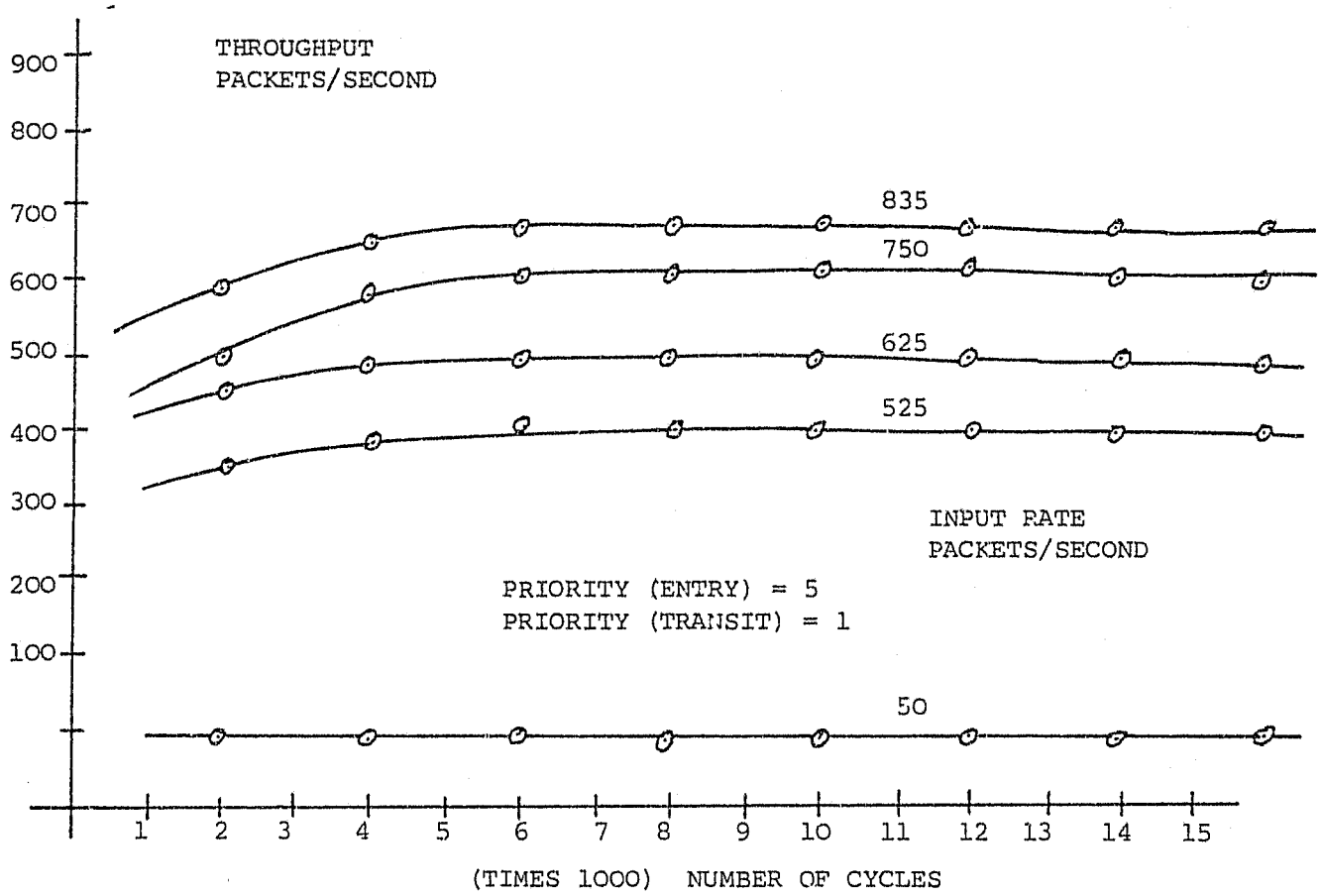

THROUGHPUT
PACKETS/SECOND
900
800
700
600
500
400
300
200
100
835
750
625
525
50
INPUT RATE
PACKETS/SECOND
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 1000) NUMBER OF CYCLES

Figure 7.4

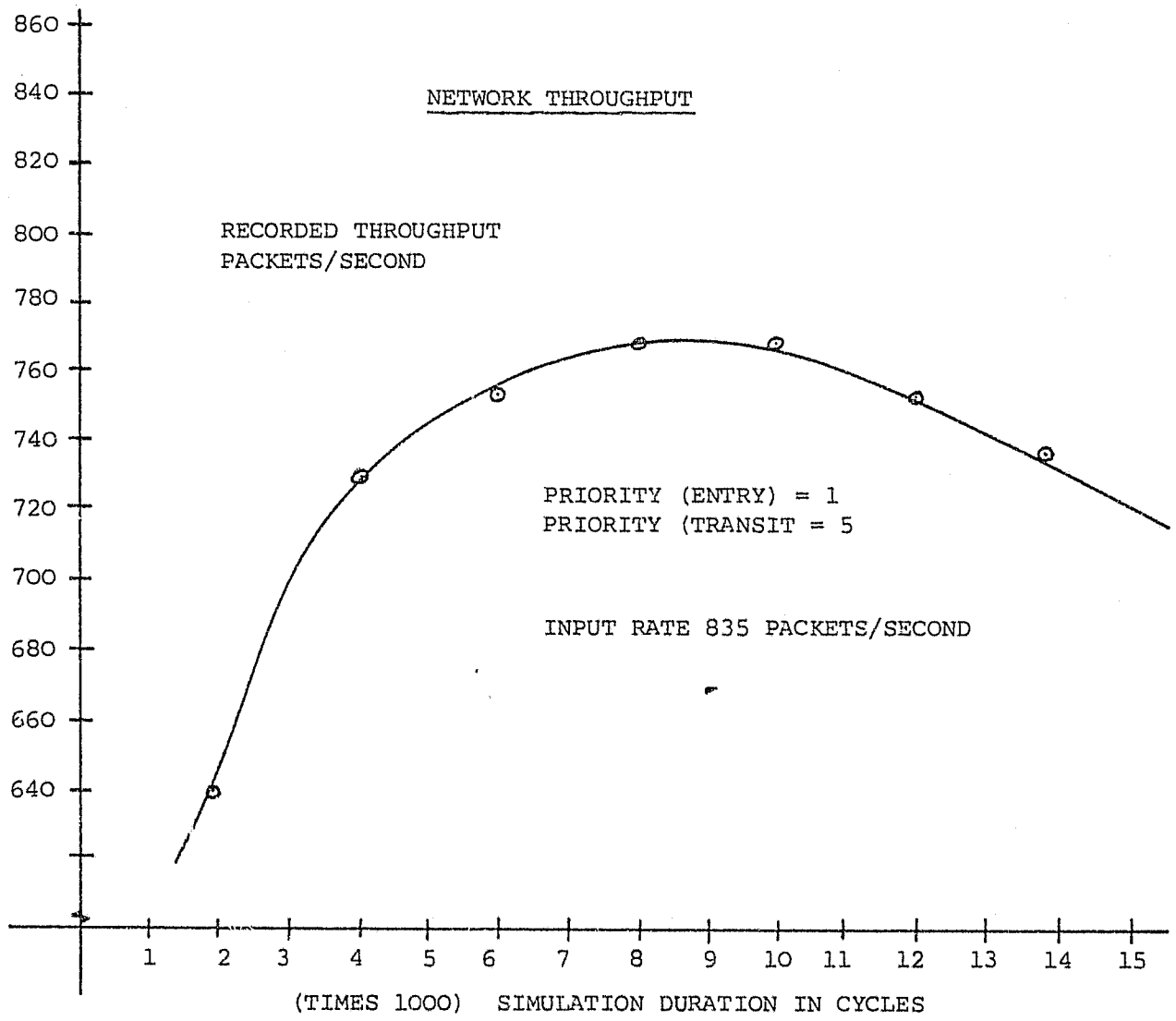

NETWORK THROUGHPUT
RECORDED THROUGHPUT
PACKETS/SECOND
860
840
820
800
780
760
740
720
700
680
660
640
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT = 5
INPUT RATE 835 PACKETS/SECOND
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
(TIMES 1000) SIMULATION DURATION IN CYCLES

Figure 7.5

This implies that the amount of data transferred by each node remains almost constant, but the channel activities continually increase. This takes place due to the destination channel buffers having become blocked, thereby increasing the packet retransmission rate. As demonstrated in Chapter 5, a packet network cannot function properly if line utilization factors tend to unity, due to the excessive delays encountered by packets.

The above is illustrated again with the throughput curves. The curve at input level 835 packets per second shows that the number of packets transmitted by the network peaks, and then decreases with time. At the lower levels the throughput remains constant.

There thus exists a certain level of input traffic to a network at which congestion takes place, resulting in decreased throughput and longer response times for packets. Such takes place due to certain of the network channels becoming congested.

The results produced indicate that simulations for the 'stable' input levels (625 packets per second or less) need not be run for 16 000 cycles or longer. A run of 6 000 cycles is sufficient.

Finally, mention is made of an alternative technique often used in modelling: the simulation is run until desired accuracies

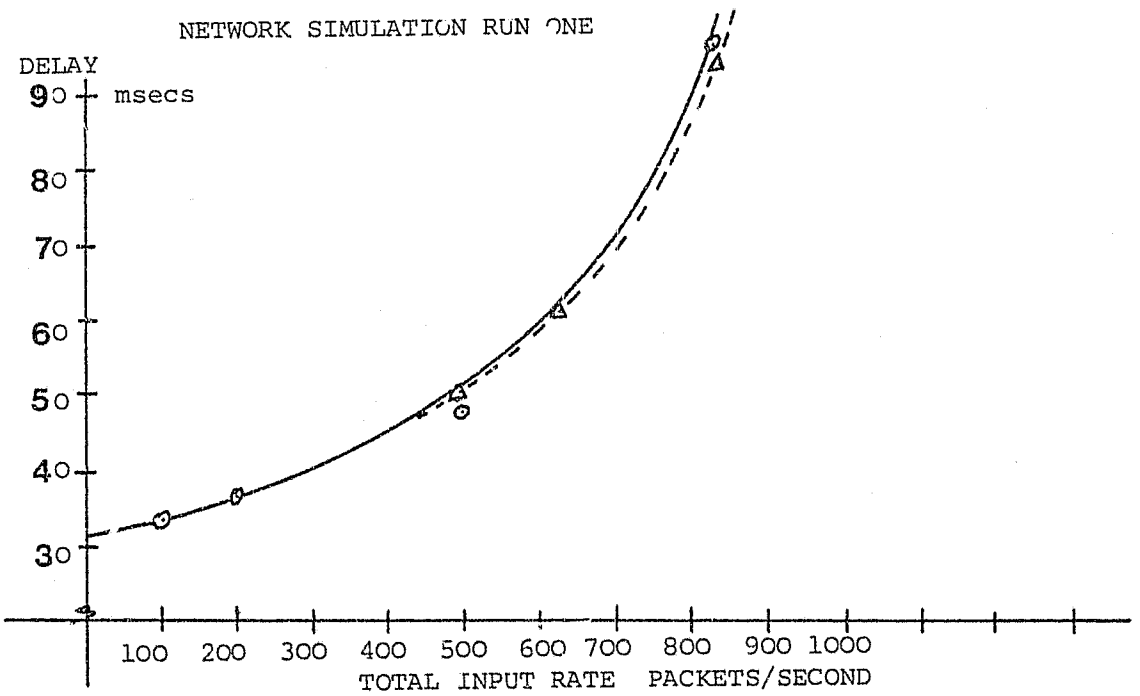
in mean and variance values are attained. This method is impractical for the models tested here, where mean and variances tend to increase very rapidly when the network enters a congested state. It was therefore necessary to resort to the method described above.

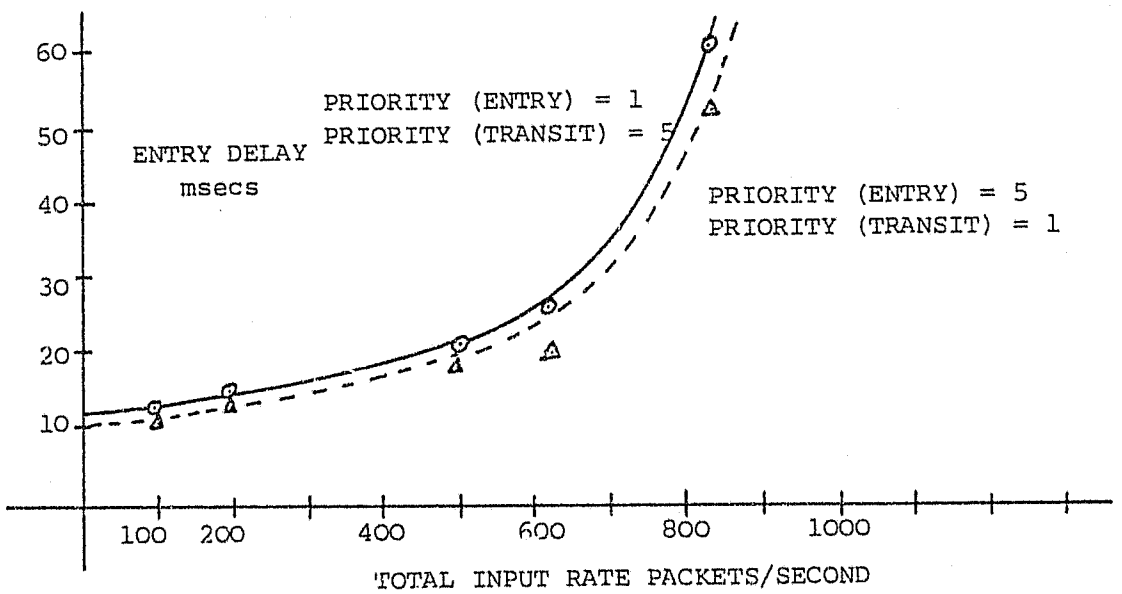
7.3.3 Response and Throughput

The response and throughput for both network and nodal simulations is discussed. Again, bus rates were set at 640 KBPS, with the retransmission interval at 40 milliseconds. Details of the results are to be found in Tables E.13 to E.23, Appendix E.

Simulations have been repeated three times for a given traffic input. The seeds of random number generators injecting packets with variable length and interarrival times were different for each of the three runs. These are termed Run One, Run Two, and Run Three, they are shown in Figures 7.6 to 7.11. The repetition of runs meant that a mean value for throughput could be found, and that the results from a number of runs using different packet inputs (following the same distribution function, however) could be compared.

The mean response has been plotted for total entry and transit delays, corresponding deviation values may be found in Appendix E.


NETWORK SIMULATION RUN ONE
DELAY
msecs
90
80
70
60
50
40
30
100 200 300 400 500 600 700 800 900 1000
TOTAL INPUT RATE PACKETS/SECOND


60
50
40
30
20
10
ENTRY DELAY
msecs
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND

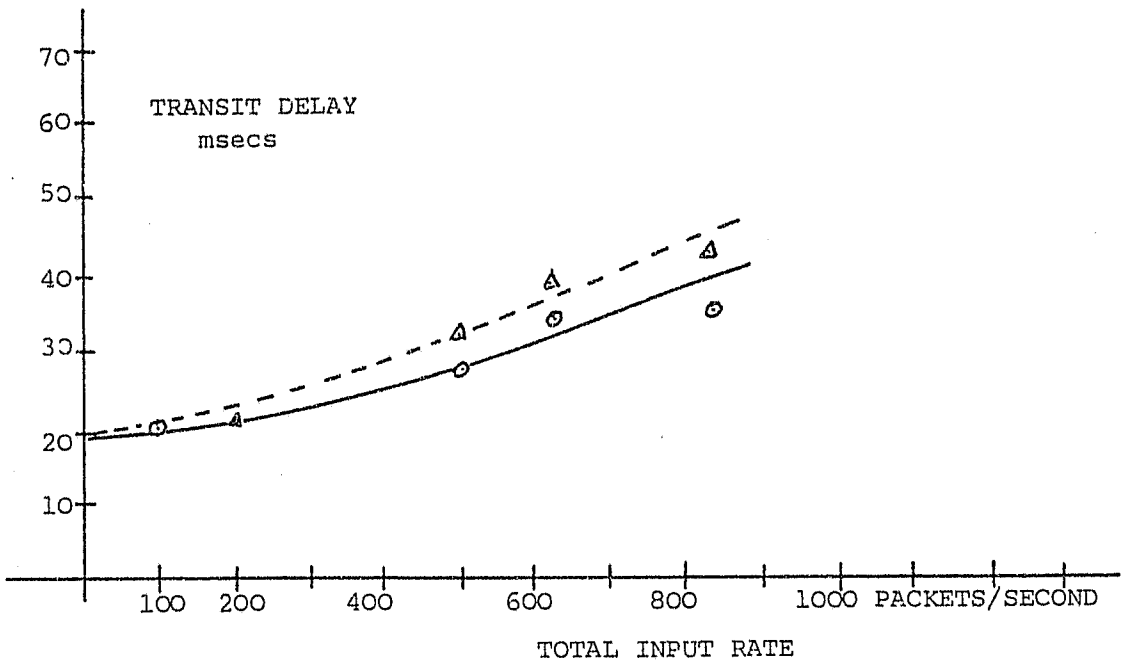

70
60
50
40
30
20
10
TRANSIT DELAY
msecs
100 200 400 600 800 1000 PACKETS/SECOND
TOTAL INPUT RATE

Figure 7.6

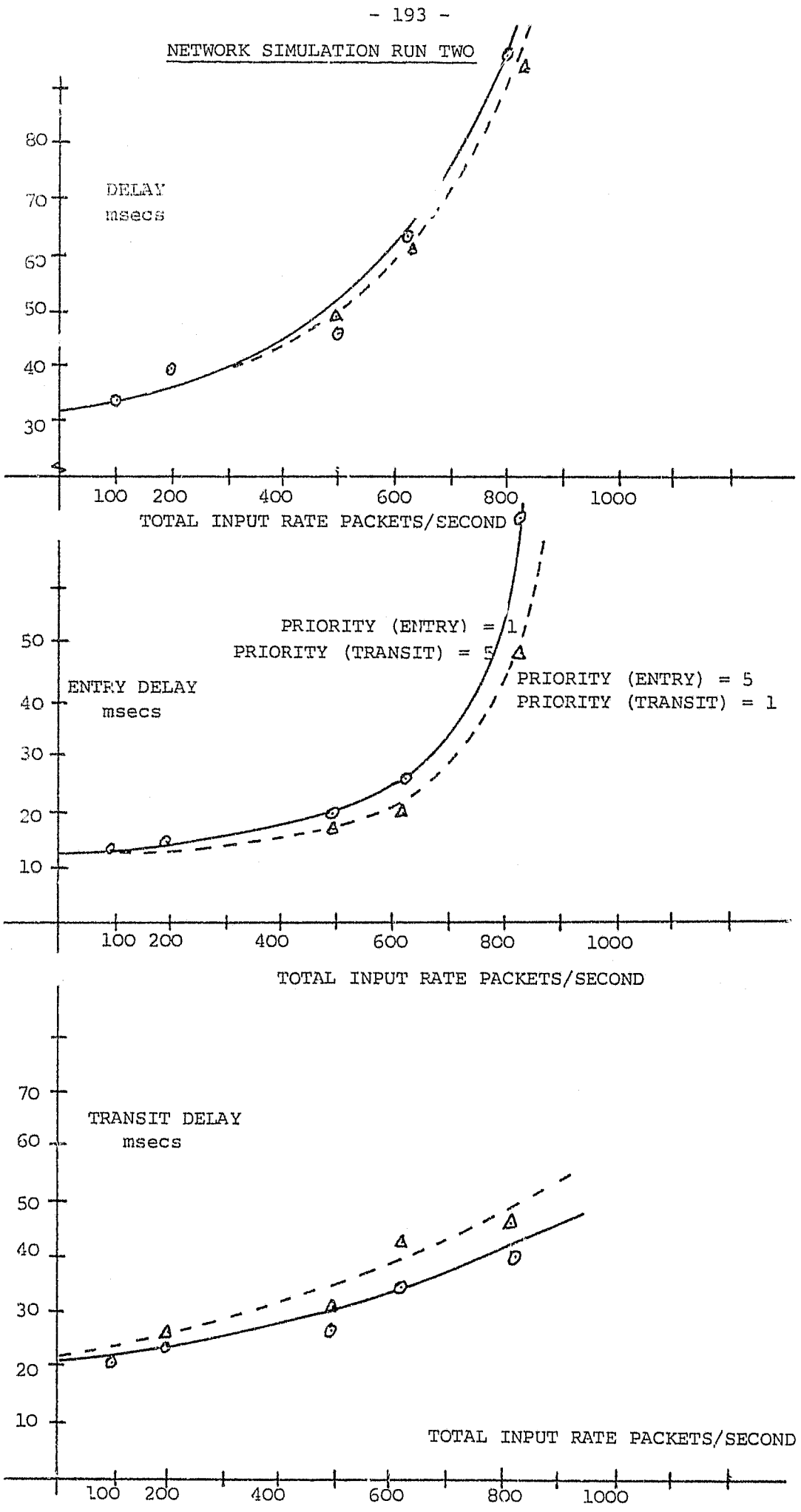
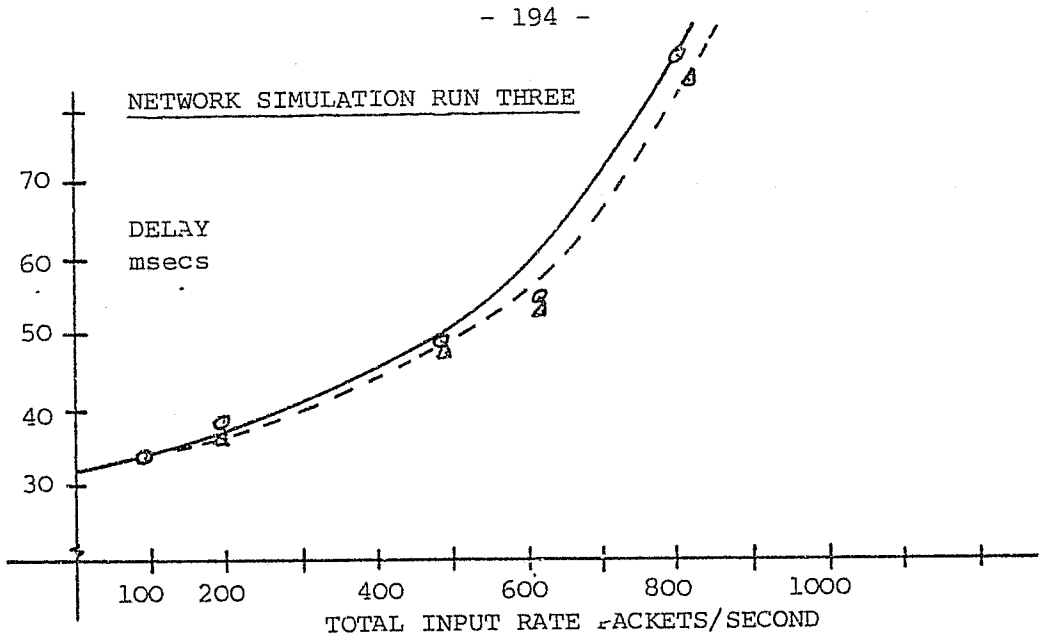
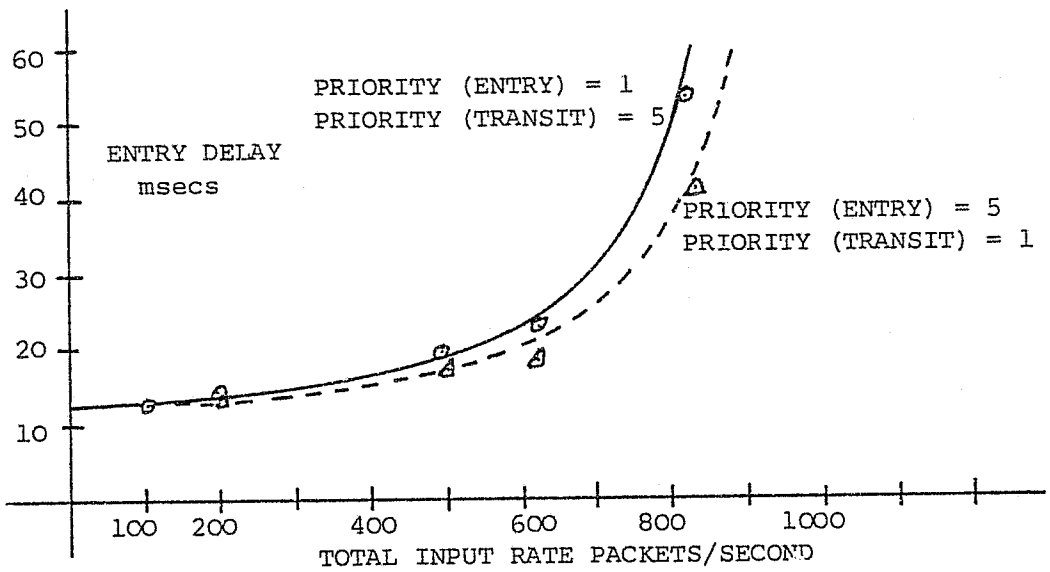

NETWORK SIMULATION RUN TWO
DELAY
msecs
80
70
60
50
40
30
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
ENTRY DELAY
msecs
50
40
30
20
10
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND
TRANSIT DELAY
msecs
70
60
50
40
30
20
10
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND

Figure 7.7


NETWORK SIMULATION RUN THREE
DELAY
msecs
70
60
50
40
30
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND


ENTRY DELAY
msecs
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
60
50
40
30
20
10
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND

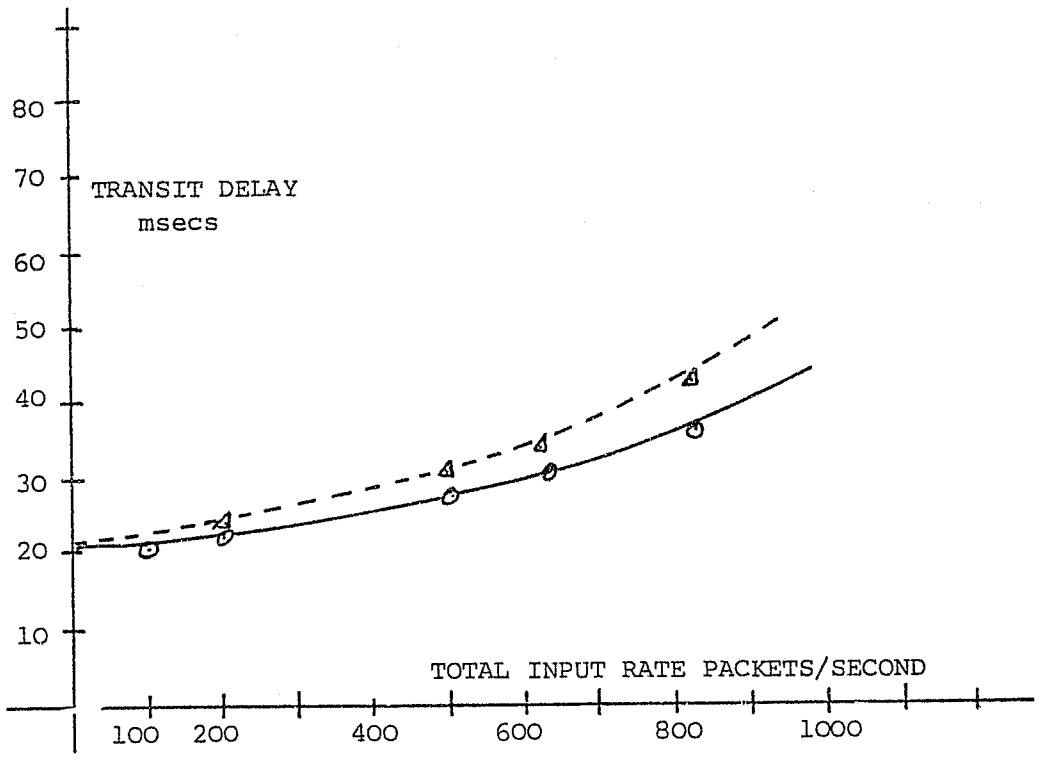

TRANSIT DELAY
msecs
80
70
60
50
40
30
20
10
TOTAL INPUT RATE PACKETS/SECOND
100
200
400
600
800
1000

Figure 7.8

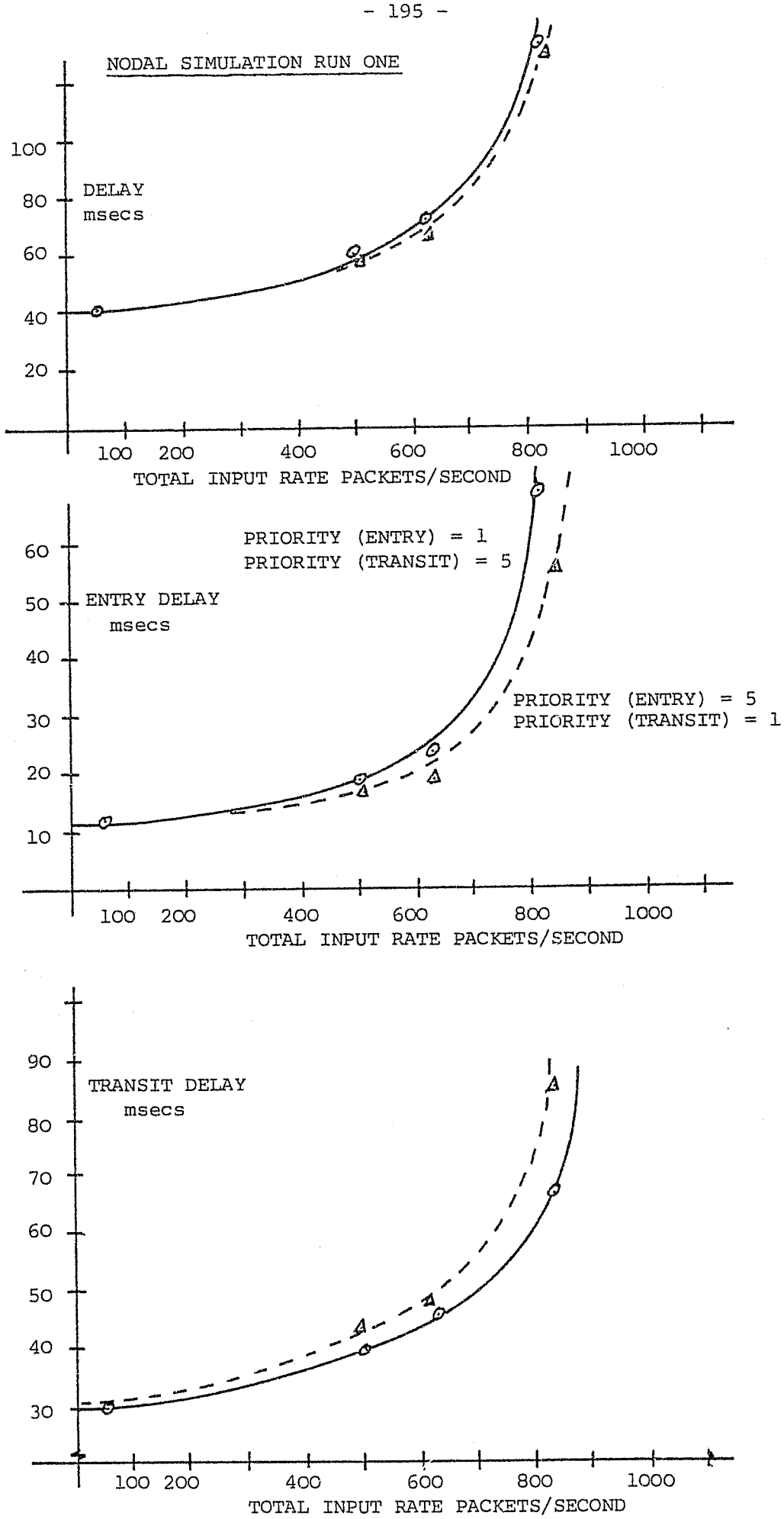

NODAL SIMULATION RUN ONE
DELAY
msecs
100
80
60
40
20
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
ENTRY DELAY
msecs
60
50
40
30
20
10
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND
TRANSIT DELAY
msecs
90
80
70
60
50
40
30
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND

Figure 7.9

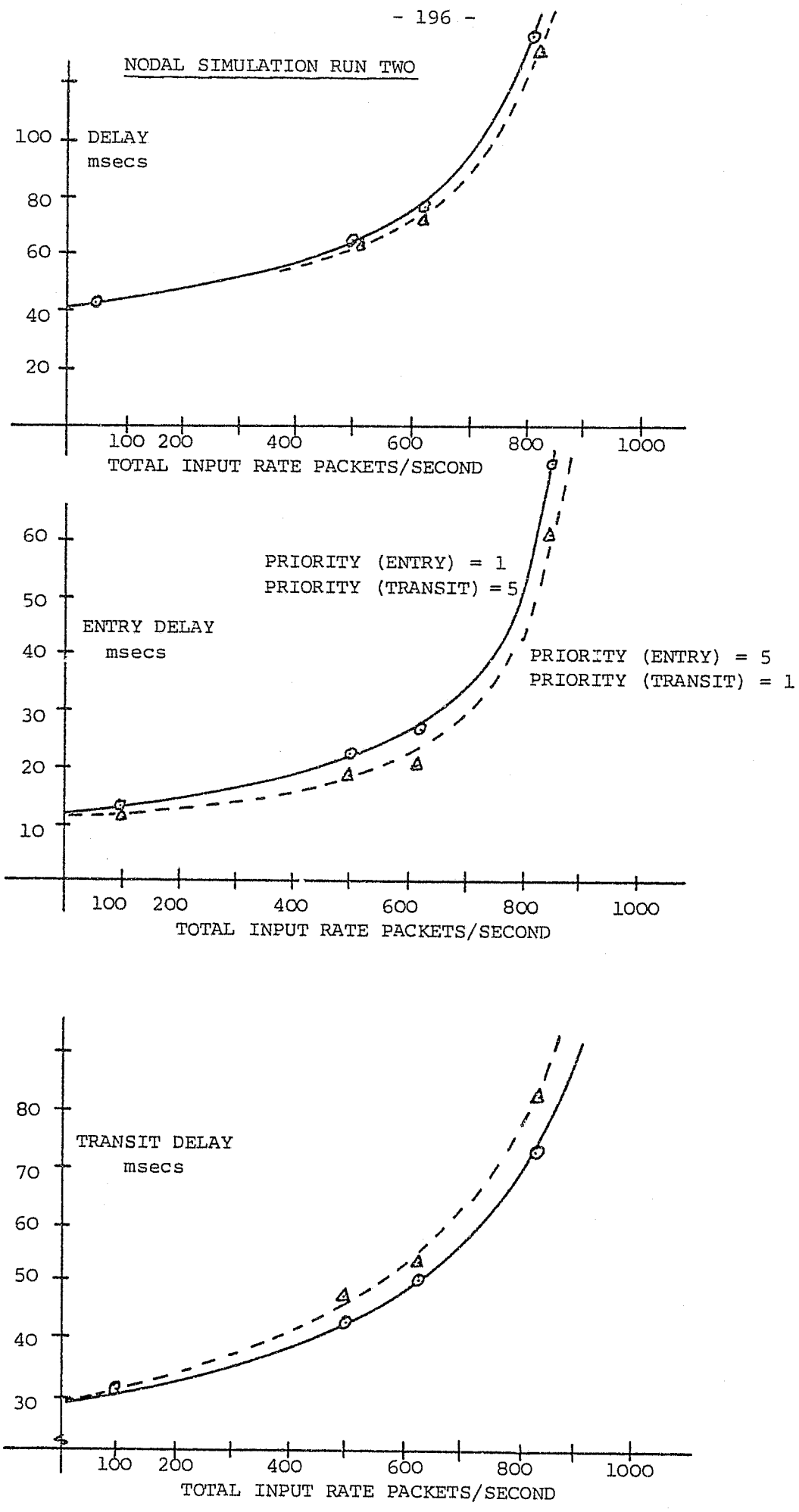
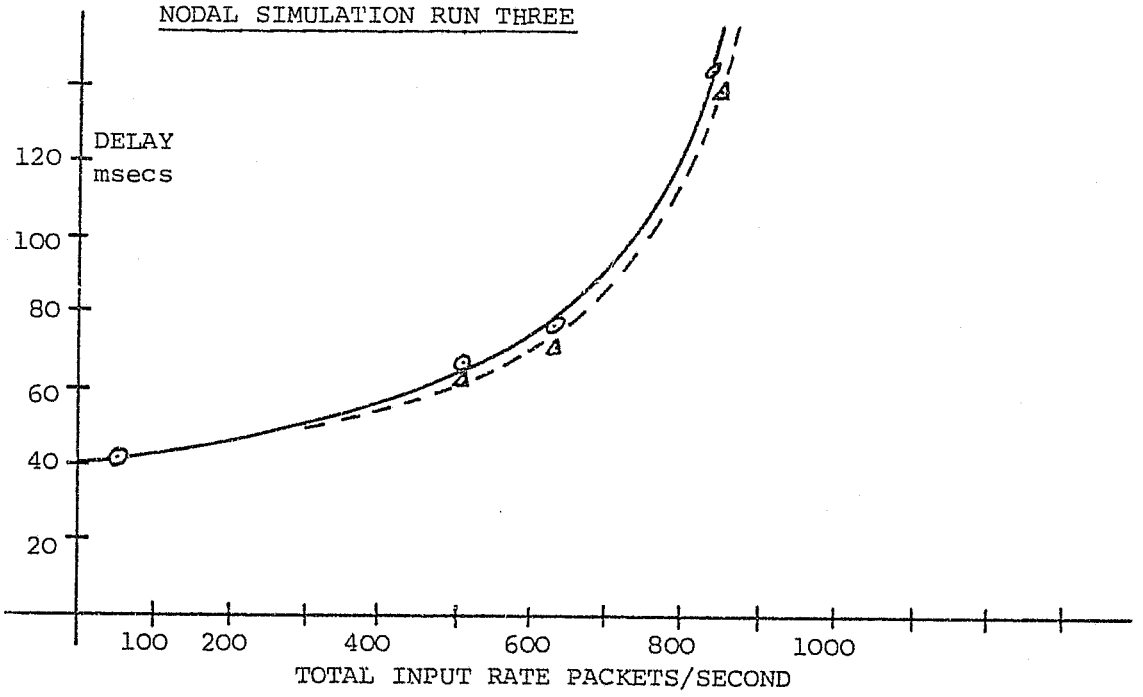
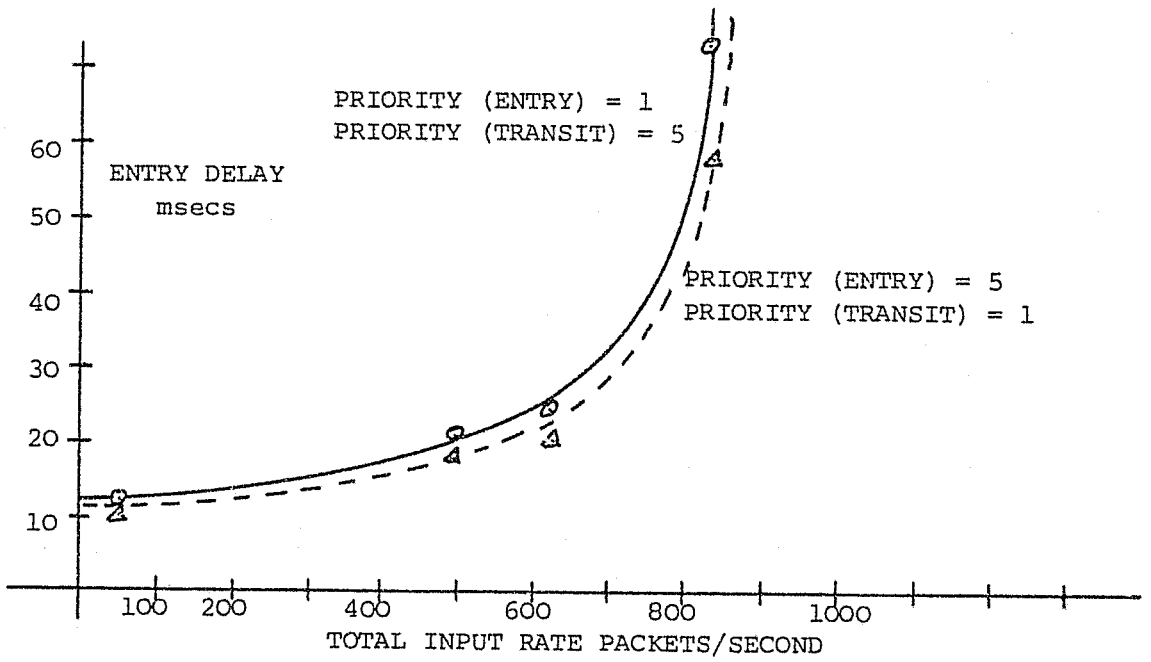

NODAL SIMULATION RUN TWO
DELAY
msecs
100
80
60
40
20
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
ENTRY DELAY
msecs
60
50
40
30
20
10
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND
TRANSIT DELAY
msecs
80
70
60
50
40
30
100 200 400 600 800 1000
TOTAL INPUT RATE PACKETS/SECOND

Figure 7.10


NODAL SIMULATION RUN THREE
DELAY
msecs
120
100
80
60
40
20
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND


PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
ENTRY DELAY
msecs
60
50
40
30
20
10
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND

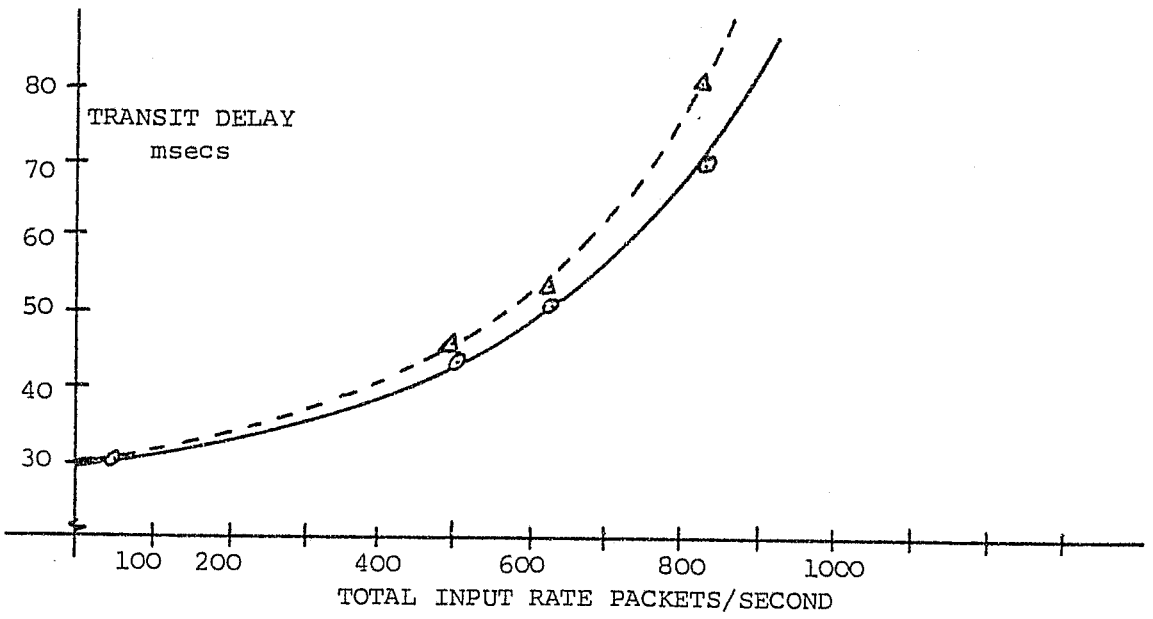

TRANSIT DELAY
msecs
80
70
60
50
40
30
100
200
400
600
800
1000
TOTAL INPUT RATE PACKETS/SECOND

Figure 7.11

The curves plotted for the three runs do not differ significantly. In all case : curves are of the exponential form. The response time as a function of input increases gradually until a critical input level is reached (in the region of 500 packets per second for both networks), thereafter the mean time required for packets to traverse the network becomes extremely long.

The curve characteristic is summed up in Table 7.1

Network (msecs) Simulation	Nodal (msecs) Simulation	Input Levels in Packets/sec
Total Delay Increment 12 Entry Delay Increment 8	20 9	100 to 500 100 to 500
Total Delay Increment 50 Entry Delay Increment 40	80 50	500 to 800

Table 7.1 Response Characteristic

The curves show that the main component of the delay is that of transit. The time taken for a packet to traverse the network takes longer than the time spent in waiting for - and being serviced by an interface processor. This result only holds for the network in an uncongested state. Congestion causes the mean entry values to exceed those of transit delay. Under this condition the network starts to block packets attempting to enter the switching section.

Channel congestion may be confirmed by examining the line utilization factors in Appendix E. For example, Channel 4 in the network simulation, recorded factors of 0,78, 0,73, 0,72, and 0,80, 0,80, 0,84 at input level 835 packets per second for each of the three runs for priorities, PRIORITY (TRANSIT) = 5, PRIORITY (ENTRY) = 1 and PRIORITY (TRANSIT) = 1, PRIORITY (ENTRY) = 5, respectively.

Values for mean throughput are found plotted in Figures 7.12 and 7.13. These curves hold only for data packets transmitted from source to destination ICP. For comparison, a linear curve has been drawn indicating at which points the input and throughput packet rates coincide. As the input level is increased the network is able to maintain its throughput at the equivalent rate. At a certain input rate, however, the input starts to exceed the capacity of the network to transmit packets. Such occurs at an input level of approximately 500 packets per second.

Comparison with the delay curves shows that the above input rates at which the model throughput curves start to deviate from the ideal throughput plot, lie approximately in the same region as the input rates for which the mean packet delays start to increase rapidly and become unbounded. This behaviour results due to channel saturation. This statement may be substantiated by examining the contents of Table 7.2 depicting the behaviour of node 1 as a function of input traffic levels.

NODAL SIMULATION THROUGHPUT

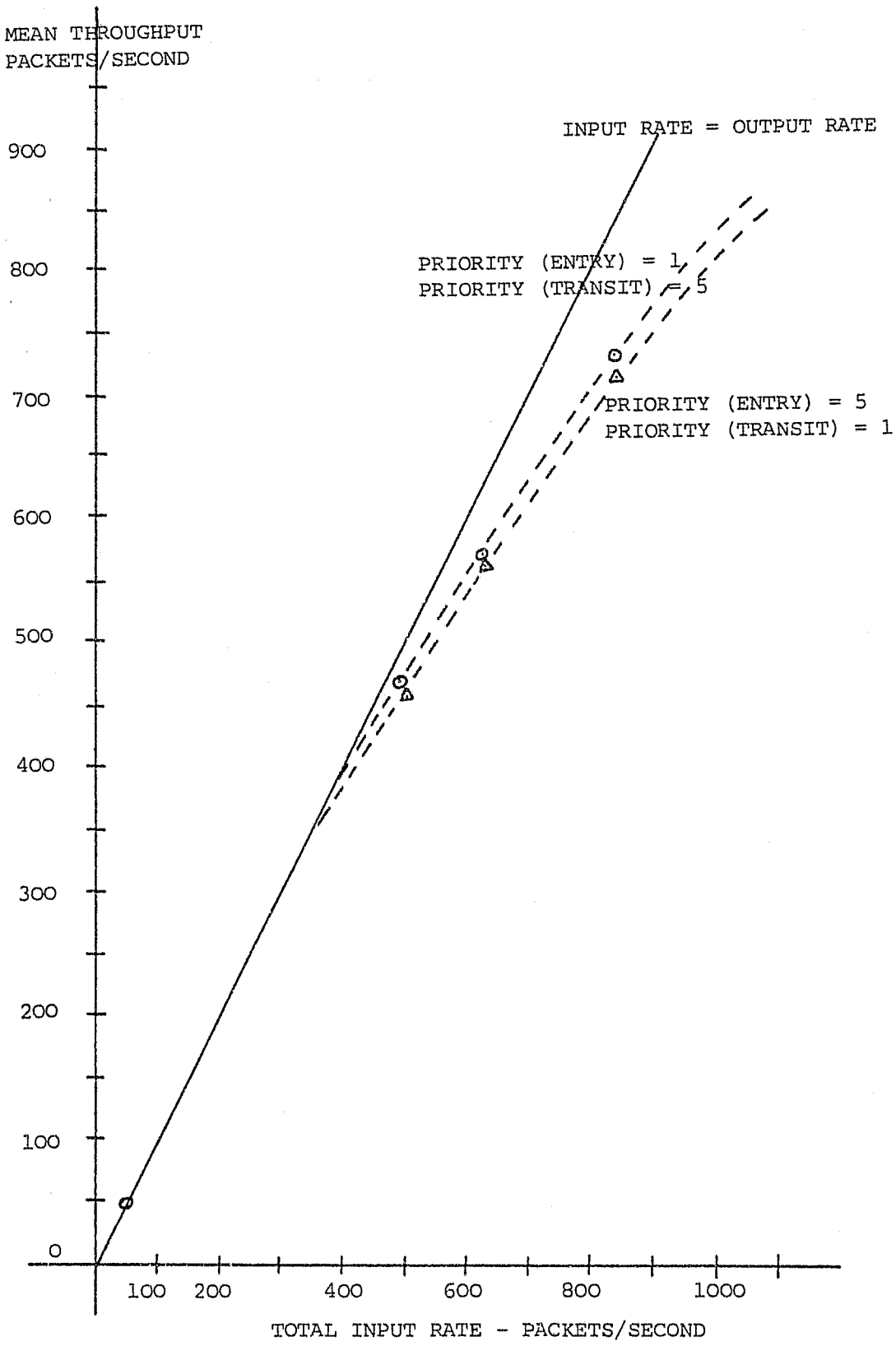


Figure 7.12

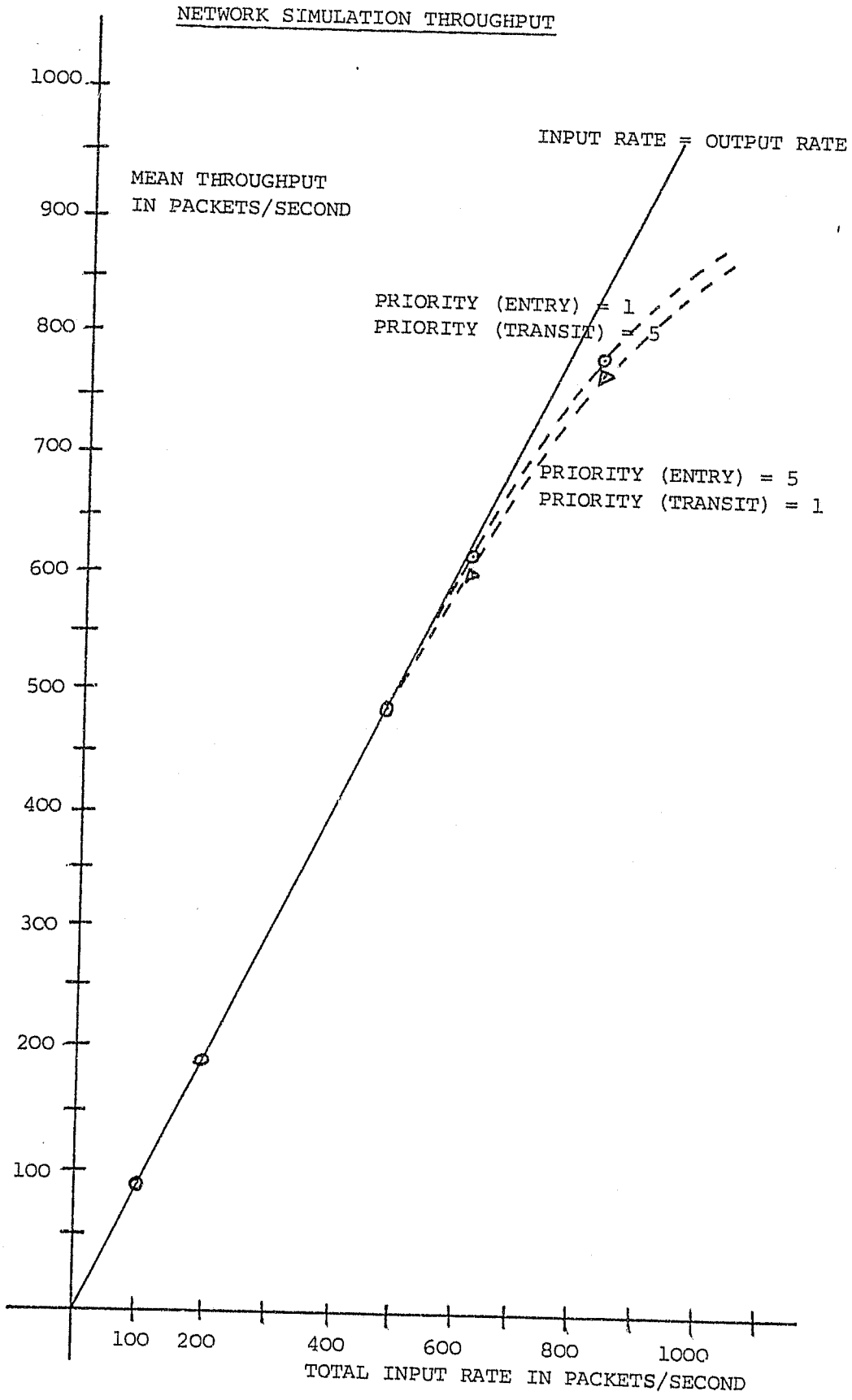

NETWORK SIMULATION THROUGHPUT
MEAN THROUGHPUT
IN PACKETS/SECOND
1000
900
800
700
600
500
400
300
200
100
INPUT RATE = OUTPUT RATE
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT) = 5
PRIORITY (ENTRY) = 5
PRIORITY (TRANSIT) = 1
100
200
400
600
800
1000
TOTAL INPUT RATE IN PACKETS/SECOND

Figure 7.13

PERFORMANCE OF NODE1 : NETWORK SIMULATION		
RUN TWO : PRIORITY (ENTRY) = 1, PRIORITY (TRANSIT) = 5		
BUS UTILIZATION : 22 VIRTUAL CIRCUITS HANDLED : DERIVED FROM TABLE E.14 : CHANNEL CAPACITY 640 KBPS		
INPUT/VC: PCKS/SEC	RECORDED	PREDICTED
2	0,03	0,03
4	0,07	0,06
10	0,18	0,16
12,5	0,21	0,20
16,7	0,28	0,27
LINE UTILIZATION : LINE 1 and LINE 4 : 6 VIRTUAL CIRCUITS HANDLED EACH : LINE 3 and LINE 7 : 4 VIRTUAL CIRCUITS HANDLED EACH : DERIVED FROM TABLE E.15 : CHANNEL CAPACITY 64 KBPS		

INPUT/VC: PCKS/SEC	PREDICTED	RECORDED		PREDICTED	RECORDED	
		LINE 1	LINE 4		LINE 3	LINE 7
2	0,09	0,09	0,09	0,06	0,07	0,06
4	0,18	0,21	0,20	0,12	0,13	0,13
10	0,44	0,42	0,56	0,29	0,44	0,29
12,5	0,55	0,60	0,66	0,37	0,43	0,36
16,7	0,74	0,80	0,83	0,49	0,76	0,88

Table 7.2 NODE1 Performance

The predicted values were found by determining the product of the mean packet length (470 bits) and the mean total input rate to the resource, divided by the channel capacity. There is little difference between the predicted and recorded bus utilization factors. At the lower input levels (less than 500 packets per second) the line utilization factors for predicted and simulated cases are similar, but at the higher input rates the recorded factors are greater than those predicted. It must therefore be inferred that considerable retransmissions due to blocked buffers take place, resulting in a corresponding performance degradation of the network.

It is noted that no restrictions were placed on the number of packets entering the network; the majority of packets were generated by virtual circuits. In practice the number allowed access to the network would be limited, each logical link being allowed to transmit some specified maximum number of packets at one time.

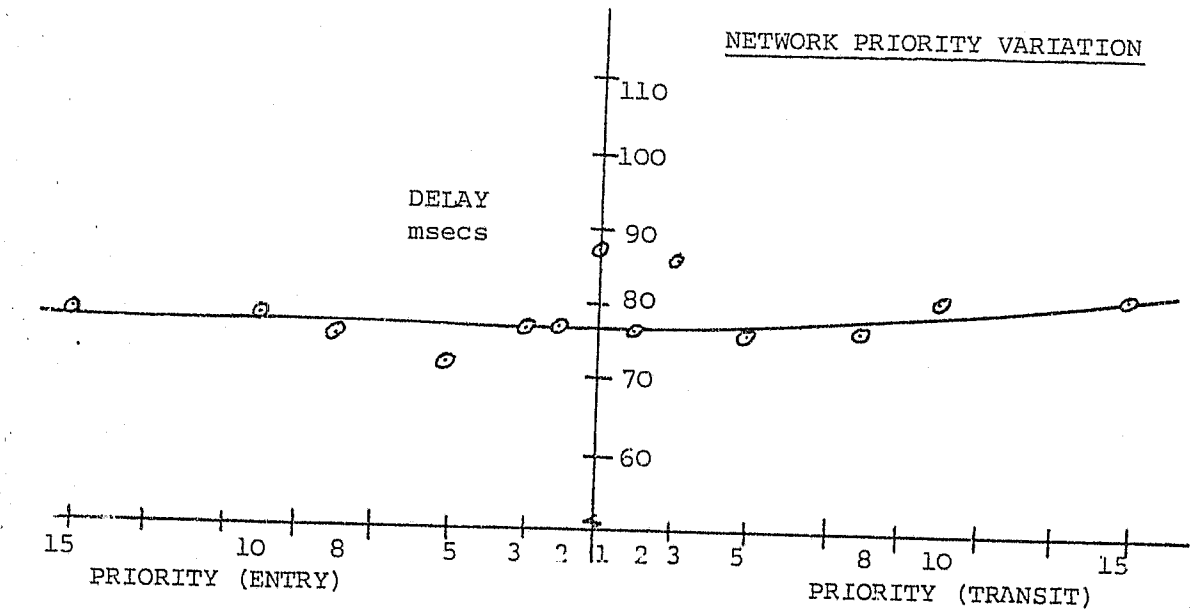
Finally, examination of the line utilization factors for the network simulation indicates that saturation of only a few channels can cause performance degradation. From Table E.15, a limited number of channels are utilized highly with factors in the region of 0,8 (channels 1, 4, 8, 9, 14), whilst the remainder have factors ranging from 0,35 to 0,76. Adaptive routing is one solution, but only up to a certain input load. Thereafter congestion occurs.

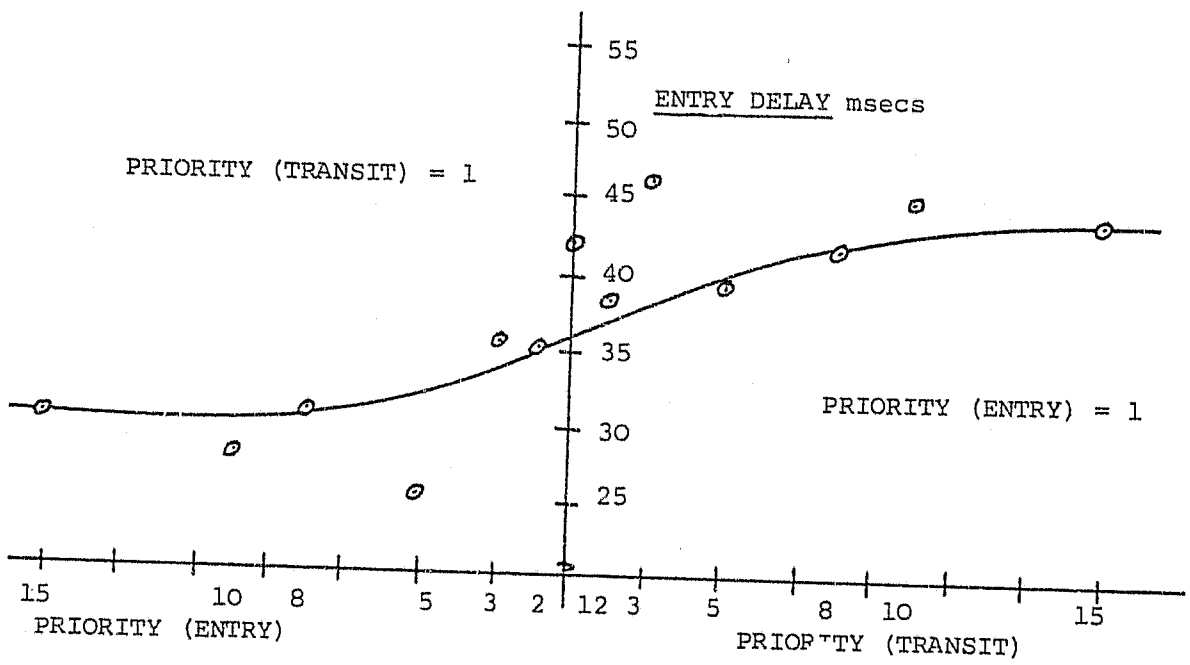
7.3.4 Switching Algorithm Priority Variations

This section deals with a proposed method for local flow control; by local is meant operating in the immediate surroundings of the node. A distinction has been made between packets entering, and packets in transit or leaving the network. The two cases are denoted by PRIORITY (ENTRY) and PRIORITY (TRANSIT) respectively. Weighting factors are allocated to packets waiting for service in a node, and these are incremented with time by the appropriate priority value. The initial priority of packets arriving at a node is zero. The aim of the simulation is to determine whether less blocking in the switching section of the network will result if packets in transit or leaving the network be given higher priority increments than those entering.

The details of this section are contained in Tables E.22 and E.23, Appendix E. The tests were conducted on the network simulation, using an input level of 750 packets per second. Network parameters such as the bus rate and retransmission interval are the same as for previous runs.

The results are plotted in Figure 7.14. The total delay curve shows little variation as a function of priority, being slightly longer when the transit priority values exceed those of the entry values. In the region where both priority cases take on an increment value of one, the points are scattered conveying


NETWORK PRIORITY VARIATION
DELAY
msecs
110
100
90
80
70
60
15 10 8 5 3 2 1 2 3 5 8 10 15
PRIORITY (ENTRY)
PRIORITY (TRANSIT)


PRIORITY (TRANSIT) = 1
ENTRY DELAY msecs
55
50
45
40
35
30
25
15 10 8 5 3 2 1 2 3 5 8 10 15
PRIORITY (ENTRY)
PRIORITY (ENTRY) = 1
PRIORITY (TRANSIT)

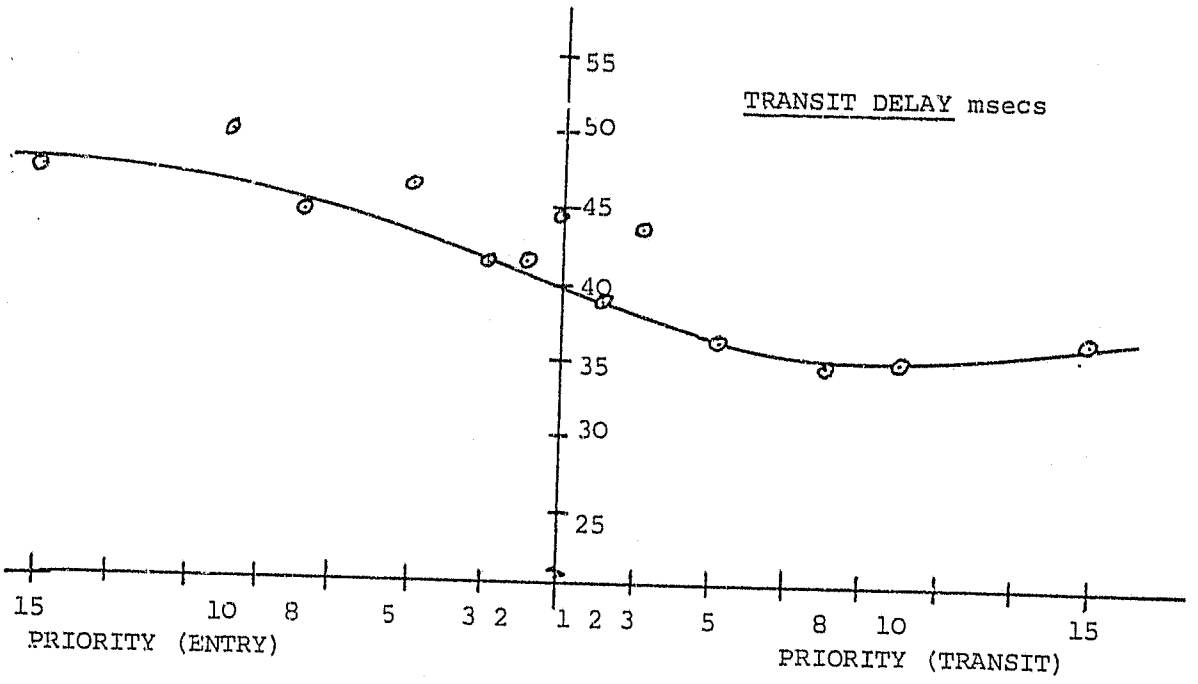

TRANSIT DELAY msecs
55
50
45
40
35
30
25
15 10 8 5 3 2 1 2 3 5 8 10 15
PRIORITY (ENTRY)
PRIORITY (TRANSIT)

Figure 7.14

a certain instability in delay performance. When either of the priority increments is increased to 15, the other kept at 1, the delays tend to a more defined limit; in the case of transit priority greater, to 80 milliseconds, for the entry priority greater 77 milliseconds.

The entry and transit delays show clearly the effect of priority variations. Higher entry increment values cause a decrease in the waiting time spent in the ICP, whilst a larger transit increment causes a decrease in the time required for a packet to traverse the network. As with the total delay, the entry and transit delays tend to certain limits, these being 30 and 45 milliseconds for entry, 47 and 35 milliseconds for transit responses. That the curves do tend to certain limits is substantiated by the analytically derived graph shown in Figure 2.3 of Chapter 2. Further increases in priority increments than those indicated will have little effect.

The time-dependent priority technique was purposely chosen since its effect becomes apparent at the higher traffic intensities, when the ability to give precedence to transit traffic is most needed. Such may be seen from Figures 7.6 to 7.8. The difference in response times for the two priority cases becomes greater with increase in the traffic input level.

The allocation of priorities to packets and the queueing

structure inherent in the network are properties that allow for an approximation to a conservative system; this means that no work is created or destroyed in the system. For example, destruction of work would occur if a packet were to leave the system before completing its service, and the creation of work might correspond to a server standing idle in the face of a non-empty queue. These systems are termed work conserving. In priority systems this means that preferential treatment given to one class of customers is afforded at the expense of another class.(21) It can be shown(21) that so long as the queueing discipline selects customers in a way that is independent of their service time, then the distribution of the mean waiting time of customers is invariant to the order of service, i.e., any attempt to modify the genuine discipline so as to reduce the waiting time of one group of packets, will force an increase in the waiting time of another group. This is the phenomenon that has occurred for entry and transit delays as shown in Figure 7.14. That the total delay remains more or less constant is due to the fact that all packets at some stage become members of both priority groups. The mean entry and transit responses can thus be controlled, but the total delay cannot be manipulated, other than by controlling the input loads.

In practice the switching processors would function by allocating transit traffic higher priorities. The effect of this would be a better transit response, whilst the main time spent in waiting would be at the ICP stage where the greater portion of storage is concentrated.

An assumption inherent in conservative queueing systems is that the amount of storage is infinite, i.e., no blocking of packets takes place. This condition is met when the input loads are less than about 600 packets per second and the line utilization factors less than 0,70. In this case the amount of blocking is extremely small (see Figure 5.21 of Chapter 5). At the higher input loads, more blocking occurs so that the conservation principle is less applicable, resulting in slight variations in the total delay for the two priority cases.

A better transit response will result in more packets per unit time interval being able to make use of the finite number of resources in the switching section. In store-and-forward buffering the longer a packet occupies a buffer, whether due to line errors or blocked destination storage, the slower the effective circuit rate becomes. The smaller transit response time implies that fewer packets are blocked on traversing the network.

The priority technique cannot be used as a measure on its own. It does not provide for sufficient rejection of packets when certain of the network regions enter a state of congestion. Global flow control wherein the number of logical links and the number of packets per logical link is limited, must be incorporated.

7.3.5 Preview Concerning Nodal Characteristics

The following sections are concerned with the study of certain nodal parameters and their influence on the capacity of the node to transmit packets. The tests were conducted for the nodal simulation. Nodal parameters are altered, but in a network environment. Thus a parameter such as the data bus rate is varied for each node, and its effect on the mean packet delay in the network is recorded. When a parameter is varied it is done so for all nodes, being set to the same value for all nodes. Finally, the priority setting for all tests conducted was PRIORITY (TRANSIT) = 5, PRIORITY (ENTRY) = 1.

7.3.6 Variation of the Retransmission Interval

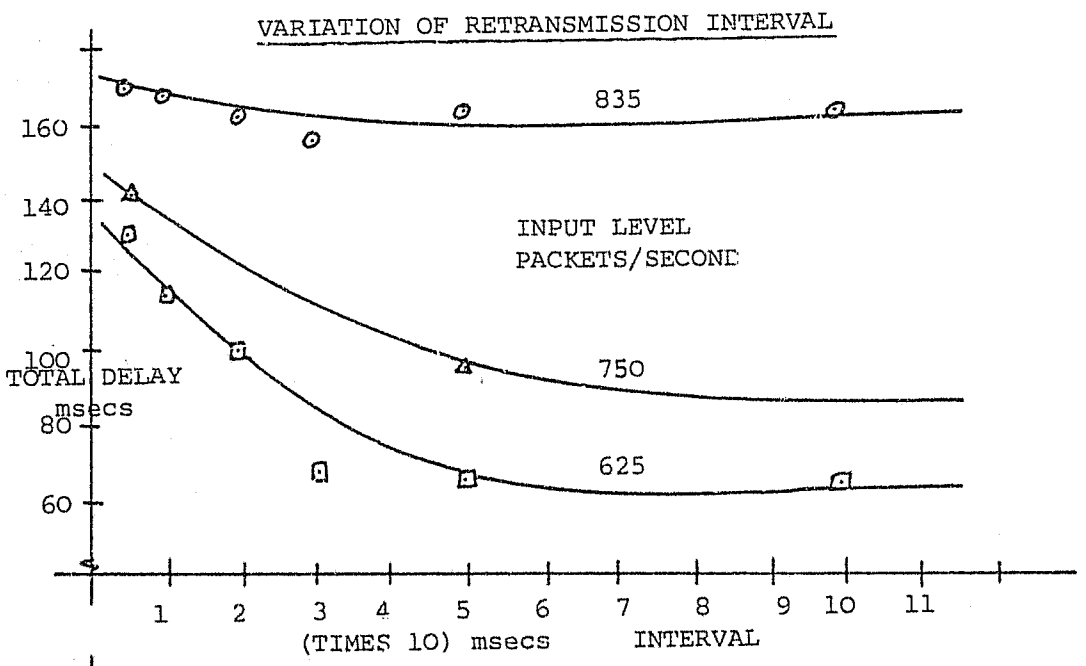
The retransmission curves are shown in Figures 7.15 to 7.17. The line and bus utilization factors are to be found in Tables E.24 and E.26 of Appendix E.

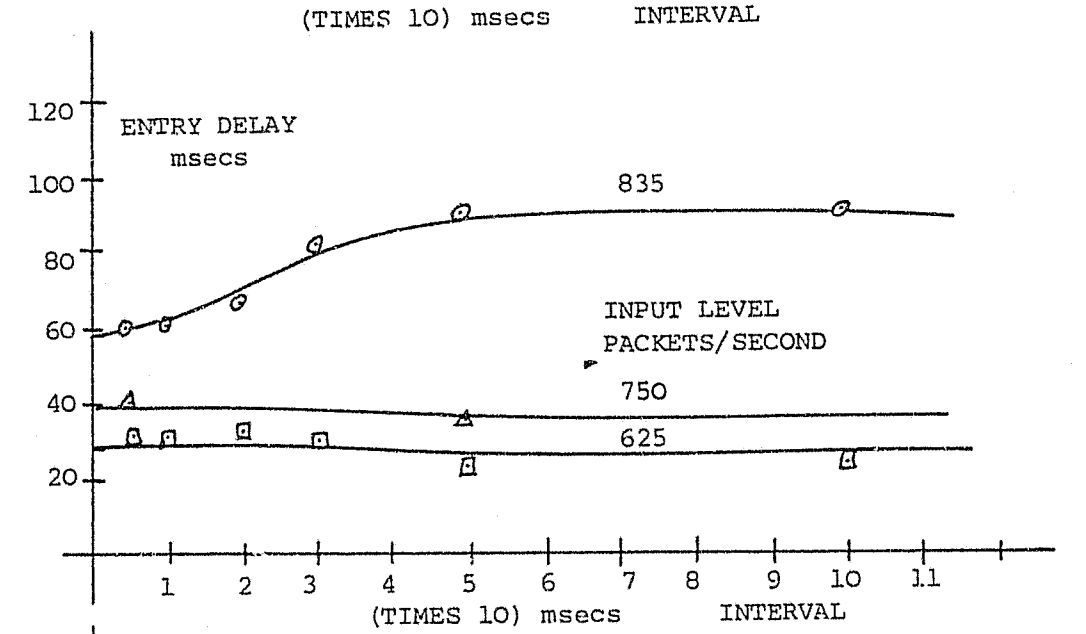
A retransmission may take place in two ways. A packet transmitted to a destination channel controller may find all storage occupied; a retransmission is attempted after some interval of time. Alternatively, the packet itself or its acknowledgement may have incurred errors. The source controller will retransmit the packet after a certain time period, having received no acknowledgement. The duration of time after which retransmission takes place is the same for both cases.

Figure 7.15 shows the effect of variation in interval duration on the mean packet response. The curves are characterized by two main parts; a section where no variation in response results for the interval period ranging from 35 to 100 milliseconds, and a section where the period is less than 30 milliseconds causing the mean packet delay to increase rapidly.

The threshold behaviour occurs when the retransmission interval is decreased to such an extent that it is of shorter duration than the time required to receive an acknowledgement signal. A rough set of calculations is in order to substantiate this.

The time taken to transmit a packet of mean length 470 bits via a 64 KBPS link is about 7,3 milliseconds. Assuming a control packet of length 70 bits to be directly transmitted as acknowledgement, a lower bound for the process is 7,3 (packet) + 1,1 (ACK) or 8,4 milliseconds. An upper bound occurs when the acknowledgement must wait for the line to become free; it is assumed an averaged sized packet of 470 has just been transmitted. In addition, it is assumed the acknowledgement is piggybacked when the line becomes available. The upper bound is then 7,3 (packet) + 7,3 (wait for channel) + 7,3 (piggybacked ACK) or 22 milliseconds. The threshold behaviour for the simulation occurs at 30 milliseconds. It may be inferred that packets are being retransmitted without properly ascertaining whether errors or blocked storage have in fact occurred.


VARIATION OF RETRANSMISSION INTERVAL
160
140
120
100
80
60
TOTAL DELAY
msecs
835
INPUT LEVEL
PACKETS/SECOND
750
625
1 2 3 4 5 6 7 8 9 10 11
(TIMES 10) msecs
INTERVAL


120
100
80
60
40
20
ENTRY DELAY
msecs
835
INPUT LEVEL
PACKETS/SECOND
750
625
1 2 3 4 5 6 7 8 9 10 11
(TIMES 10) msecs
INTERVAL

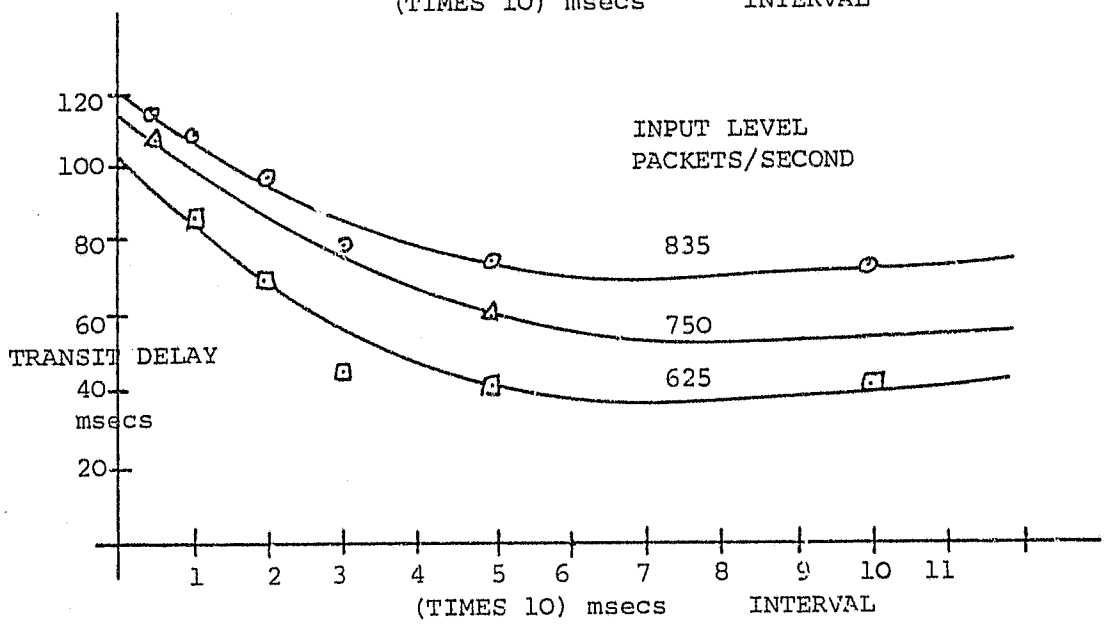

120
100
80
60
40
20
TRANSIT DELAY
msecs
INPUT LEVEL
PACKETS/SECOND
835
750
625
1 2 3 4 5 6 7 8 9 10 11
(TIMES 10) msecs
INTERVAL

Figure 7.15

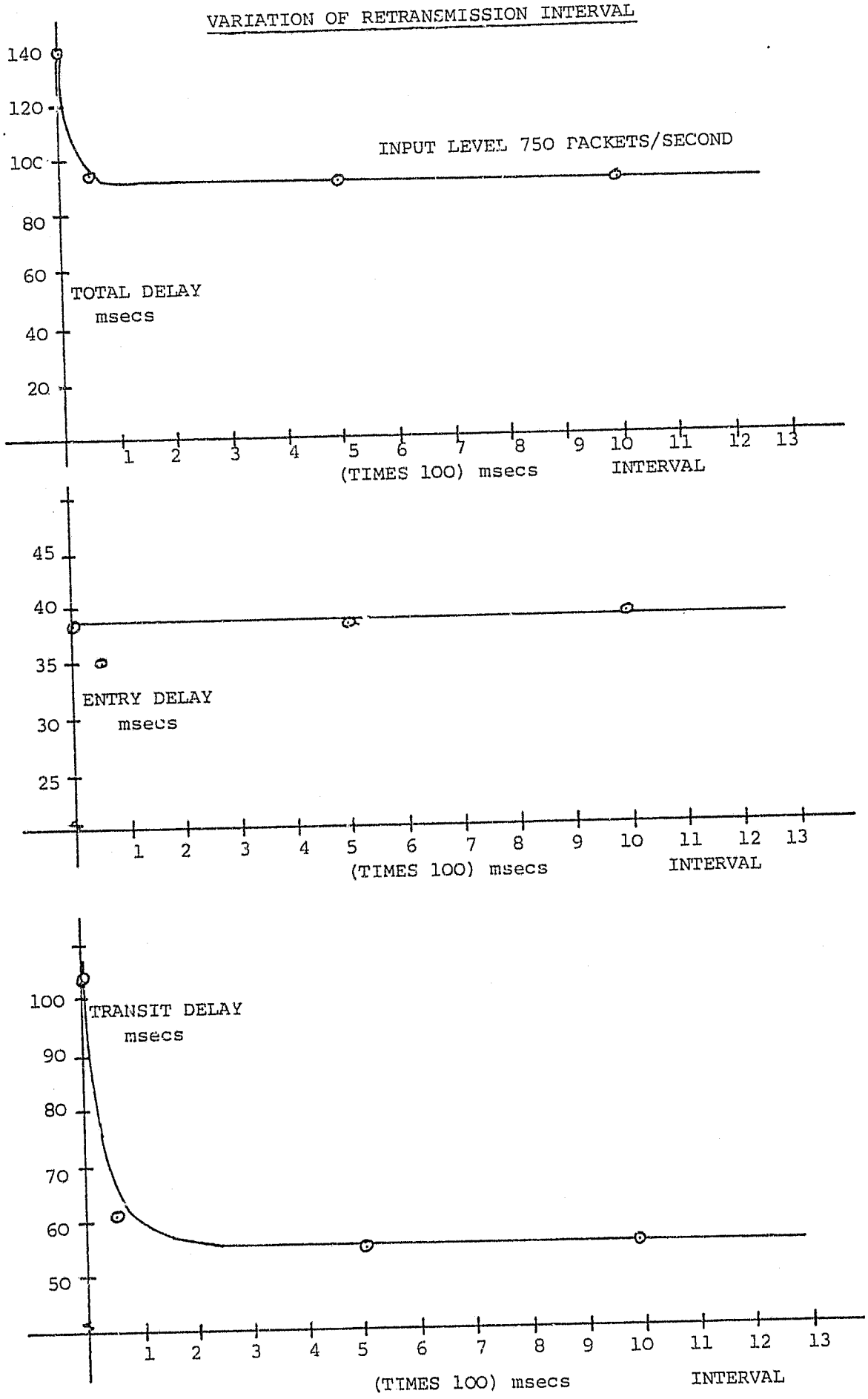
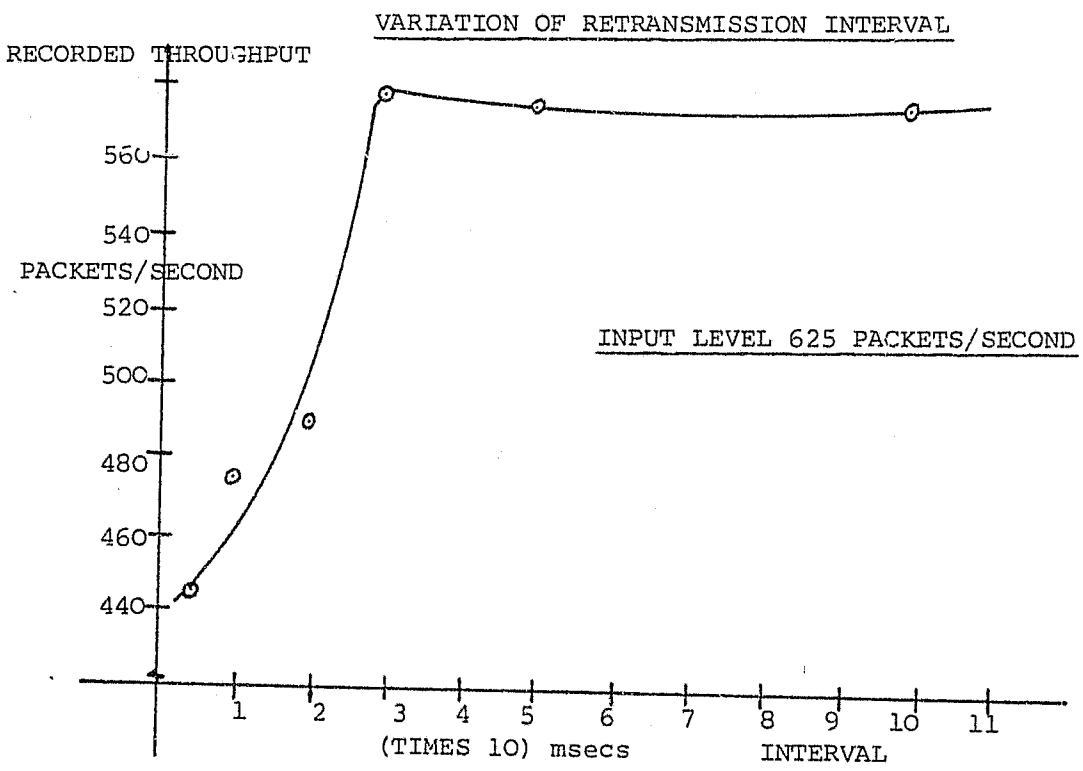

VARIATION OF RETRANSMISSION INTERVAL
140
120
100
80
60
40
20
INPUT LEVEL 750 PACKETS/SECOND
TOTAL DELAY
msecs
1 2 3 4 5 6 7 8 9 10 11 12 13
(TIMES 100) msecs
INTERVAL
45
40
35
30
25
ENTRY DELAY
msecs
1 2 3 4 5 6 7 8 9 10 11 12 13
(TIMES 100) msecs
INTERVAL
100
90
80
70
60
50
TRANSIT DELAY
msecs
1 2 3 4 5 6 7 8 9 10 11 12 13
(TIMES 100) msecs
INTERVAL

Figure 7.16


VARIATION OF RETRANSMISSION INTERVAL
RECORDED THROUGHPUT
560
540
PACKETS/SECOND
520
500
480
460
440
INPUT LEVEL 625 PACKETS/SECOND
1
2
3
4
5
6
7
8
9
10
11
(TIMES 10) msecs
INTERVAL

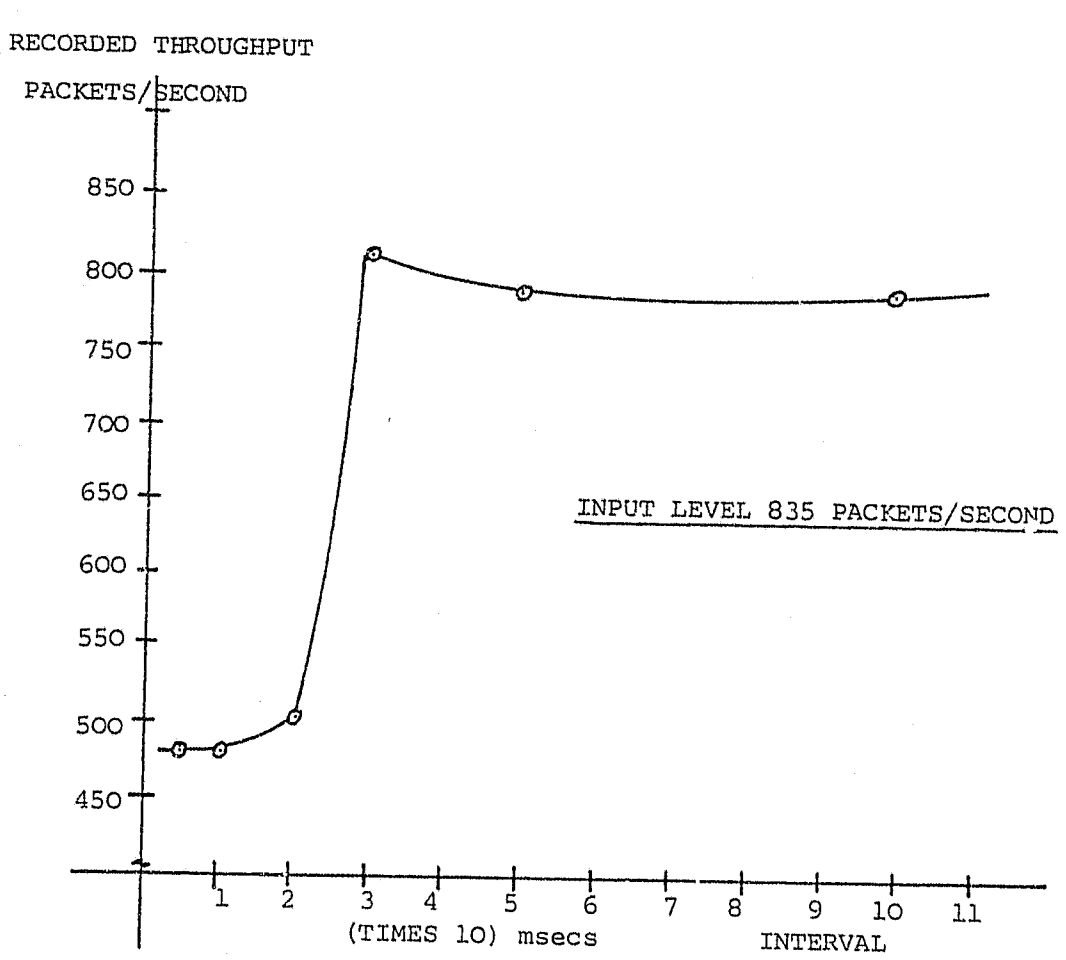

RECORDED THROUGHPUT
PACKETS/SECOND
850
800
750
700
650
600
550
500
450
INPUT LEVEL 835 PACKETS/SECOND
1
2
3
4
5
6
7
8
9
10
11
(TIMES 10) msecs
INTERVAL

Figure 7.17

Examination of the line utilization factors confirms the increased channel activity. As an example, at the input rate of 625 packets/sec. in channel 10, the factors at the interval periods of 50, 30 and 10 milliseconds are respectively 0,40, 0,46 and 0,83 (Table E.25).

The effect of decreasing the retransmission interval causes the node to transmit less packets. The effect is to reduce the amount of work performed by the node. For the above case consider the bus utilization factors for the test node; these are 0,43, 0,43, and 0,39 respectively (Table E.24), i.e., the channels are occupied : too many retransmitted packets.

Figure 7.16 shows the effect on delay when the interval period is varied over the large range of 5 to 1 000 milliseconds. The result is somewhat surprising in that the delay does not increase at the interval periods in excess of 100 milliseconds. It appears that the number of retransmissions due to either line errors or blocked storage, at the input level of 750 packets/sec., is minimal compared to the number sent along each link.

Again, examination of the line utilization factors confirm the above (see Table E.26). For the input level 750 packets/sec., retransmission period 1 000 milliseconds, the highest utilization factor recorded is 0,63 for channel seventeen; the remainder lie in

the range of 0,31 to 0,57. Provided sufficient buffering is available in the switching section of the network, and provided mean line error rates are low (a mean of one in 10^5 bits was used), the duration of the retransmission interval has little effect on the response so long as the channel is operated in the stable region, in this case an interval period of greater than 30 milliseconds.

The throughput graphs in Figure 7.17 show that retransmission periods of less than 30 milliseconds or greater than approximately 40 milliseconds cause a drop in the number of packets transmitted by the network.

Summarizing, a retransmission interval that is too short will result in a form of channel instability, causing large packet delays and low network throughput. At the larger interval duration, a slightly lower throughput occurs, caused by packets occupying buffers for longer than is necessary. An optimum retransmission period was found, satisfying both response and throughput indices. The period was found to be 30 milliseconds.

7.3.7 Variation of Bus Rate and Algorithm Period

Two further nodal parameters, the data bus rate and the switch algorithm execution time, are discussed. The algorithm execution time has been assumed to be a constant, even though in

practice it would depend on the amount of data processed. Its effect on nodal performance may then be more clearly demonstrated.

Consider first the mean packet response as a function of the nodal bus rate. Threshold behaviour is observed in Figure 7.18. As the bus capacity is increased the packet response improves, until a point is reached where the response remains constant despite further increases in bus capacity. The point occurs at a capacity of approximately 600 KBPS. The poorer response at the lower bus rates takes place since the service time required for a node to perform the work of transferring a packet from one processor module to another, becomes longer, i.e. packets queued in ICP's and NCP's must wait longer before being serviced. The overall effect is an increase both in the transit response (NCP servicing) and entry responses (ICP servicing).

A rough estimate of the bus capacity required may be calculated on the basis that each of the processor modules be able to transmit and to receive one packet per cycle (see Chapter 6). A cycle was defined to be the time taken for a network processor to transmit a packet of maximum size over a 64 KBPS channel. Effectively, this reduces the nodal bus to a statistical multiplexing facility. The number of packets transmitted by the ICP's is negligible compared to the number handled by the NCP's; recall that virtual circuits are used to control the traffic intensities in the network. With ten NCP's in the test node, a theoretical bus rate

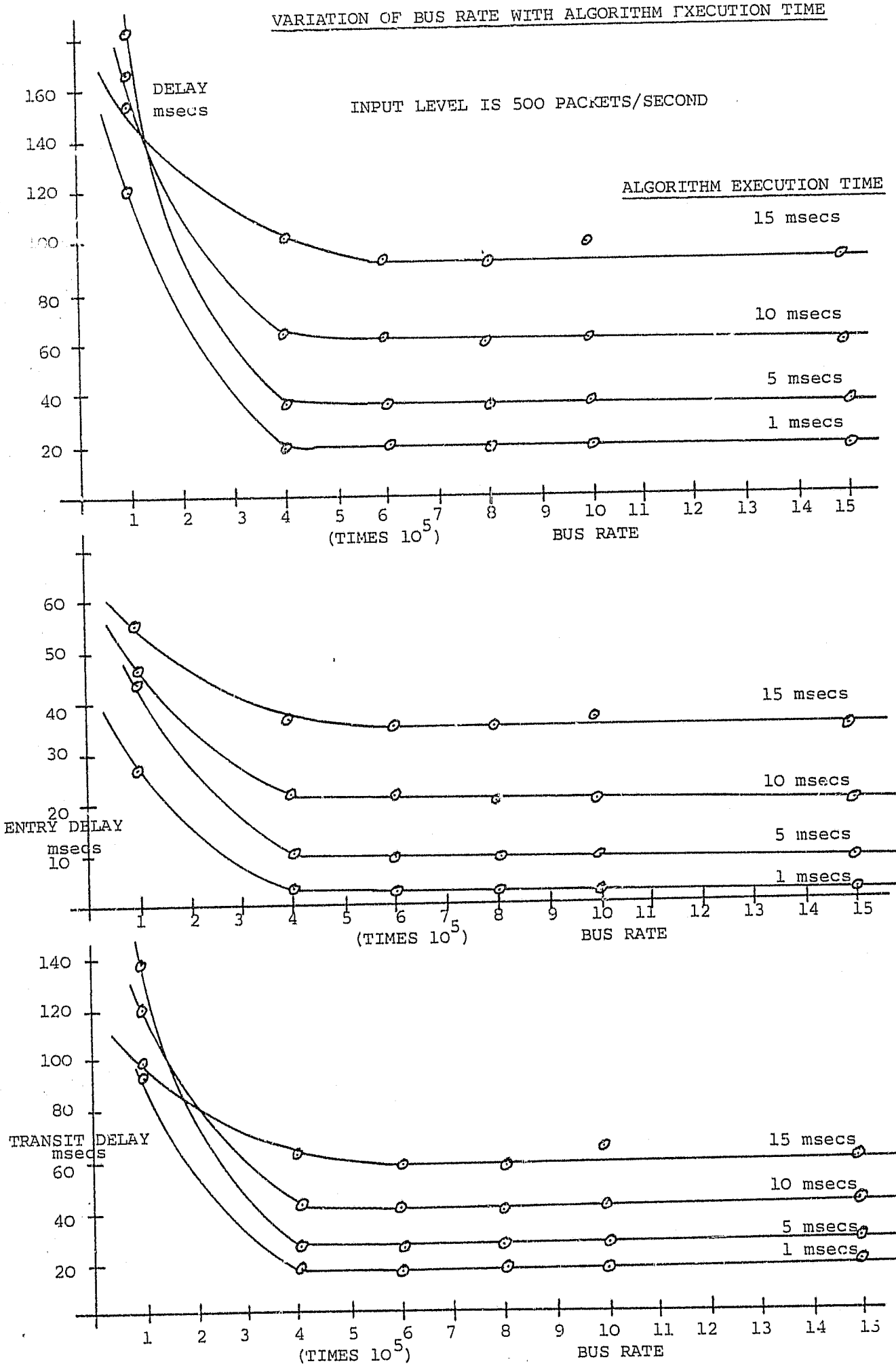

VARIATION OF BUS RATE WITH ALGORITHM EXECUTION TIME
INPUT LEVEL IS 500 PACKETS/SECOND
DELAY msecs
ALGORITHM EXECUTION TIME
15 msecs
10 msecs
5 msecs
1 msecs
160
140
120
100
80
60
40
20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(TIMES 10^5)
BUS RATE
60
50
40
30
20
10
ENTRY DELAY msecs
140
120
100
80
60
40
20
TRANSIT DELAY msecs

Figure 7.18

of 640 KBPS is required. Thus, increasing the bus rate beyond a certain point (600 KBPS as in the simulation) will have little effect, since the NCP and ICP modules are not able to transmit at a higher rate.

It is noted that for these simulations no limit has been placed on the data bus module transfer rate, but that the store-and-forward buffering technique itself restricts the amount of data transferred per unit time interval. The effective internodal circuit rate depends on the amount of buffering, the channel capacity and the response time of the acknowledgement. For above given parameters, a channel throughput limit is reached, after which further increases in the nodal bus capacity have little effect.

The second parameter shown in Figure 7.18 is the switch execution time. It is defined to be the time required to sample, process and output status data for controlling nodal operation. In general the algorithm period should not be of greater duration than the cycle time, otherwise the switch module will become a bottleneck. A decrease in the algorithm period results in a corresponding decrease in packet response time tending to a limit as the period approaches the value of one millisecond. In Figure 7.19, the total response has been plotted against the logarithms of algorithm duration time, at the bus rate of 1,5 MBPS. The graph indicates the limit the response tends to, in this case 16 milliseconds.

VARIATION IN DELAY vs. EXECUTION TIME

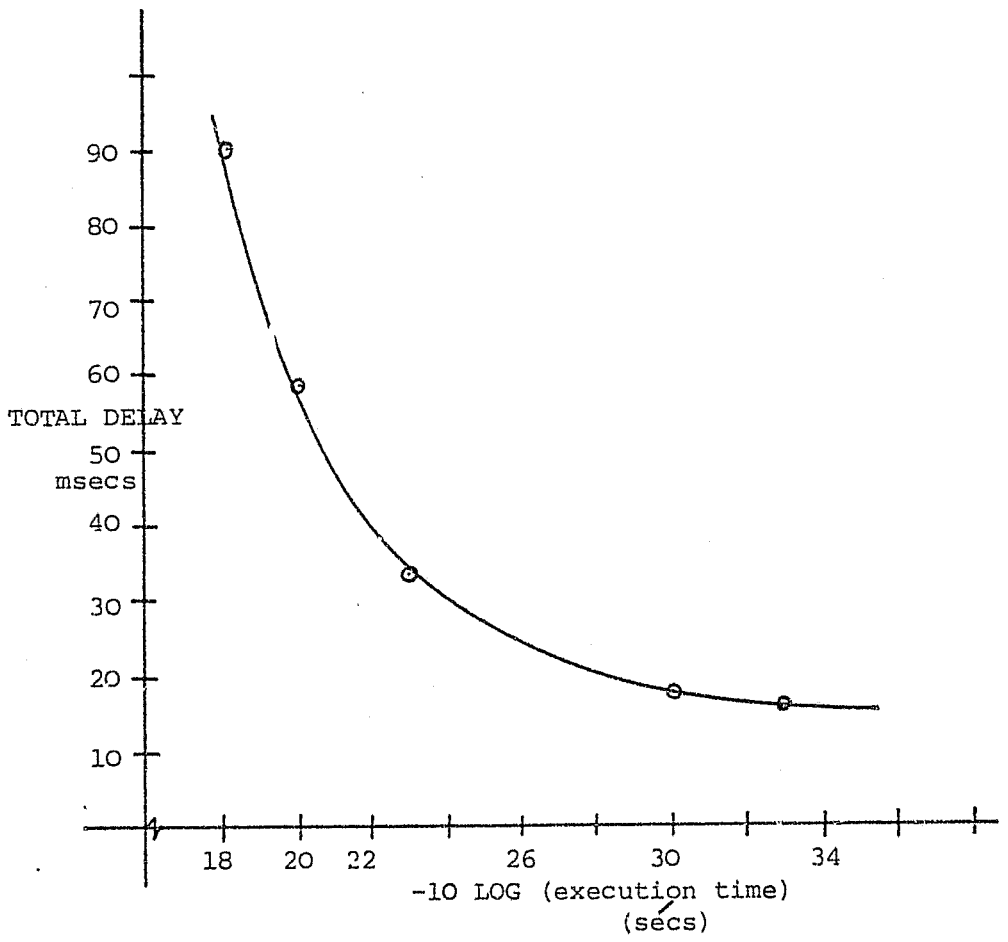


Figure 7.19

Two plots of recorded throughput versus bus rate, for algorithm periods of 5 and 15 milliseconds, are shown in Figure 7.20. The same characteristics are found to occur as for delay. Bus rates of less than 400 KBPS result in a marked decrease in throughput, as does an increase in the algorithm period.

The bus utilization factors are indicative of the activity taking place on the data bus when the bus capacity is reduced. For example, at the algorithm duration period of 5 milliseconds, the utilization factors recorded for the test node at bus rates of 100, 600, 1000, and 1500 KBPS are 0,99, 0,37, 0,23, and 0,15, respectively (Table E.28). These figures clearly illustrate how performance indices became degraded. At the bus rate of 100 KBPS the bus is 99 percent utilized on average, causing a substantial bottleneck in nodal operation.

An examination of the line utilization factors shows an increased line activity with decrease in bus rate. If the data bus is unable to service packets in the NCP's at a sufficient rate, blocking of packets at the destination buffers occurs, causing an increase in the number of retransmissions. The result is a reduction of the effective internodal circuit rate.

The above indicates the importance of keeping the algorithm duration time to a minimum. Due to the repetitive nature of this aspect of the switch the algorithm may be implemented in hardware,

RECORDED THROUGHPUT vs. VARIATION IN BUS RATE

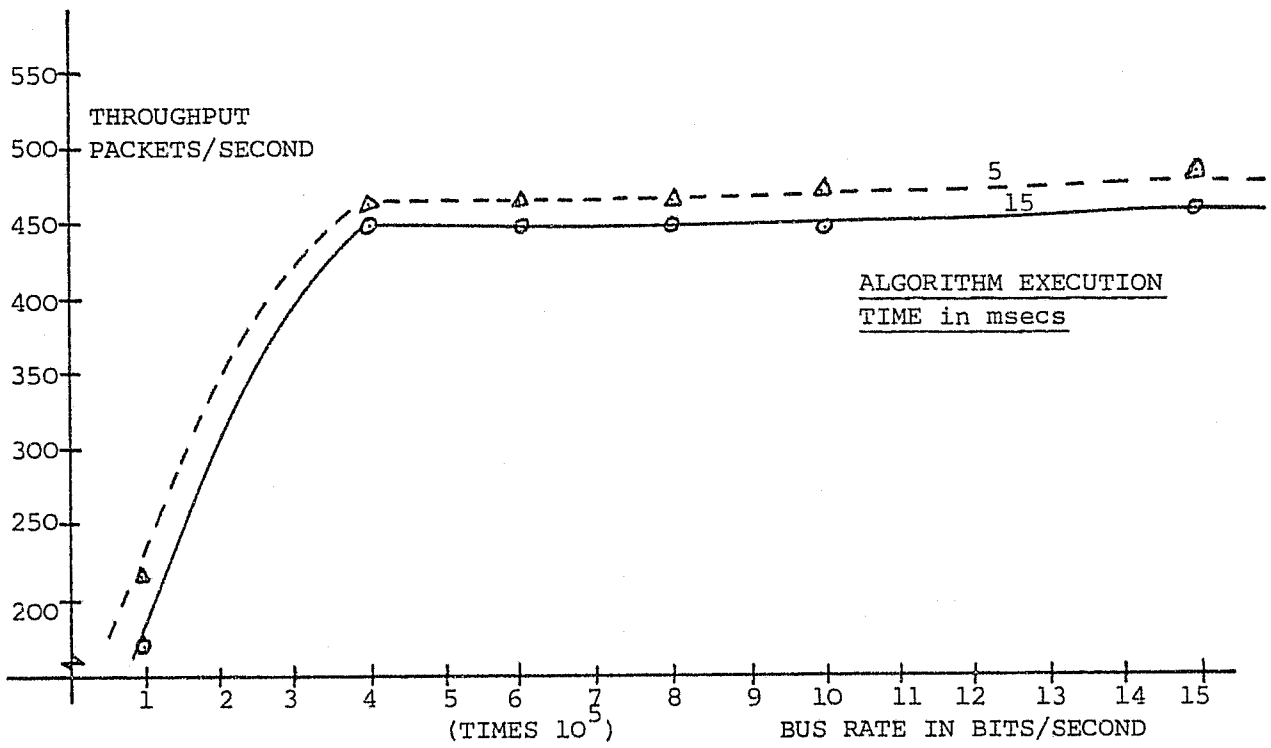


Figure 7.20

which should result in faster execution than if coded in software. The bus rate, too, should be high enough so as not to cause the bus to become a bottleneck.

7.3.8 Nodal Capacity

For the network simulation the mean packet response has been plotted for algorithm duration parameters of one and ten milliseconds, as a function of the input level in Figure 7.21. In addition an analytical approximation has been shown.

The analytical model is based on a set of interconnected M/M/1 queues. Kleinrock's Independence Assumption has been used, i.e., the length of a packet is regenerated at each node as it passes through the network. The model equations are derived in Kleinrock.(21)

The graphs in Figure 7.21 show that the plot based on a one millisecond algorithm duration approaches that of the analytical model. Certain aspects, such as the estimated packet processing time in the simulation model, are not taken into account. In addition, the analytical model approximates a node by a simple M/M/1 queue, whereas in fact it is far more complex consisting of a number of interconnected queues. This implies that the mean packet delay at a given input load will be less as derived by the analytical model. Use of a longer algorithm duration (10 milliseconds) results in a poorer response, as was encountered for the nodal simulation.

VARIATION OF ALGORITHM EXECUTION TIME

NETWORK SIMULATION

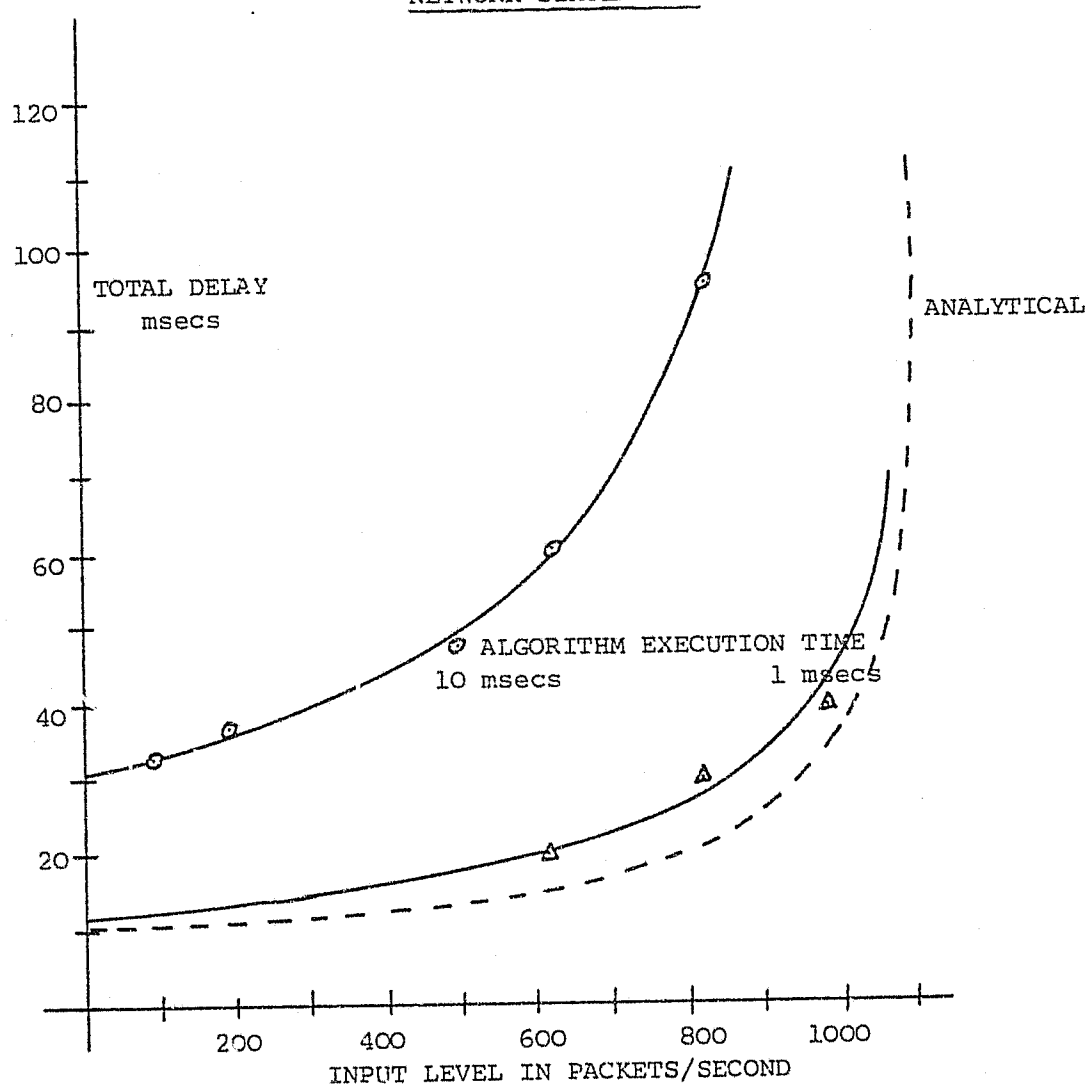


Figure 7.21

Figure 7.22 shows similar plots to those of Figure 7.21, but making use of the nodal simulation. An additional variation has been introduced. Whereas before, a double set of channels interconnected the test node with each of its neighbours (referred to as the double channel case), a simulation has also been performed wherein the test node is linked to each of its neighbours via only a single channel. In effect the number of NCP's in the test node has been reduced from ten to five. This has been done to gain insight into network performance as a function of the number of processors in the node. It is important to note that traffic distributions used were identical for both the single and double channel cases.

In Figure 7.22 the analytical model again shows a fair approximation to the simulated results, for algorithm durations of one millisecond. This model was in fact applied to the ARPANET and compared with measurements conducted on the network. The model was found to be able to give reasonable predictions for packet delays.[21]

A figure for nodal capacity must be considered in terms of the utilization of the nodal resources. When a resource becomes utilized such that $p > 0,70$, then it will tend to become a bottleneck for the rest of nodal operation. The contents of Table 7.3 are instructive in this regard. In the single channel case the nodal links have become congested, whilst the bus utilization

NUMBER OF CHANNELS vs. ALGORITHM EXECUTION TIME

NODAL SIMULATION

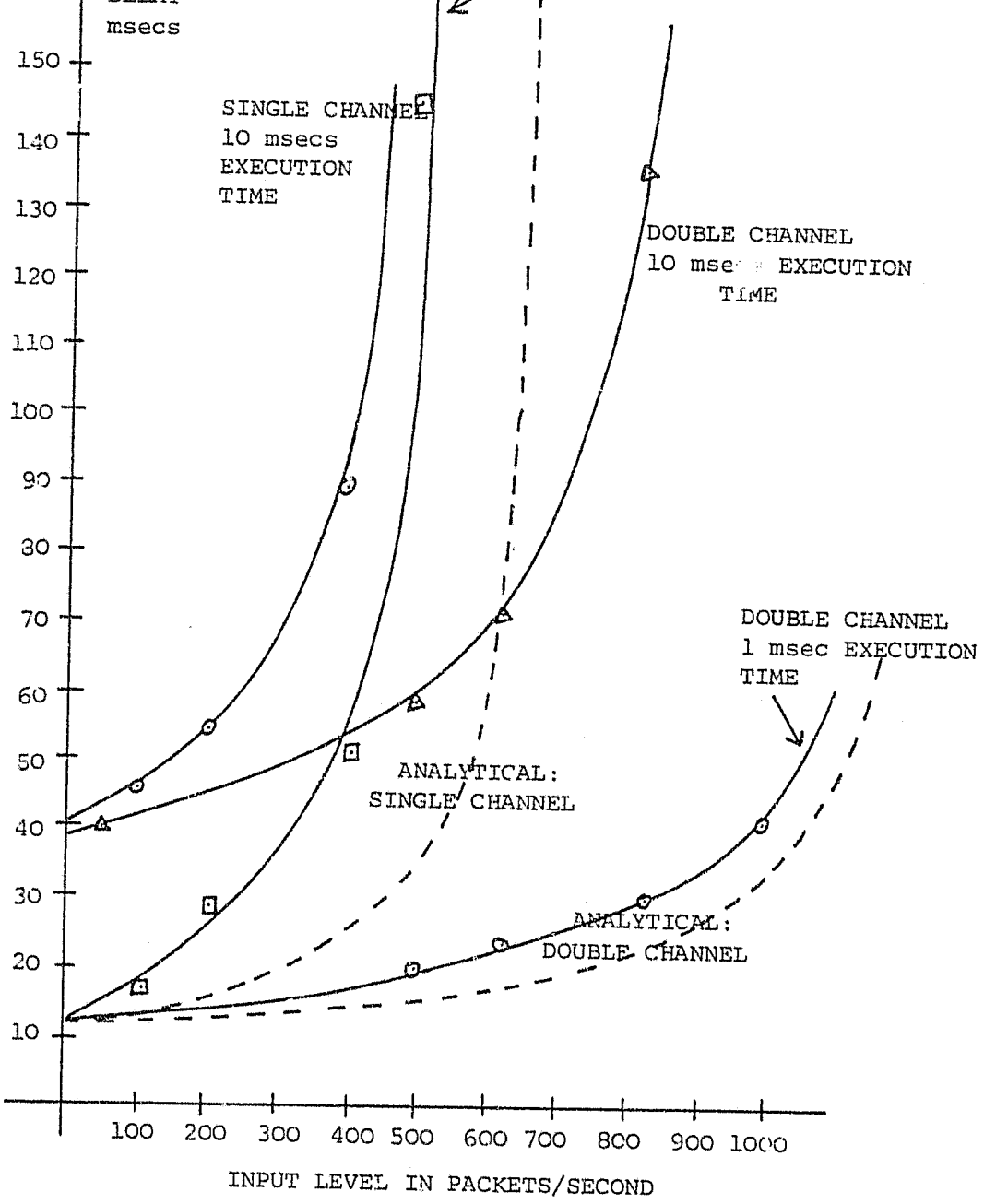


Figure 7.22

factor is only 0,37. In the double channel case both the links and the bus are used to full capacity - with utilization factors for resources greater than 0,70.

It is thus possible to drive the nodal unit at full capacity, provided the bus rate and algorithm duration are at 640 KBPS and one millisecond respectively. By full nodal capacity is thus meant using the resources at up to a 0,70 utilization factor. A proviso is that traffic distributions be more or less uniform.

A change in the number of processors per node will require a repetition of the simulations similar to those encountered in this chapter, to determine the required bus rates and algorithm duration times for optimum nodal operation, given the channel capacity and traffic distribution parameters.

SINGLE CHANNEL CASE (TABLE E.32, E.33)
INPUT LEVEL 500 PACKETS/SEC, ALG DURATION 1 msec
BUS UTILIZATION: 0,37
CHANNEL (UTILIZATION):
1(0,96) 3(0,95) 5(0,93) 7(0,96) 9(0,94)
DOUBLE CHANNEL CASE (TABLE E.31)
INPUT LEVEL 1 000 PACKETS/SEC, ALG DURATION 1 msec
BUS UTILIZATION: 0,84
CHANNEL (UTILIZATION):
1(0,76) 2(0,70) 3(0,85) 4(0,76) 5(0,71)
6(0,71) 7(0,73) 8(0,74) 9(0,65) 10(0,66)

Table 7.3 Channel Utilizations

7.4 Review

The purpose of this chapter has been to study certain of the characteristics of a defined packet switching network. The performance indices concentrated on were those of response and throughput. A dynamic priority allocation technique was studied as a possible local flow control measure. The nodal parameters of retransmission period, data bus rate and switch algorithm execution duration were examined. The simulations have shown that the network structure defined in the previous chapters, is able to transport messages between source and destination host machines. The coordination between the various protocols and measures is sufficient for satisfactory data transmission. In each of the performance tests conducted it has been attempted to explain the underlying mechanisms contributing to the resultant behaviour.

CHAPTER 8

CONCLUSIONS

8.1 Discussion

The preceding chapters have outlined the system design of a data adaptor unit. The modules constituting the unit and their functions, the protocols and the flow control measures for regulating traffic flow were dealt with.

Three types of modules were defined for carrying out the communication functions in a node. The interface processor was used to interface user devices to the network. Nodes were interconnected by means of the network processors. The switching processor was used to regulate the flow of packets within the node. It did this by sampling the status of the interface and network processors, and then issuing commands so that packet exchanges in the node could be carried out in orderly fashion.

An examination of the delay and throughput characteristics of packet-switch based networks showed that flow control measures are needed to prevent congestion. A global flow control technique was implemented using an end-to-end protocol. At the local level, transit traffic was granted priority over that of entry traffic.

Further features implemented to aid in traffic flow control

included the node-to-node protocol and the partitioning of packet buffers along topological lines. The node-to-node protocol was designed to provide for only incorrect packet retransmission. It allowed for the efficient usage of network channels by superimposing a number of logical paths onto a single physical link.

The algorithms implemented in the switch processor for the sorting, switching, and formatting of status data were described. It was shown that such a technique provides a means both for resolving bus access problems and for controlling data flow within the node.

The performance of the data adaptor unit was evaluated as part of a network. The response and throughput indices were determined in simulation tests. The purpose of these tests was to determine the capability of the adaptor for sharing high speed communication channels amongst a number of data transmitting devices. The simulation results also showed the effect that certain parameters, such as bus rate and retransmission interval, have on nodal performance.

8.2 Conclusions

1. The combination of protocols, interprocessor signals, and switch algorithmic control, as incorporated in each of the

data units constituting the network were able to function as a system for the transmission of data packets.

2. The network was able to function in a decentralized mode.

3. Simulation results indicated that performance degradation of the network occurred primarily due to the congestion experienced on the internodal links. The decrease in packet response was a result of the blocking at destination network processors, and an increase in the number of retransmissions that subsequently took place. A possible solution is to incorporate more of the available pcm channels for communication between nodes.

4. Tests wherein more buffers for the network processors were provided showed no improvement in the mean packet delay; increased queue lengths resulted in longer waiting times before service.

5. Throughput of th node may be limited by too slow data bus and switch algorithm execution rates. In this event, congestion in the nodal channels also resulted.

6. A chart giving an indication of the performance of a number of networks, including that of a network based upon the hypothetical data adaptor, is shown in Table 8.1 (Different line capacities, packet sizes, line utilization factors, etc.,

of the networks makes direct comparison difficult). It is found that the performance of the hypothetical network is similar to that of the operational examples.

7. It is observed that the traffic utilization factors of all four operational networks are in the region 0,07 to 0,40. The networks are purposely kept at low utilization to minimize the packet queueing times, thus ensuring a fast response. This is in agreement with the simulation results obtained, i.e., packet networks must be operated at low traffic utilization to avoid excessive resource-sharing conflicts.

8. The throughput of a packet network is determined by its available channel capacities, and the ability to utilize those channels at a percentage of the capacity (40 to 60 percent), without packet delays becoming excessive. Simulations indicated that the data adaptors were able to achieve a mean line utilization of 0,39 for its 64 KBPS channels without delays becoming unbounded.

9. A comparison of response times may be derived by considering the time for a packet to travel a unit path length. The following results are obtained.

NETWORK	PROCESSOR	PACKET SIZES	LINE CAPACITY	MEAN PATH LENGTH	LINE UTILIZATION	RESPONSE
ARPANET RESOURCE-SHARING 56 NODES, 74 LINKS	HONEYWELL DDP-516 HONEYWELL H-316	MAX 1008 BITS OVERHEAD 168 BITS MEAN 218 BITS	50 KBPS	3,3 LINKS	TRM CAPACITY 0,07 BUSIEST LINE 0,48	93 msecs ROUND TRIP
GE INFORMATION SERVICES COMMERCIAL TIME-SHARING 16 CENTRAL CONCENTRATORS 2 SWITCHES	HONEYWELL 4020 DIGINET 1600 (SWITCHES)	MAX 168 CHARS OVERHEAD 12 CHARS	50 KBPS	-	-	ROUND TRIP 400 msecs USA 800 msecs INTERNATIONAL
SITA COMMERCIAL NETWORK FOR AIRLINES 9 COMPUTER CENTRES 15 LINKS	PHILIPS PS-714 UNIVAC 418-II UNIVAC 418-III	A) 30 CHARS INQUIRY 120 CHARS RESPONSE B) 200 CHARS (TELE-PRINTER & COMP.) MAX 240 CHARS	4800 BPS	-	LINE CAPACITY 0,30 to 0,40	END-TO-END 700 msecs
TYMNET COMMERCIAL TIME-SHARING 150 NODES 260 LINKS	VARIAN 620 i VARIAN 620 L	MAX 64 CHARS OVERHEAD 16 BITS AVG 100 BITS	2400 BPS 4800 BPS 9600 BPS	2,5 LINKS	MEAN TRAFFIC CAPACITY 0,20	END-TO-END 300 msecs
HYPOTH. NETWORK SIMULATION RUN ONE 5 NODES 6 LINKS	PROPOSED DATA ADAPTOR	MAX 500 BITS OVERHEAD 70 BITS AVG 400 BITS	64 KBPS	1,12 LINKS	MEAN 0,13 / MAX 0,23	END-TO-END 37 msecs
					0,30 / 0,48	48 msecs
					0,39 / 0,65	60 msecs
					0,46 / 0,83	20 msecs

Table 8.1 Performance of Networks

NETWORK	RESPONSE	LINE UTILIZATION	LINE CAPACITY
ARPANET	14 msecs	0,07	50 KBPS
TYMNET	120 msecs	0,13	4 800 BPS
HYPOTHETICAL NETWORK	18 msecs	0,46	64 KBPS, 1 msec ALGORITHMIC PERIOD
HYPOTHETICAL NETWORK	33 msecs	0,13	64 KBPS, 10 msec ALGORITHMIC PERIOD

Table 8.2 Network Response Times

The non-linear characteristic of the line utilization function makes direct comparison difficult. ARPANET and the simulation results are similar. The response values attained by the proposed data adaptor may be considered sufficient for interactive processing to be performed using a network of such units.

10. By comparison of the simulated network with operational networks, it is concluded that the data adaptor proposed has the response and throughput capability to serve as the basis for a network dedicated to handling data traffic.

11. It is noted that the hypothetical system has a totally different architecture to that of the practical systems (which are based on minicomputer architectures) so that performance comparisons show only that the data adaptor is able to utilize network channels at a level comparable to that of existing systems.

12. It is realized that problems may be encountered in the hardware implementation of the data adaptor. The architecture makes use of two separate address and data busses. Conflicts arising from the simultaneous addressing of a given processor unit via the two bus systems must be resolved - in hardware.

8.3 Future Developments

This dissertation provides the basis for implementation of a packet switched network. Further development in certain areas might be worthwhile. Suggestions for work, both at the nodal and network levels, is contained.

8.3.1 Implementation of the Data Adaptor

1. Development concerning the format the inter-module signals would take is necessary. These might take the form of a command word, expressing source and destination processor addresses, and a code signifying the operation that is to take place.

2. It is suggested that the switch algorithm be implemented in hardware, or perhaps bit-slice technology to meet the stringent algorithm execution period requirements. The essentially repetitive nature of the algorithm makes this possible.

3. An additional switch processor might be needed for reliability purposes, and to take care of functions such as addressing, logical link establishment, and the execution of diagnostic routines to check and to recover from fault conditions.

4. The network processors may, for the most part, be implemented in hardware. If use is made of conventional bit-oriented protocols such as SDLC, internodal packet transmission may be realized by use of integrated circuits such as the Western Digital Micro Packet Interface, which contains both the physical and link control procedures.

5. The interface processor implementation requires a processing unit together with general purpose input/output integrated circuits, so as to enable the node to be connected to a variety of host machines with different interface standards.

8.3.2 Alternate Nodal Architecture

1. The use of common memory rather than a data bus for packet transfer between processors may result in a higher nodal throughput. Use of such memory will require the solution to a number of problems specific to multiprocessor design, e.g., resolution of conflict between simultaneous memory accesses, adequate storage management techniques.

2. The simulation program may be used to test different strategies other than the priority differentiation technique. Different algorithms may be experimented with to test local flow control measures.

8.3.3 Developments at the Network Level

1. The possibility of incorporating adaptive routing exists. The algorithm would be executed by the switch processor, which is ideally placed to examine the status of the nodal channels. The main problem is to execute the algorithm at a sufficiently high rate, as a large number of additions and multiplications are necessary. A high speed mathematics unit will be required.

2. The concept of providing links on a dynamic basis, subject to varying traffic demands, may be looked into. The establishment of a link may take place by one node dialling another, using the existing facilities in the telephone system. A topology may be configured to suit traffic conditions.

APPENDICES

A. Array Definitions

B. Simulation Programmes

C. Parameters for the Network Simulation

D. Parameters for the Nodal Simulation

E. Simulation Results

APPENDIX A

ARRAY DEFINITION

This appendix contains a list of the arrays used in the simulation. The coordinates of each array are defined.

Array IOP (I, J)

I = 1 to n, where 'n' is the sum of the number of logical links and virtual circuits simulated.

J = 1 : input port number

J = 2 : source node number

J = 3 : source ICP number

J = 4 : destination node number

J = 5 : destination ICP number

J = 6 : source channel capacity

J = 7 : destination channel capacity

J = 8 : mean message length

J = 9 : mean message interarrival time

J = 10 : number of messages generated

(when set to a negative number, no upper limit is placed on the number of messages generated for that logical link).

J = 11 : counter; queue of number of messages generated by source device

J = 12 : counter; value indicates number of packets constituting message, excluding 'first packet'

J = 13 : temporary buffer containing generated message length by device, used for device simulation

J = 14 : temporary buffer containing generated message length by device, used for ICP simulation

J = 15 : device - network signalling

IOP (I, 15) = 1 : CALLES

IOP (I, 15) = 2 : RFNS

IOF (I, 15) = 3 : MSGRDY

IOP (I, 15) = 4 : EOM

IOP (I, 15) = 5 : SDEOM (device)

IOP (I, 15) = 9 : TRMNCK

IOP (I, 15) = 10 : SDEOM (network)

IOP (I, 15) = 11 : wait state

IOP (I, 15) = 12 : wait state terminated

CALLES : device-to-network: call establishment

RFNS : network-to-device or destination ICP-to-source ICP: ready for next segment

MSGRDY : device-to-network: message ready

EOM : device-to-network: end of message

SDEOM(device) : network-to-device: an acknowledgement signal indicating to device that message has been successfully transmitted to destination device.

TRMNCK : destination ICP-to-source ICP: no buffering available at destination ICP

SDEOM(network) : destination ICP-to-source ICP: an acknowledgement signal indicating to source ICP that message has been successfully transmitted to destination device.

wait state : source device awaiting buffering at the source ICP

wait state : source device granted buffers

J = 16 : counter: program control for multipacket transmission

J = 17 : counter: program control for the translation of messages to packets

J = 18 : flag: when flag set to 1, indicates to source ICP to transmit next segment

J = 19 : IOP (I, 19) = 1: multisegment buffer release

IOP (I, 20) = 2: multipacket buffer release

J = 20 : segment length storage: decremented by amount equal to packet size as segment is translated to packets

---

Array NODEST (I, J, K)

I = 1 to NONDS, where NONDS is the number of nodes in the network. I is therefore the nodal identifier

J = 1 to NOPCKS where NOPCKS is the maximum number of packets provided for in the network simulation

A number of packets in excess of NOPCKS in transit will result in overflow of simulation arrays; an error message is subsequently generated indicating which array overflowed. The above holds equally for the arrays PACKET (I, J), QLIST (I, J, K) and Q(I, J, K).

K = 1 : packet identifier

K = 2 : processor identifier

K = 3 : 1: interface processor

2: network processor

Array PACKET (I, J)

I = 1 to NOPCKS where NOPCKS is the maximum number of packets provided for in the network simulation

I = packet identifier

J = 1 : logical link or virtual circuit identifier

J = 2 : bit length of packet

J = 3 : PACKET (I, 3) = 0 : logical link

PACKET (I, 3) = 1 : virtual circuit

J = 4 : value recording total number of packets in segment

J = 5 : packet format : data or control

PACKET (I, 5) = 1 data packet

PACKET (I, 5) = 7 data packet belonging to either final or only segment in message in transmission

PACKET (I, 5) = 2 : Buffer access ack

PACKET (I, 5) = 4 : ready for next segment

PACKET (I, 5) = 5 : no buffering available at ICP

PACKET (I, 5) = 6 : message/segment acknowledgement

The final three conditions define control packets from destination to source ICP.

J = 6 channel controller transmission priority value

Array SWITCH (I, J)

I = 1 to NOI where NOI is the maximum number of processors per node

I : processor work space for switching algorithm execution

J = 1 source processor identifier

J = 2 : 1 - interface processor

: 2 - network processor

J = 3 destination processor identifier

J = 4 : 1 - interface processor

: 2 - network processor

J = 5 packet identifier

Array RDNCTR (I)

I = 1 to n where n is the number of logical links simulated

RDNCTR (I) : is a counter recording the number of messages transmitted via the logical link defined by I. The random number generates a maximum of NRDN deviates; it is essential that NRDN not be exceeded in the simulation if error conditions are not to arise. An error message and program termination results if NRDN is exceeded.

Array MSGLNZ (I, J) MSGEXP (I, J)

I = 1 to n where n is the number of logical links

I : logical link identifier

J = 1 to NRDN where NRDN is the maximum number of deviates generated

MSGLNZ (I, J) : recording of message length generated

MSGEXP (I, J) : recording of message interarrival time generated

MSGLNZ is an integer - whilst MSGEXP is a real array. Both are used to record the length and the interarrival time characteristics of a message and are determined in the initialization section of the program, i.e., these are determined beforehand and there is thus no need to call the appropriate subroutine each time a message is generated. Additionally, a proper distribution is formulated by generating a large number of deviates at one time.

Array Q (I, J, K)

I = 1 to NONDS, where NONDS is number of nodes in network

I : nodal identifier

J = 1 to QUS where QUS is the maximum number of packet identifiers that can be stored in the switching processor

K = 1 pointer to QLIST array location as defined by dimension J of QLIST

K = 2 interval dependent priority value

Array QLIST (I, J, K)

I = 1 to NONDS

I : nodal identifier

J = 1 to QUS

K = 1 : source processor identifier

K = 2 : 1 - interface processor

: 2 - network processor

K = 3 : destination processor identifier

K = 4 : 1 - interface processor

: 2 - network processor

K = 5 : packet identifier

K = 6 : partition number

K = 7 : weighting factor, static priority

K = 8 : pointer to Q array locations as defined by dimension J of Q

Array NCPCNT (I)

I = 1 to NONCPS, where NONCPS is number of NCP's in network

I : source network processor identifier

NCPCNT (I) : destination network processor identifier

Array NODPLT (I, J, K)

I = 1 to NONDS

J = 1 to NONDS, where NONDS is the number of nodes in network

I : present nodal identifier

J : destination nodal identifier

K = 1 : partition number

K = 2, 3, ... (ℓ + 1) where ℓ is the maximum number of channels between any two nodes

NODPLT (I, J, K $\neq$ 1) : for K = 2, 3, 4 ... (ℓ + 1) : listing of network processor identifiers

---

Array ERRORG (I, J)

I = 1 to NONCPS where NONCPS is the number of NCP's in network

I = network processor identifier

J = 1 : bit generation value

J = 2 : bit counter value

J = 3 : error flag

Arrays FIELD1 (I, J, K), FIELD2 (I, J, K, 3 (I, J, K)

I = 1 to NONCPS where NONCPS is the number of NCP's in the network

I : network processor identifier

J = 1 to 8

J : field identifier; in this simulation eight field identifiers have been provided for.

FIELD1 (I, J, K) : transmitting field

K = 1 : packet identifier

K = 2 : partition number

K = 3 : least significant bit position

K = 4 : most significant bit position

K = 5 : buffer release synchronization flag

FIELD2 (I, J, K) : receiver field

K = 1 : least significant bit position

K = 2 : most significant bit position

K = 3 : field command bit position

FIELD3 (I, J, K) : retransmission field

K = 1 : flag to initiate time-out interval

K = 2 : retransmission code execution flag control

K = 3 : flag to indicate retransmission of packet necessary, i.e., time-out interval completed before acknowledgement signal received

Arrays BFPOOL (I, J, K), BFPOLR (I, J, K)

I = 1 to NONCPS where NONCPS is number of NCP's in network

I = network processor identifier

J = 1 ... ℓ where ℓ is number of buffer blocks storage in receiver or transmitter section of network processor

K = 1 : packet identifier

K = 2 : partition number

BFPOOL : transmitter storage in NCP

BFPOLR : receiver storage in NCP

Arrays PRTNS1 (I, J, K), PRTNS2 (I, J, K)

I = 1 to NONCPS where NONCPS is number of NCP's in network

I : network processor identifier

J = 1 to 8

Eight fields per NCP are simulated

PRTNS1 (I, J, K) : transmitter partitioning

K = 1 : partition number

K = 2 : TRMBUF storage location

K = 3 : QBUF storage location

PRTNS2 (I, J, K) : receiver partitioning

K = 1 : partition number (destination NCP)

K = 2 : TRMBUF storage location

K = 3 : QBUF storage location

K = 4 : partition number (present NCP)

K = 5 : flag indicating partition blocked

K = 6 : code execution flag control

Arrays PACTR (I, J, K), PACRC (I, J, K)

I = 1 to NONCPS where NONCPS is number of NCP's in network

I : network processor identifier

J = 1 K = 1 : frame field number

K = 2 : least significant bit position

K = 3 : most significant bit position

K = 4 : packet identifier

K = 5 : partition number

K = 6 : error flag

K = 7 : timer flag control

J = 2 K = 1 to 8 : partition commands

J = 3 K = 1 to 8 : least significant bit field position

J = 4 K = 1 to 8 : most significant bit field positions

Array LINE (I)

I = 1 to NONCPS where NONCPS is number of NCP's in network

I : network processor identifier

LINE (I) : flag : code execution control

LINE (I) = 0 : channel free

LINE (I) = 1 : channel in use

Array IFBUFR (I)

I = 1 to NONCPS where NONCPS is number of NCP's in network

I : network processor identifier

IFBUFR (I) : packet identifier stored in interface buffer of network processor I

Array NONCC (I)

I = 1 to NONDS where NONDS is number of nodes in network

I : nodal identifier

NONCC (I) : bus (data) cycle counter

Array REASSM (I, J)

I = 1 to n where n is number of logical links in network

J = 1 : packet identifier of most recent arrival

J = 2 : number of packets in segment

J = 3 : number of packets assembled

J = 4 : accumulated segment length

J = 5 : program execution control flag

J = 6 : end-of-message execution control

J = 7 : multisegment messages - flag

J = 8 : accumulated destination ICP buffer storage

Arrays DEST (I), SRCIO (I), STATEG (I), MESSAG (I)

I = 1 to n where n is number of logical links in network

The above arrays serve to control the execution of code in the simulation program

---

Array MAPICP (I, J)

I = 1 to NONDS

I : nodal identifier

J = 1 to ℓ where ℓ is the maximum number of interface processors per node

MAPICP : interface processor identifiers

Array MAPNCP (I, J)

I = 1 to NONDS

I : nodal identifier

J = 1 to k where k is the maximum number of networ! processors per node

MAPNCP : network processor identifiers

The above two arrays serve to associate interface and network processors to a given node

---

Arrays PT1ICP (I, J, K), PT2ICP (I, J, K) PT1NCP (I, J, K), PT2NCP (I, J, K)

PT1ICP, PT1NCP } transmit pointers for ICP and NCP respectively

PT2ICP, PT2NCP } queue pointers for ICP and NCP respectively

For PT1ICP and PT2ICP : I = 1 to NOICPS

I : interface processor identifier

For PT1NCP and PT2NCP : I = 1 to NONCPS

I : network processor identifier

J = 1 to m where m is the number of cycles per period

K = 1 : initial pointer

K = 2 : = 1 : interface processor

= 2 : network processor

K = 3 : packet identifier

K = 4 : destination processor identifier

K = 5 : = 1 : interface processor

= 2 : network processor

K = 6 : terminating pointer

K = 7 : = 1 : interface processor

= 2 : network processor

K = 8 : end of cycle indicator

Arrays DATABS (I), SWCONT (I)

I = 1 to NONDS

I : nodal identifier

program execution control arrays

Array PKMUX (I)

I = 1 to NONDS

I : nodal identifier

PKMUX (I) : number of packets multiplexed per cycle in node I

Array DATIMR (I)

I = 1 to NONDS

I : nodal identifier

DATIMR (I) : data bus timing control

Array LGLINK (I, J)

I = 1 to n where n is the number of logical links simulated

J = 1 : source processor (ICP) flag

J = 2 : destination processor (ICP) flag

LGLINK (I, J) = 1 logical link waiting for buffer

LGLINK (I, J) = 2 logical link allocated buffers, message transmission proceeds

---

Array STO (I, J)

I = 1 to NONCPS

I : network processor identifier

STO (I, 1) :
STO (I, 2) : tabulation of number of free buffer blocks in NCP I

Array PARTNP (I, J, K)

I = 1 to NONCPS

I : network processor identifier

J = 1 to 8 : number of partitions allowed used in simulation

K = 1 : partition number

K = 2 : = 0 : unblocked partition

: = 1 : blocked partition

Array BUSSEM (I), SEMPH (I)

I = 1 to NONDS, nodal identifier

BUSSEM (I) } switching algorithm execution and data bus packet
SEMPH (I) } transfers synchronization flags

Array SEMPRC (I, J)

I = 1 to NONDS

I : nodal identifier

J = 1 : SOP : start of processing : processor identifier

J = 2 : = 1 : interface processor

= 2 : network processor

J = 3 : = -200 : end of processing

≠ -200 : processing in operation

J = 4 : NOP : next to process : processor identifier

J = 5 : = 1 : interface processor

= 2 : network processor

NB: the word 'process' as used here is meant to indicate the processing done by a module in transmitting packets onto - and receiving packets from the data bus

---

Array BLOCK (I, J)

I = 1 to NOICPS

I : interface processor identifier

J = 1 : number of source blocks used

J = 2 : number of single packet blocks used

J = 3 : number of multipacket blocks used

J = 4 : number of common blocks used

J = 5 : number of source blocks

J = 6 : number of single packet blocks

J = 7 : number of multipacket blocks

J = 8 : number of common blocks

Array ALLOC (I, J, K)

I = 1 to NOICPS

I : interface processor identifier

J = 1 to N where N is number of locations for storing addresses of logical links allocated buffers

K = 1 : logical link number

K = 2 : type

K = 3 : number of blocks of type used

K = 4 : number of blocks of common used

Array WAITLN (I, J, K)

I = 1 to NOICPS

I : interface processor identifier

J = 1 to ℓ : number of locations for reserving buffer requests allowed for

K = 1 : logical link number

K = 2 : number of buffers required

K = 3 : clock value at time of request

K = 4 : type of buffers required

K = 5 : priority value

Array LMA (I)

I = 1 to NOICPS

LMA (I) : 'waiting state' algorithm execution flag

Arrays TIMERZ (I), EVENT (I), SIGNTM (I), IDTINV (I)

I = 1 to p where p = (4 x N) + NONCPS + (2 x NONDS) + 16

N : number of logical links

NONCPS : number of network processors simulated

NONDS : number of nodes simulated

TIMERZ (I) : interval generated by simulation event

EVENT (I) : queue of events

IDTINV (I) : queue of labels of events

SIGNTM (I) = 2 : wait for interval to elapse

= 3 : interval elapsed

APPENDIX B

DATA NETWORK SIMULATION PROGRAMS

This appendix provides the program listings and flowcharts for the simulation model. Reference to Appendix A must be made for descriptions of the arrays and their subscripts.

The simulation program has been written in modular fashion so that the various blocks constituting the model can be separated and described. The procedure followed here will be to identify the modules and blocks and to provide flowcharts at the lowest level of blocks.

The highest levels consist of an initialization and an execution block. The initialization block is responsible for reading the input data parameters, and initializing the array contents. The execution block constitutes the model simulation, and is comprised of nine main modules. These are the packet injection facility, the source host- and destination host-network communication, the node-to-node communication, the data bus transfers, the switch processor operation, the clock, the buffer management allocation and the buffer management reservation modules. These are illustrated in Figures B-1 and B-2. It is assumed that Chapter 3, describing the overall program structure and operation, has been read. In particular it is necessary to understand the concept of signalling used between the clock and the remaining modules.

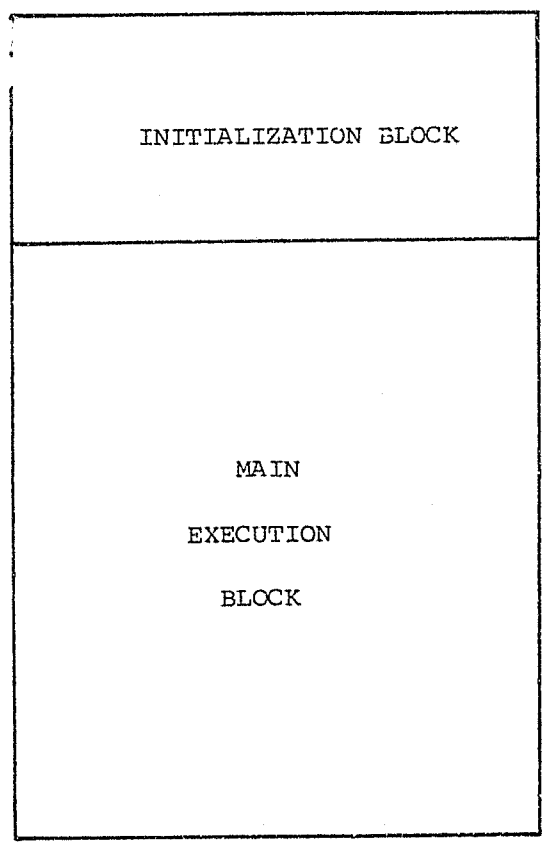


Figure B.1 Main Simulation Structure

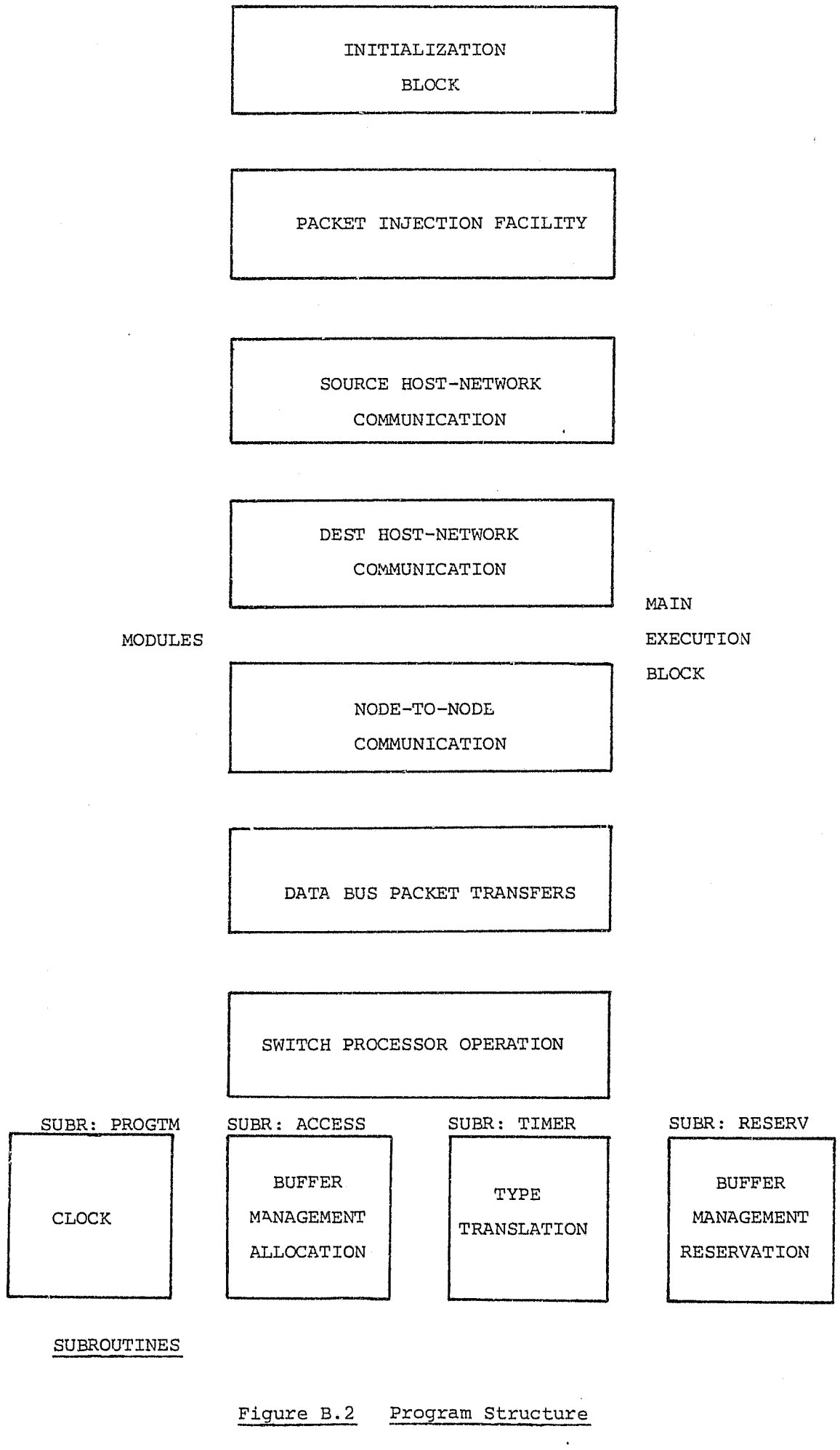
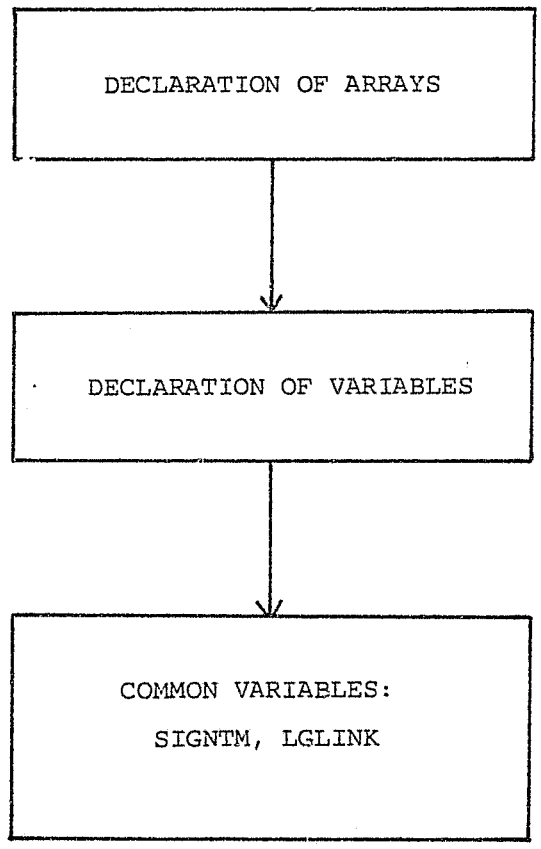


Figure B.2 Program Structure

BLOCK INITIALIZATION



See Appendix A for array descriptions and dimensions

Figure B.3 Array and Variable Declarations

BLOCK INITIALIZATION

READ IN FOLLOWING VARIABLE CONTENTS:

N	PSZND	NMAXIC
NUMBER	NRDN	NMAXIN
SCAN	NRRN	QUS
BUFBLK	MNOBFS	NVC
NSOURC	BITLNG	SWPROC
SGMENT	NOI	MUXRT
NONDS	MAXCH	TXMAX
NOICPS	NOPCKS	PXMAX
NONCPS	CMAX	NPZ
NODPRC	PMEANC	

See Chapter 3 for description of these parameters

READ LOGICAL LINK DESCRIPTORS
VIRTUAL CIRCUITS : IOP

READ PROCESSOR PARTITION MATRIX : PRTNS1
READ NODE PLOT DATA : NODPLT
READ CHANNEL PROCESSOR DATA : NCPCNT
READ MULTIPLEXING MATRIX : PKMUX
READ NODE TO ICP MAPPING : MAPICP
READ NODE TO NCP MAPPING : MAPNCP
READ CHANNEL DATA 1 ...(n) : NODPLT
PLACE SENDER PARTITION : PRTNS1 CONTENTS INTO
RECEIVER PARTITION : PRTNS2 LOCATIONS

WRITE ABOVE DATA FOR VERIFICATION

Figure B-4 Input Data

```
L.001 C
L.002 C
L.003 C     DATA  NETWORK  SIMULATION
L.004 C
L.005 C     ******************************************************************
L.006 C
L.007       INTEGER IR(100),MSGLNZ(20,100),RDNCTR(20),NRDN
L.008       REAL WK(100),R(100),MSGEXP(20,100)
L.009 C
L.010       REAL TIMERZ(130)
L.011       INTEGER SIGNTM(130),LGLINK(20,2)
L.012 C
L.013 C     MAX NO LOGICAL LINKS+VIRTUAL CIRCUITS: 24
L.014       INTEGER IOP(24,20),VIRTM(24)
L.015 C
L.016 C     MAX NO LOGICAL LINKS: 20
L.017       INTEGER REASSM(20,8),DEST(20),KNTR(20),STATEG(20),MESSAG(20)
L.018       INTEGER CNTR(20),SRCIO(20)
L.019 C
L.020 C     MAX NO "NCPS" IN NETWORK: STORAGE 20 BLOCKS: NO CYCLES=1 : 20
L.021       INTEGER NCPCNT(20),BFPOOL(20,60,2),BFPOLR(20,60,2),IFBUFR(20)
L.022       INTEGER PACTR(20,4,8),PACRC(20,4,8),LINE(20),STO(20,2)
L.023       INTEGER PARTNP(20,8,2),PT1NCP(20,1,8),PT2NCP(20,1,8)
L.024       INTEGER ERRORG(20,3),FIELD1(20,8,5),FIELD2(20,8,4)
L.025       INTEGER FIELD3(20,8,4),PRTNS1(20,8,6),PRTNS2(20,8,6)
L.026       INTEGER DUPLCT(50,20)
L.027 C
L.028 C     MAX NO "ICPS" IN NETWORK: NO CYCLES=1 : 10
L.029       INTEGER DRSICP(10),PT1ICP(10,1,8),PT2ICP(10,1,8)
L.030 C
L.031 C     MAX NO NODES IN NETWORK: 6
L.032       INTEGER NODPLT(6,6,3),SEMPH(6),BUSSEM(6),DATABS(6),SEMPRC(6,5)
L.033       INTEGER DATIMR(6),QLIST(6,50,8),Q(6,50,2),PKMUX(6),INITLZ(6)
L.034       INTEGER SWCONT(6),NODEST(6,200,3),NONCC(6)
L.035 C
L.036 C     MAX NO "NCPS" PER NODE: 10
L.037       INTEGER MAPNCP(6,10)
L.038 C
L.039 C     MAX NO "ICPS" PER NODE: 10
L.040       INTEGER MAPICP(6,10)
L.041 C
L.042 C     MAX NO ("ICPS" + "NCPS") PER NODE: 10
L.043       INTEGER VARTM(10),SWITCH(10,5)
L.044 C
L.045 C     PACKET INFORMATION IN NETWORK
L.046       INTEGER PACKET(100,6)
L.047 C
L.048 C     TIMER LABEL INFORMATION
L.049 C     MAX NO LABELS=130
L.050 C     NO=(4*N)+(NONCPS)+(2*NONDS)+16
L.051 C     ******************************************************************
L.052 C
L.053       INTEGER CALLES,RFNS,EOM,SDEOM,RST,NRB,LINKNR,TNR
L.054       INTEGER ICPNR,ACC,SOM,NSOURC,ST,ICP,LNK,NO,CL
L.055       INTEGER RES,TPE,NFF,NMM,NX,WAITST,WAITFN,BUFBLK
L.056       INTEGER DESTNO,NONCP,PCKNO,PRSNND,PARTON,QS
L.057       INTEGER PKNN,BITLNG,DSSNCP,PCCNO,PAANO,PACCN,PAARTN
L.058       INTEGER FRMFLD,PCCKNO,PRES,QUS,CMAX,SCAN,PSZNO
L.059       INTEGER DIVIS,ZTYP,SGMNO,GAMMA,CNZ,ID,INTERV
L.060       INTEGER NCR,NSOU,NSING,NMUL,QCNTR,SGMENT
L.061       REAL NPZ,NODPRC,SWPROC,CLOCK
L.062 C
L.063       COMMON SIGNTM,LGLINK
L.064       DOUBLE PRECISION DSEED
L.065 C     ******************************************************************
```

```
L.066 C
L.067 C      READ  INPUT  DATA  FOR  NETWORK
L.068 C
L.069 C
L.070        READ (5,95) N,NUMBER,SCAN,BUFBLK,NSOURC,SGMENT,NONDS,NOICPS,NONCPS
L.071        READ (5,96) PSZNO,NRDN,NRRN,MNOBFS,BITLNG,NOI,MAXCH,NOPCKS,CMAX
L.072        READ (5,97) NMAXIC,NMAXIN,QUS,NVC
L.073        SWPROC=0.01
L.074        MUXRT=640000
L.075        TXMAX=0.040
L.076        PXMAX=0.030
L.077        NPZ=0.0
L.078        NODPRC=0.0
L.079        PMEANC=0.50
L.080  95    FORMAT (T2,9I7)
L.081  96    FORMAT (T2,9I7)
L.082  97    FORMAT (T2,4I7)
L.083 C      IOP SOURCE DATA
L.084        NTTLI=N+1
L.085        NTTL=N+NVC
L.086        READ (5,101) ((IOP(I,J),J=2,10),I=1,NTTL)
L.087        DO 100 I=1,NTTL
L.088        J=1
L.089        IOP(I,J)=I
L.090  100   CONTINUE
L.091  101   FORMAT(T2,9I7)
L.092 C      READ PROCESSOR PARTITION DATA
L.093        READ (5,108) ((PRTNS1(I,K,1),K=1,8),I=1,NONCPS)
L.094  108   FORMAT (T2,8I7)
L.095 C      READ NODE PLOT DATA
L.096        READ (5,111) ((NODPLT(I,J,1),J=1,NONDS),I=1,NONDS)
L.097  111   FORMAT (T2,6I7)
L.098 C      READ CHANNEL PROCESSOR DATA
L.099        READ (5,114) (NCPCNT(I),I=1,8)
L.100        READ (5,115) (NCPCNT(I),I=9,16)
L.101  114   FORMAT (T2,8I7)
L.102  115   FORMAT (T2,8I7)
L.103 C      NUMBER OF PACKETS MULTIPLEXED PER CYCLE FOR INDIVIDUAL NODES
L.104        READ (5,118) (PKMUX(I),I=1,NONDS)
L.105  118   FORMAT (T2,6I7)
L.106 C      NODE TO ICP MAPPING
L.107        READ (5,122) ((MAPICP(I,J),J=1,NMAXIC),I=1,NONDS)
L.108  122   FORMAT (T2,2I7)
L.109 C      NODE TO NCP MAPPING
L.110        READ (5,126)  ((MAPNCP(I,J),J=1,NMAXIN),I=1,NONDS)
L.111  126   FORMAT (T2,5I7)
L.112 C      READ IN CHANNEL DATA
L.113        READ (5,130) ((NODPLT(I,J,2),J=1,NONDS),I=1,NONDS)
L.114  130   FORMAT (T2,6I7)
L.115 C      READ IN CHANNEL DATA (2)
L.116        READ (5,134) ((NODPLT(I,J,3),J=1,NONDS),I=1,NONDS)
L.117  134   FORMAT (T2,6I7)
L.118 C      RECEIVER PARTITION BUFFERING
L.119        DO 139 K=1,NONCPS
L.120        MLG=NCPCNT(K)
L.121        DO 138 M=1,8
L.122        PRTNS2(MLG,M,1)=PRTNS1(K,M,1)
L.123        PRTNS2(K,M,4)=PRTNS1(K,M,1)
L.124  138   CONTINUE
L.125  139   CONTINUE
L.126 C
L.127 C      ***************************************************************
```

```
L.128 C
L.129         WRITE (6,103)
L.130         WRITE (6,104)
L.131         WRITE (6,102) ((IOP(I,J),J=1,10),I=1,N)
L.132   102   FORMAT(T2,10I7)
L.133   103   FORMAT(T10,'SRCNOD SRCICP DSTNOD DSTICP SRCCPC DSTCPC')
L.134   104   FORMAT(T3,'LINENO',T52,'MLNGTH MARRVL NO.MSG')
L.135         WRITE (6,105)
L.136   105   FORMAT( T2,///)
L.137         WRITE (6,160)
L.138   160   FORMAT (T2,'  INTERNAL PACKET GENERATION')
L.139         WRITE (6,161) ((IOP(I,J),J=1,10),I=NTTLI,NTTL)
L.140   161   FORMAT (T2,10I7)
L.141         WRITE (6,162)
L.142   162   FORMAT (T2,///)
L.143         WRITE (6,109)
L.144   109   FORMAT (T2,'  PROCESSOR PARTITION MATRIX')
L.145         WRITE (6,110) ((PRTNS1(I,K,1),K=1,8),I=1,NONCPS)
L.146   110   FORMAT (T2,8I7)
L.147         WRITE (6,106)
L.148   106   FORMAT (T2,///)
L.149         WRITE (6,112)
L.150   112   FORMAT (T2,'  NODE PLOT MATRIX, PARTITIONS')
L.151         WRITE (6,113) ((NODPLT(I,J,1),J=1,NONDS),I=1,NONDS)
L.152   113   FORMAT (T2,6I7)
L.153         WRITE (6,107)
L.154   107   FORMAT (T2,///)
L.155         WRITE (6,116)
L.156   116   FORMAT (T2,'  NCP TO NCP CONNECTION')
L.157         WRITE (6,117) (NCPCNT(I),I=1,16)
L.158   117   FORMAT (T2,16I4)
L.159         WRITE (6,140)
L.160   140   FORMAT (T2,///)
L.161         WRITE (6,119)
L.162   119   FORMAT (T2,'  PACKET MULTIPLEXING ON BUS')
L.163         WRITE (6,120) (PKMUX(I),I=1,NONDS)
L.164   120   FORMAT (T2,6I7)
L.165         WRITE (6,121)
L.166   121   FORMAT (T2,///)
L.167         WRITE (6,123)
```

```
L.168   123  FORMAT (T2,'  MAPPING: NODE TO ICP')
L.169        WRITE (6,124) ((MAPICP(I,J),J=1,NMAXIC),I=1,NONDS)
L.170   124  FORMAT (T2,2I7)
L.171        WRITE (6,125)
L.172   125  FORMAT (T2,///)
L.173        WRITE (6,127)
L.174   127  FORMAT (T2,'  MAPPING:NODE TO NCP')
L.175        WRITE (6,128) ((MAPNCP(I,J),J=1,NMAXIN),I=1,NONDS)
L.176   128  FORMAT (T2,5I7)
L.177        WRITE (6,129)
L.178   129  FORMAT (T2,///)
L.179        WRITE (6,131)
L.180   131  FORMAT (T2,'  ROUTING DATA: FIRST CHANNEL')
L.181        WRITE (6,132) ((NODPLT(I,J,2),J=1,NONDS),I=1,NONDS)
L.182   132  FORMAT (T2,6I7)
L.183        WRITE (6,133)
L.184   133  FORMAT (T2,///)
L.185        WRITE (6,135)
L.186   135  FORMAT (T2,'  ROUTING DATA: SECOND CHANNEL')
L.187        WRITE (6,136) ((NODPLT(I,J,3),J=1,NONDS),I=1,NONDS)
L.188   136  FORMAT (T2,6I7)
L.189        WRITE (6,137)
L.190   137  FORMAT (T2,///)
L.191        WRITE (6,151) N,NUMBER,SCAN,BUFBLK,NSOURC,SGMENT,NONDS,NOICPS,NONCPS
L.192        WRITE (6,152) PSZNO,NRDN,NRRN,MNOBFS,BITLNG,NOI,MAXCH,NOPCKS,CMAX
L.193        WRITE (6,153) NMAXIC,NMAXIN,QUS,NVC
L.194   151  FORMAT (T2,9I7)
L.195   152  FORMAT (T2,9I7)
L.196   153  FORMAT (T2,4I7)
L.197        WRITE (6,154)
L.198   154  FORMAT (T2//)
L.199   150  CONTINUE
L.200 C
```

BLOCK: INITIALIZATION

INSERT SEED FOR RANDOM NUMBER GENERATOR: DSEED - DOUBLE PRECISION;

FOR EACH LOGICAL LINK DO: INITIALIZE THE FOLLOWING PGM CONTROL VARIABLES:

STATEG(I) = 2	VIRTM(I) = 1	MESSAG(I) = 1
SRCIO(I) = 1	CNTR(I) = 1	DEST(I) = 1
RDNCTR(I) = 1	REASSM(I,1) = 1	

END;

- LOGICAL LINKS: 1...N (I)
- VIRTUAL CIRCUITS NTTLI = N+1

: NTTLI, NTTL (I)

INITIALIZE ALL IOP ARRAY LOCATIONS FOR LOGICAL LINK AND VIRTUAL CIRCUIT DESCRIPTORS TO ZERO: IOP(I,11) TO IOP(I,20);

FOR EACH VIRTUAL CIRCUIT DO: INITIALIZE PGM CONTROL VARIABLE:

VIRTM(I) = 1;

END;

INITIALIZE	SUBR	ACCESS;
INITIALIZE	SUBR	RESERV;
INITIALIZE	SUBR	PROGTM;

FOR EACH NCP DO (220):
INITIALIZE MESSAGE BUFFER IFBUFR(I) = 0;
INITIALIZE PGM CONTROL VARIABLE LINE(I) = 0;

SET MEAN BIT ERROR RATE FOR LINES ERROG(I, 1);
INITIALIZE ERRORG(I,2), ERROR(I,3) = 0;

```
INSERT PARTITION NUMBERS OF PRTNS1 INTO PARTNP;

INITIALIZE 'NO FREE BUFFERS' INTO STO;

INITIALIZE 'NO OCCUPIED BUFFERS IN COMMON' BFPOOL = 0

INITIALIZE TRM AND REC CONTROL VARIABLES PACTR = 0   PACRC = 0;

INITIALIZE REMAINING PARTITION BUFFERS PRTNS1 = 0   PRTNS2 = 0;

INITIALIZE SWITCH COMMANDS PT1NCP = 0   PT2NCP = 0;

INITIALIZE ALL FIELD CONTROL PARAMETERS
         FIELD1 = 0          FIELD2 = 0          FIELD3 = 0

220   CONTINUE;
      END;

FOR EACH ICP DO (275):

INITIALIZE PGM CONTROL VARIABLE DRSICP = 0;

INITIALIZE SWITCH COMMANDS PT1ICP = 0     PT2ICP = 0;

275   CONTINUE;
      END;

FOR EACH NODE DO (280):

INITIALIZE FOLLOWING PGM CONTROL VARIABLES:
SWCONT = 1      SEMPH = 3      BUSSEM = 3
DATABS = 3      SEMPRC = (-200);
```

```
INITIALIZE REQUEST BUFFER:
NODEST = 0;

INITIALIZE SWITCH WORKSPACE
QLIST = 0      Q = 0

280    CONTINUE
       END;

SET ALL PACKET ARRAY LOCATIONS
PACKET = 0
DUPLCT = 0

GENERATE MESSAGES WITH MEAN LENGTH OF IOP (I,8) AND MEAN INTER
ARRIVAL TIME OF IOP (I,9):

FOR EACH LOGICAL LINK DO (298):

NRDN = NUMBER OF MESSAGES GENERATED;

DO FOR EACH MESSAGE (294):
CALCULATE A MESSAGE LENGTH;
294    CONTINUE
       END

DO FOR EACH MESSAGE (296):
CALCULATE MESSAGE INTERARRIVAL TIME;
296    CONTINUE;
       END

298    CONTINUE
       END

MESSAGE LENGTHS AND INTERARRIVAL TIMES ARE OF POISSON DISTRIBUTION
```

```
      DSEED=675465.000
      DO 200 I=1,N
      STATEG(I)=2
      VIRTM(I)=1
      MESSAG(I)=1
      SRCIO(I)=1
      CNTR(I)=1
      DEST(I)=1
      RDNCTR(I)=1
      REASSM(I,1)=0
      DO 200 J=11,20
      IOP(I,J)=0
  200 CONTINUE
      DO 205 I=NTTLI,NTTL
      DO 205 J=11,20
      IOP(I,J)=0
      VIRTM(I)=1
  205 CONTINUE
      RST=4
      ST=5
      CNZ=6
      CALL ACCESS(RST,NRB,LINKNR,ICPNR,TNR,ACC,SDM,NCR,NSOU,NSING,NMUL)
      CALL RESERV(ST,ICP,LNK,NO,CL,RES,TPE)
      CALL PROGTM(TIMERZ,CLOCK,CNZ,ID)
      DO 220 I=1,NONCPS
      IFBUFR(I)=0
      LINE(I)=0
      ERRORG(I,1)=100000
      ERRORG(I,2)=0
      ERRORG(I,3)=0
      DO 225 J=1,8
      PARTNP(I,J,1)=PRTNS1(I,J,1)
      PARTNP(I,J,2)=0
      IF (J.GE.3) GO TO 225
      STO(I,J)=MNOBFS
  225 CONTINUE
      DO 230 J=1,MNOBFS
      DO 230 K=1,2
      BFPOOL(I,J,K)=0
      BFPOLR(I,J,K)=0
  230 CONTINUE
      DO 235 J=1,4
      DO 235 K=1,8
      PACTR(I,J,K)=0
      PACRC(I,J,K)=0
      PRTNS1(I,K,2)=0
      PRTNS1(I,K,3)=0
      PRTNS2(I,K,2)=0
      PRTNS2(I,K,3)=0
      PRTNS2(I,K,5)=0
      PRTNS2(I,K,6)=0
  235 CONTINUE
      DO 240 J=1,CMAX
      DO 240 K=1,8
      PT1NCP(I,J,K)=0
      PT2NCP(I,J,K)=0
  240 CONTINUE
```