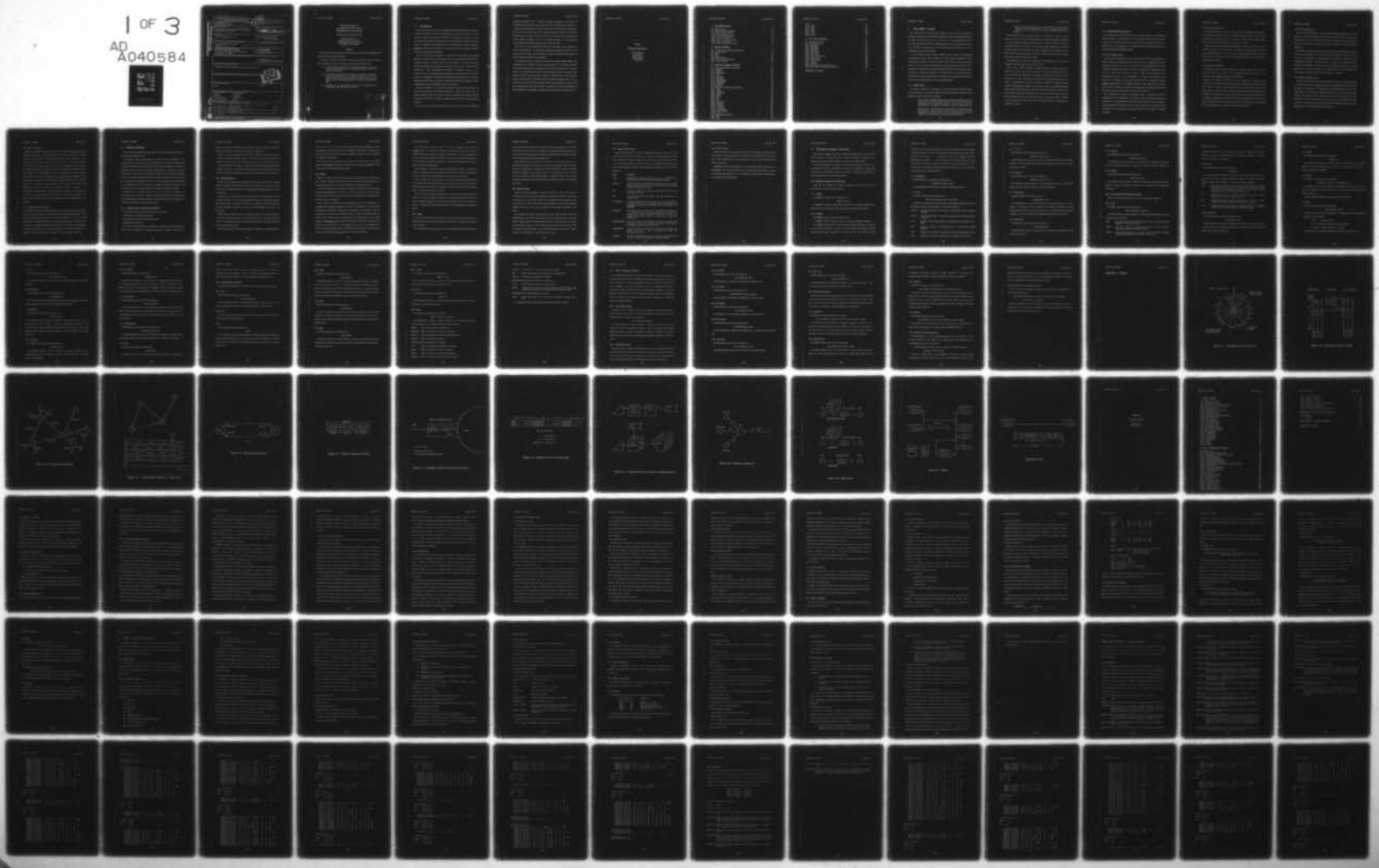
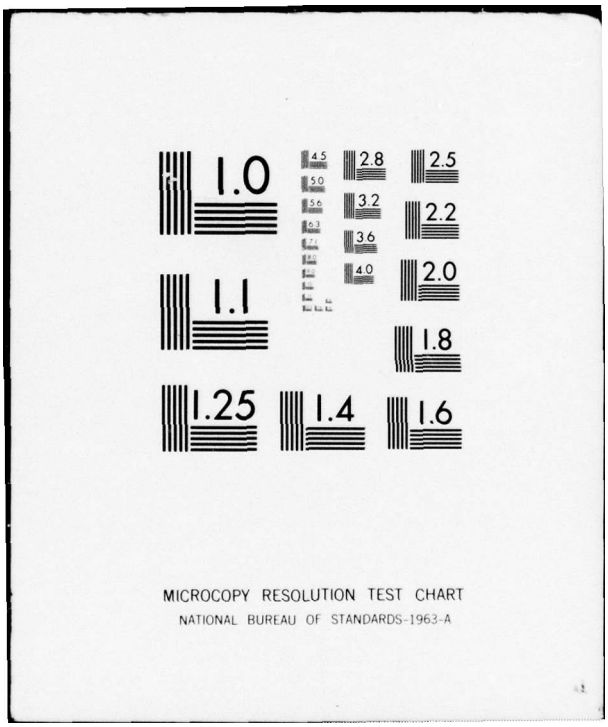
AD-A040 584 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 17/2
THE SIMULATION OF AN INTEGRATED VOICE/DATA COMMUNICATIONS NETWO--ETC(U)
MAY 77 M R BARBACCI DCA100-76-C-0058

UNCLASSIFIED DCA-100-76-C-0058 NL

1 OF 3
AD A040584




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: DCA100-76-C-0058	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): THE SIMULATION OF AN INTEGRATED VOICE/DATA COMMUNICATIONS NETWORK.	5. TYPE OF REPORT & PERIOD COVERED: Final Report, Phase I, Aug 1976 — May 1977 (on Phase I)
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Mario R. Barbacci, et. al.	8. CONTRACT OR GRANT NUMBER(s): DCA100-76-C-0058
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Schenly Park, Pittsburgh, Pennsylvania (252 P.)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element - 33126K
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Agency, 8th Street and South Court House Road, Arlington, VA 22204	12. REPORT DATE: 29 May 1977
	13. NUMBER OF PAGES: 171
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Packet Switch	Minicomputers	Voice
Coding	Networks	
Switching	Simulation	
Circuit Switch	Communications	
Multiprocessors	Data	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The final report covers three basic technical areas as follows:

a. Design and simulation of appropriate protocols for the circuit switched portion of an integrated network.
b. Design and simulation of appropriate protocols for the packet switched portion of an integrated network.
c. Simulation of different voice/data integration schemes such as SENET, PVC, etc..

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 403 081

SECURITY CLASSIFICATION OF THIS PAGE

DCA100-76-C-0058 May 29, 1977

The Simulation of an Integrated Voice/Data Communications Network

DCA Contract DCA100-76-C-0058

Mario R. Barbacci (Ed.)
Research Computer Scientist
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa

Abstract

This report covers the activities related to DCA Contract DCA100-76-C-0058 during the period August, 1976 to May, 1977.

The report is organized according to the three areas of the problem in which we have concentrated our efforts. The areas and the individuals involved are:

1) The design and implementation of appropriate protocols to handle the circuit switched portion of the Integrated Network traffic (M. Barbacci, K. Sakallah).

2) The design and implementation of appropriate protocols to handle the packet switched portion of the Integrated Network traffic. They include the mechanisms used to handle transient and permanent failures of lines and nodes (M. Barbacci, D. Levner, D. Siewiorek, W. Wu).

3) Variations in the basic SENET scheme used to implement the Integrated Network (M. Barbacci, W. Paulsen).

1. Introduction

During the period covered by this report (8/76-5/77) CMU continued the study of a network of integrated switches. An integrated switch is a message processor that can handle a range of message types. Principally, the message types are: Ordinary voice, encrypted voice, digital (100 bps to 50 kbps), and some video and other real time traffic. Integrated switches interconnected in a communications network constitute a backbone system. Local access lines tie in to the processors in the backbone network to provide user access to the network.

Prior to this effort, CMU emulated a single node of the network using C.mmp [Wul72], a multiprocessor system consisting of up to 16 miniprocessors (DEC PDP-11) connected through a crosspoint switch to a central memory. This work had several results: We identified the functional characteristics of the system, we developed and implemented a task decomposition that performs the different functions of an integrated switch, we developed theoretical models of performance for the switch, and finally, we developed a reliability study of the hardware components of the system. Statistics gathered during the experiments [Bar76b] indicated the feasibility of using multiprocessor nodes using relatively slow mini-processors.

During the emulation of the single node we ignored several critical issues since we were concerned mainly with measuring throughput. Among these issues were: 1) Routing Protocols, 2) Real Time traffic, and 3) Error Detection and Correction. During the period covered by this report we addressed these aspects of the integrated network.

The first part of this report describes the Network Simulator and the Command

Language Interpreter used to initialize and perform experiments. This part also describes the format of the trace files used to collect raw data from the experiments.

Some projections indicate that Class I traffic will constitute the major part (approx. 87%) of the total traffic carried by an integrated network in 1985. It is, therefore, essential to pay particular attention to the design of an effective Class I protocol that is both simple and flexible. Part II of the report describes a set of experiments conducted to study protocols for efficient Class I communications.

Part III of the report describes the experiments conducted to study protocols for reliable communications. The experiments studied three problems that arise in real communication networks: 1) node congestion, 2) line faults (transient and permanent), and 3) node faults (transient and permanent).

The network simulator is capable to simulate other schemes besides SENET. Some experiments were conducted to measure the performance of a "frameless" network transmitting Packetized Voice and Data (or PVD, for short). A PVD network is somewhat similar to the Packetized Voice Circuit (PVC) developed at Lincoln Laboratories, MIT [For76]. The PVC scheme treats Class I traffic in a similar manner to Class II and III traffic. Real time traffic is transmitted in packets through a preassigned path. There is no buffer reservation mechanism and Class I packets must compete for resources with other packets, albeit with a higher priority. The experiments with the PVD scheme and some comments on its similarities and differences with PVC are the subject of Part IV.

Part I

Network Simulator

Mario R. Barbacci
Navindra Jain
William Paulsen
Karem Sakallah
William Wu

1. The SENET Scheme

The integration of circuit and packet switching is based on the transmission of information using a Time Division Multiplex (TDM) scheme first proposed in [Cov75]. In this scheme, time slices of fixed size - a frame period - are transmitted synchronously over high speed lines between two adjacent switches in the network. The frame acts as an envelope for smaller time slices of variable size, each acting as a slot for transmitting a "packet" of information.

In this Slotted Envelope NETwork (SENET) scheme, frames are transmitted synchronously between adjacent nodes and a circuit switching subnetwork can be defined by the reservation of fixed slots inside a frame. Since the slots carry the same amount of bits every frame period, beginning at exactly the same point in time, this is equivalent to the reservation of dedicated channels between the two adjacent nodes. The rest of a frame can be allocated on a demand basis. If a suitable protocol is defined for the proper interpretation of the information transmitted on this part of a frame, a packet switching network can be easily implemented.

1.1. Traffic Types

The primary driving force in the design of a communications system is the character of the information it must handle. Based on existing communication networks, three different classes of traffic can be identified:

> Class I Traffic.- Characterized by long transactions requiring continuous real time response (voice, video, facsimile). This type of traffic can be transmitted in the synchronous portion of the frame. They are representative of transactions transmitted in a circuit switching "network".
>
> Class II Traffic.- Characterized by short discrete transactions requiring near real time response (interactive data). This type of traffic can be transmitted by dynamic allocation of slots in the asynchronous portion of a frame. They are transmitted in a packet switching "network".

> Class III Traffic.- Characterized by long transactions requiring neither continuous nor immediate response (bulk data). This type of traffic can be transmitted, as the previous class, on the asynchronous portion of a frame.

The details of a frame are shown in Figure 1.1. Starting at 12 o'clock a certain number of bits are reserved for CCIS (Common Channel Interswitch Signaling), followed by a Class I region containing the real time traffic. The end of the Class I region is indicated in Figure 1.1 at 5 o'clock. Class II and III regions containing the interactive and bulk traffic occupy the rest of the frame. (Unless we make it explicit, we shall make no distinctions between Class II and Class III traffic. We will use the term Class II to indicate both interactive and bulk data).

The differences between the requirements and characteristics of the above classification of traffic imply different types of service:

Class I traffic is either accepted or rejected, with short connection delays and without error control. Class I traffic requires low delay and a constant throughput in order to maintain the intelligibility of the message. Class II traffic is always accepted but may incur a system delay, with short connection and cross-network delays. The traffic is characterized by bursts of information followed by "waiting" periods, requiring a high degree of reliability. Class III traffic is always accepted (although the network might temporarily suspend this type of service), with longer connection and cross-network delays than the previous class. In Class III traffic, variations in the delay of individual packets is not important and this can be used to reduce the impact of the long periods of high traffic volume. As in Class II traffic, a high degree of reliability is required.

1.2. Implementation Assumptions

The above discussion summarises the top level characteristics of an Integrated Switching Network. For the purposes of the project, however, we had to take a closer look at lower level considerations that will play a role in the actual implementation of a SENET system. The remainder of this section outlines the major characteristics of a possible implementation [Bar76a, Bar76b].

1.2.1 Class I Mapping Tables

The real time requirements of Class I traffic dictates that it be processed in a special fashion by the integrated switch. Since voice carries a significant amount of redundancy, a larger error rate can be tolerated in most applications (secure voice transmission presents lower tolerance to errors). By eliminating or reducing the need for error control, the transmission of Class I traffic can be performed in a straightforward manner. The establishment of a logical circuit between two subscribers is reflected, on each switch along the path, in the updating of Class I Mapping Tables internal to the integrated switches. These tables indicate, for each slot in the Class I region of an input frame, both the output frame (output channel) and slot position reserved for the logical circuit, as shown in Figure 1.2.

The reservation of entries in the mapping tables is performed during the establishment of the communication and remains in effect until the communication is terminated. The reservation of the slots along the communication path guarantees a fixed bandwidth for this type of traffic. As traffic requirements vary during the day the portion of a frame reserved for Class I traffic can be reduced or expanded accordingly.

1.2.2 Non-Homogeneous Links

The input and output links in the network need not necessarily be homogeneous. For a given time window length, say 10 milliseconds, the length of a frame varies according to the capacity of the link. Thus, for a T1 carrier, a frame contains 15,440 bits. For slower carriers, the frame will be accordingly smaller.

Allowing the use of lines with different capacities permits the use of similar nodes in different capacities in the network. They can appear as switching nodes, connecting high speed trunk lines, as part of the access system connecting and concentrating large numbers of low volume users, connecting HOST computers into a communications network, etc (Figure 1.3).

1.2.3 Asynchronous Behavior

The scheme does not assume the existence of a master network clock. Each link transfers frames at a constant rate (1 frame/time slice) but the links are not necessarily synchronized among themselves (Figure 1.4).

Requiring a centralized network clock has a serious impact on the reliability of the network. In an asynchronous network each link works independently from all other links. There is no central critical component (a clock) whose failure will bring the network down. Each link between two nodes constitutes an isolated entity whose rate of failure has no impact (other than perhaps on the volume of traffic permissible) on the network. This approach also allows the presence of transient errors in a link which might possibly throw it out of step. Link protocols and error detection and reporting are the responsibility of the two nodes interconnected by the line.

1.2.4 Line Interface Devices

The line functions (detection of frame and packet headers, detection of special bit patterns or flags, Cyclic Redundancy Checks, etc) are performed by special purpose hardware devices. The actual (physical) input and output operations are performed by these Line Interface Devices (LID) using their Direct Memory Access (DMA) capabilities (Figure 1.5).

The presence of hardware devices to handle line error detection frees part of the node processing capability. From the LID, the frame is loaded directly into memory and the event is indicated to the integrated switch processor(s). Having special line handling devices allows the detection of errors on a packet per packet basis. If error detection were performed over the entire frame, a single error anywhere in the frame might make it invalid and therefore the frame might have to be retransmitted, an unacceptable situation from the point of view of Class I transmission.

1.2.5 Packet Transmission Format

The use of the Advanced Data Communication Control Procedures (ADCCP) [ADC75], or variant thereof, seems reasonably well-suited to this application. Each packet to be transmitted is augmented, as in ADCCP, by hardware-generated header and trailer fields, Figure 1.6. The header field identifies the start of a packet (by a unique flag-character) and passes control information to the Line Interface Device (LID) at the destination. The trailer field contains the cyclic redundancy field followed by the flag-character to indicate the end of the packet. By so delimiting the packet with unique flag-characters, error detection becomes straightforward.

1.2.6 Class I Transmission

Since Class I slots are small (say, 80 bits for an 8 Kbps vocoder channel [see For75]), sending each as a separate packet would entail a great deal of overhead. Indeed, the main reason for using separate packets for Class II traffic is error detection, which is not a concern with Class I transmissions. The reasonable conclusion is to send the entire Class I portion of each frame as a single packet. This has the advantage of minimizing the number of overhead bits required (a single header/trailer pair). However, it is important that this Class I packet should always be accepted by the destination LID, even if a transmission error is detected. What is needed is a type field in each packet header which identifies the packet as either Class I or Class II (Figure 1.7). The solution to the problem is then to have the destination LID intentionally ignore any possible error indication for Class I packets. As a consequence, control information used to establish or break Class I communications must be send as Class II packets. Thus, a frame can not be dedicated exclusively to Class I traffic, some small portion must be reserved for Class II control packets for this purpose.

1.2.7 Frame Generation and Timing

According to the original Coviello and Vena concept, new real-time data appears say, every 10 milliseconds (in a T1 carrier this corresponds to 15,440 bits/frame). Hence, every 10 milliseconds a Class I packet must be transmitted. The start of a frame would simply be indicated by receipt of each Class I packet header. No explicit start-of-frame marker would be required. In other words, a frame consists of a single Class I packet followed by zero or more Class II packets. The intervals between Class I packets are filled, either partially or completely, by Class II packets (Figure 1.8).

2. Network Simulator

In this chapter we describe the organization of the Network Simulation system used to model an integrated network.

The simulator allows experimentation with arbitrary network topologies, traffic patterns, node processing power, line characteristics, etc. The system is implemented in SIMULA-67 and runs on the PDP-10 processors at CMU. Modelling the network in a simulation language instead of emulating it using C.mmp was dictated by practical considerations. C.mmp has a fixed upper bound in the number of processors available. If we try to simulate multiprocessor nodes, the number of nodes that can be simulated is of necessity rather small (in the order of 4 or 5). Moreover, some processors have to be dedicated during the experiments to interface with the user, perform I/O operations, etc., thus reducing even further the number of processors available.

Simulation was also better suited for some of the planned experiments. In particular, we were interested in studying the behavior of the network under transient and permanent faults in the lines and nodes. Injecting faults in the real hardware was clearly more difficult than in the simulated network.

2.1. A Global View of the Network Simulator

The network simulation experiments take place in three steps:

1) Definition of the network characteristics

2) Running the simulator and collecting trace data

3) Analysing the trace data.

The first two steps are performed interactively. The latter is performed off-line, at a later time. This partition of the experiment allows us to perform any number of

analysis procedures independently of the simulation run, i.e. we have the freedom to study the data at leisure without having to repeat the experiment.

Figure 2.1 shows the organization of the system. The rest of this chapter describe in detail the Network Simulator proper. The Command Language Interpreter, the Analysis programs, and the results of the experiments are described in later chapters.

A network (Figure 2.2) consists of a set of links (transmission lines), switching nodes, and host processors. Hosts generate traffic (packets); nodes process the traffic and route the frames and packets; lines carry the frames and packets from node to node.

2.2. Host Processors

Several types of hosts can be attached to a node. Each type of host is geared towards handling a different type of traffic. For the purpose of this section we are interested in the following types of host: Input, Output, and Voice-Generator, as shown in Figure 2.3.

Input hosts are asynchronous processes responsible for the generation of Class II and Class III packets. They are driven by tables describing the rates of generation for the different types of packets. The tables are initialized by the user through the Command Language Interpreter and remain in effect throughout the length of an experiment.

Output hosts are processes which are normally dormant. They are activated by a node when a packet destined for the host arrives to the node. The output host then removes the packet from the node and does the proper clean-up operations associated with the end of the packet life.

Voice Generators are asynchronous processes responsible for the generation of

Class I traffic. They are driven by tables describing the rates (calls/minute) and the characteristics of the calls (traffic rate and duration). These tables are initialized by the user through the Command Language Interpreter and remain in effect throughout the length of the experiment.

Hosts send and receive packets via two queues in the node. These are the Host-Input and Host-Output queues. Each node has exactly one of each, regardless of the number and type of hosts attached to the node.

2.3. Nodes

Nodes are made of asynchronous processors of, possibly, different characteristics, and a number of buffer queues used to store the packets. The characteristics of a node are specified by the user through the Command Language Interpreter.

There are two input queues from which a node gets its packets. They are the Line-Input and Host-Input queues. The former is used by the input lines to store newly arrived packets. The latter is used by the hosts of the node to store newly generated packets, as shown in Figure 2.4.

After a packet is removed from an input queue by one of the processors, it is examined and processed according to its type, source, and destination. The operations executed for the different packet types are described in later chapters. For our purposes let us say that most packets that arrive to a node are routed to either an output host or another node in the network. This is done through a set of output queues. A Host-Output queue is used to store packets destined for the output host associated with the node. A set of Line-Output queues, one for each output line, is used to store packets destined for an adjacent node. These packets are removed by the lines and placed in the Line-Input queues of the destination nodes.

Besides the input and output queues, nodes have one additional queue called the Holding queue. It is used to store copies of the packets that have been sent to an adjacent node, while waiting for a positive acknowledgement. If an acknowledgement fails to arrive within a prespecified "time-out" window, the packet in the holding queue is retransmitted. The algorithms and techniques used to handle transmission errors are described in Part 3 of this report.

Routing tables are used to drive the routing algorithms. Briefly, a routing table contains the identity of the best adjacent node which can be used as an intermediary in order to reach a given final destination. More details are given in Part 3.

Class I mapping tables are used to drive the Class I processing algorithms. Briefly, these tables contain the characteristics of each channel in use as well as the "route" that the channel must follow (i.e. the output line and the position of the channel within the frame). More details are given in Part 2.

Each node has some fixed storage capacity for packet buffers. When the storage capacity of a node is exceeded, the node is said to be congested and new traffic can not be accepted. Congestion can arise due to increases in traffic or due to component failures. The techniques used to detect, avoid, or correct congestion are described in Part 3.

2.4. Lines

Lines are synchronous processes which are responsible for taking packets out of a line output queue (in the sending node) and placing them into a line input queue (in the receiving node).

At the beginning of each frame period a line transmits the Class I packet (the Class I

portion of the frame), followed by as many Class II and III packets as can fit in the rest of the frame. If no Class II or III packet can be completely transmitted before the start of the next frame period, these packets are delayed until after the Class I packet of the next frame has been transmitted (Figure 2.5).

Lines provide the basic timing of the network i.e. while nodes and hosts are asynchronous processes, lines are driven by the frame frequency. Each line of the network has characteristics determined by the user through the Command Language Interpreter. Although the basic frame period is a network parameter (i.e. it is the same for all lines), lines can have different speeds or transmission rates. Lines are not necessarily synchronized with each other and a constant "line skew" can be specified by the user.

2.5. Packet Format

Under the working assumptions mentioned in Chapter 1, we are not particularly concerned about the specific data transmission protocol used by the lines. We assume that appropriate error detection mechanisms are available to prevent garbled packets from arriving to a node. In this section we will explain the format of the simulated packets as seen by a node, and the meaning of the information transmitted in the header.

For simplicity, the network simulator only uses four types of packet. These are labelled types 1, 2, 3, and 4. Type 2 packets are Class II with high priority. These are used to transmit control information between the nodes. Type 3 packets are the host generated Class II packets. Type 4 packets play the role of Class III packets. They are handled in the same fashion as type 3 packets but with a lower priority. Type 1 packets correspond to Class I packets in PVD experiments.

2.5.1 User Information Packets

These are the packets generated by the different hosts of the network. Thus, this group includes all packets of type 1, 3, and 4. We will now explain the different fields that appear in these packets besides the user information or data. Bear in mind that we are describing simulation packets and that in a real implementation, packets will have a more condensed header. We included extra information mainly for tracing purposes.

Field	Comments
TYPE	This field defines the packet type (i.e. 1, 3, or 4). The TYPE field is used by the nodes to process the packets in priority order.
LENGTH	This field contains the simulated packet length, in bits. The LENGTH field is used to compute the simulated transmission delays, copying delays, and storage requirements.
SN	The Source Node field identifies the node attached to the host generating this packet.
DN	The Destination Node field identifies the node attached to the destination host.
LASTIMEOUT	This field contains the time that the packet was last transmitted. It is used to detect errors by comparing the difference between LASTIMEOUT and the current time with a time out constant. See NTIMEOUT.
NTIMEOUT	This field counts the number of times this packet has "timed-out". It is used by the error detection routines. If the number of retransmissions exceeds some limit, the line and or adjacent node become suspect. This counter is reset after a successful transmission.
PACKETNUMBER	This field contains a unique identifier (an integer) associated with each packet that is ever created in the experiment. It is used to acknowledge packets and in the trace files.
SENDINGNODE	This field contains the identity of the node that last transmitted the packet. This field is used to generate and route the acknowledgement packets.
TEMPDN	This field contains the identity of the node to which the packet must be sent. This field is computed by the routing procedures.

2.5.2 Control Packets

Control packets are always packets of type 2 and have higher precedence that all other packets. Control packets carry control commands to be obeyed by the receiving node. Some control packets also carry data to be used by the receiving node (e.g. routing information).

In addition to the fields described for the user packets, control packets carry a CTLMESSAGE field. This field is an array of six (6) integer that identify a control command and its parameters, if any. The meaning of these commands and parameters will be explained in later chapters.

3. Command Language Interpreter

The Command Language Interpreter provides the facilities for the user to interact with the system. It allows the user to define the network characteristics, the experiment parameters, and the collection of trace data for further analysis.

The Command Language Interpreter is invoked automatically when the system is run. It prompts the user for commands and performs then immediately. The available commands can be classified in three major groups: 1) Network Characteristics, 2) Experiment Characteristics, and 3) Miscellaneous.

3.1. Network Characteristics Commands

These are the commands that define the topology of the network and the characteristics of the Hosts, Nodes, and Lines.

3.1.1 NEWNET

The NEWNET command takes the following form:

NEWNET <integer>

The NEWNET command is used to define the size of the network by specifying the number of nodes as an integer. This is usually the first command to be issued.

3.1.2 CONNECT

The CONNECT command takes the following form:

CONNECT <node1>-<node2> [/ <speed> / <skew12> / <skew21> / <delay>]

The CONNECT command defines a line between two nodes. The nodes are specified by two integers between 1 and the size of the network (see NEWNET). Besides specifying the nodes connected by the line, the user can specify the line

characteristics. The first optional parameter is the speed of the line in Kilobits/second. The default speed is 1544 Kbps (T1). The second and third optional parameters specify the skew of the lines. These parameters are given by an integer number of milliseconds and is used as an initial line start-up delay. The default skew is 0 ms. The last optional parameter is an integer which defines the "cost" of using the line. It is a relative delay time needed in travelling from one node to another. It is used by the routing algorithms. The default delay is 4.

3.1.3 DISCONNECT

The DISCONNECT command takes the following form:

DISCONNECT <node1>-<node2>

The DISCONNECT command is used to eliminate a line from the network.

3.1.4 HOST

The HOST command takes the following form:

HOST <node> <switch>:<arg> <switch>:<arg> .....

The HOST command defines the characteristics of the traffic generated by the hosts attached to a node (the first parameter). The allowed switches are:

H<n>:<r>	Defines which fraction, r, of the data traffic generated at this node goes to node n.
V<n>:<r>	Defines which fraction, r, of the voice traffic generated at this node goes to node n.
A:<n>	Defines n as the average number of packets/millisecond generated at this node.
D:<r>	Defines the fraction, r, of DATA packets (1-r is the fraction of BULK packets).
L:<e>	Defines e as the number of Erlangs of voice traffic generated at this node.
T:<h>	Defines h as the average holding time of calls generated at this node.

3.1.5 KILLHOST

The KILLHOST command takes the following form:

KILLHOST <nodei> <nodej> .....

The KILLHOST command removes the host associated with the nodes specified in the command. The traffic patterns generated by the remaining hosts in the network might have to be respecified.

3.1.6 PROCNUM

The PROCNUM command takes the following form:

PROCNUM <n>-<p> <n>-<p> .....

The PROCNUM command specifies a list of node/processor pairs. It defines the number of processors/node. The default value is 4 processors.

3.1.7 WMOVEDELAY

The WMOVEDELAY takes the following form:

WMOVEDELAY <delay>

The WMOVEDELAY is used to specify the speed of the processors used in the network. The parameter to this command is a real number specifying the time (in milliseconds) that takes to move a 16 bit word to a memory location. The default value is 0.005 ms (i.e. 5 microseconds) and corresponds to the speed of a PDP-11/20.

3.1.8 FRAMETIME

The FRAMETIME command takes the following form:

FRAMETIME <period>

The FRAMETIME command is used to specify the frame period, in milliseconds. The default value is 10 ms.

3.1.9 MAXHOLD

The MAXHOLD command has the following form:

MAXHOLD <node> <size>

The MAXHOLD command is used to specify the buffer space allocated to the holding queue of a node. It is specified as a number of bits. The default value is 128,000 bits.

3.1.10 MAXIN

The MAXIN command takes the following form:

MAXIN <node> <size>

The MAXIN command is used to specify the buffer space allocated to the input queues of a node. It is specified as a number of bits. The default value is 128,000 bits (16 Kbytes).

3.2. Experiment Characteristics Commands

These are the commands that define the characteristics of the experiment.

3.2.1 MODE

The MODE command takes the following form:

MODE <modetype> <modetype> .....

The MODE command is used to specify the type of experiment to be performed. The following types are defined:

DATA	Only Class II and III traffic to be generated. No Class I traffic is generated (regardless of the HOST command).
VOICE	Only Class I traffic is to be generated. No Class II or III traffic is generated (regardless of the HOST command).
ERROR	Faults will be injected in the simulation. This mode enables the reliability and fault detection/correction procedures to be active.

Note that more than one of the types can be specified. The default mode is VOICE DATA (i.e. Classes I, II, and III traffic will be generated according to the HOST command. No faults will be simulated).

3.2.2 ALLOC

The ALLOC command takes the following form:

ALLOC <mode>

The ALLOC command specifies the strategy to be used in allocating the Class I portion of the frames. The mode is specified by a 3-letter keyword that describes the traffic rates, slot sizes, and region size. The following modes are defined:

FFF — Fixed rate (i.e. all channels will use the same traffic rate. See the VRATE command), Fixed slot length (i.e. the Class I region is allocated in fixed size units), Fixed boundary (i.e. the Class I region occupies a fixed portion of the frame. See the VFRACTION command).

VFF — Variable rate (i.e. there are different channel rates. See the VDIST command), Fixed slot length, Fixed boundary.

VFV — Variable rate, Fixed slot length, Variable boundary (i.e. the Class I region is allowed to shrink and expand. See the Vfraction command).

VVV — Variable rate, Variable slot length (i.e. different channels will be allocated different size slots in the Class I region), Variable boundary.

3.2.3 VFRACTION

The VFRACTION command takes the following format:

VFRACTION <fraction>

The VFRACTION command specifies an upper limit in the size of a Class I region. In allocation modes **F (see ALLOC command) this is also the lower limit. The default value is 0.87 (i.e. 87% of the frame).

3.2.4 VRATE

The VRATE command takes the following form:

VRATE <rate>

The VRATE command specifies the basic channel rate in bits/second. In allocation mode FFF (see ALLOC command) this is the rate for all channels. The default value is 16,000 bits/second and mode FFF.

3.2.5 VDIST

The VDIST command takes the following form:

VDIST <rate>/<fraction> <rate>/<fraction> .....

The VDIST command is used to specify the distribution of channel rates. For each rate (bits/second) the user must specify the fraction of channels with such rate. The default values are:

2400/0.1 4000/0.1 8000/0.15 16000/0.5 32000/0.1 50000/0.05

3.2.6 FAIL

The FAIL command takes the following form:

FAIL <component> <time>

The FAIL command makes a component fail at a prespecified time. the following components can be specified:

N <n> Failure of node n.

L <n1>-<n2> Failure of line betwen nodes n1 and n2.

P <n>-<p> Failure of processor p in node n.

The fault times are specified in milliseconds from the start of the experiment.

3.2.7 FIX

The FIX command takes the following form:

FIX <component> <time>

The FIX command is used to indicate the repair of a failed component. See the FAIL command.

3.2.8 PKTLENGTH

The PKTLENGTH command takes the following form:

PKTLENGTH <size>

The PKTLENGTH command is used to specify the Class II and III packet lengths. The size is specified in number of bits. The default value is 2000 bits.

3.2.9 NOPKTIN

The NOPKTIN command takes the following form:

NOPKTIN <interval>

The NOPKTIN is used to specify a time limit before the absence of traffic from an adjacent node becomes suspicious. The interval is specified in milliseconds. If the interval elapses without any traffic being received from a neighbor, a special high priority packet requesting immediate response is sent. The default value is 500 milliseconds.

3.2.10 TIMEOUT

The TIMEOUT command takes the following form:

TIMEOUT <interval>

The TIMEOUT command is used to specify the "time-out" window before an unacknowledged packet is considered lost and must be retransmitted. The default value is 125 milliseconds.

3.2.11 RTXLIMIT

The RTXLIMIT takes the following form:

RTXLIMIT <limit>

The RTXLIMIT command is used to specify the maximum number of times an unacknowledged packet can be retransmitted before a line or node becomes suspect. When this number of retransmissions is exceeded, a special high priority packet requesting immediate acknowledgement is sent. The default value is 3 retransmissions.

3.2.12 HELLOLIMIT

The HELLOLIMIT command takes the following form:

HELLOLIMIT <limit>

The HELLOLIMIT command is used to specify the maximum number of times a high priority "hello" packet can be retransmitted. If this limit is exceeded the line is considered faulty and the appropriate procedures are invoked. The default value is 2 retransmissions.

3.2.13 ROUTEUPDATE

The ROUTEUPDATE command takes the following form:

ROUTEUPDATE <interval>

The ROUTEUPDATE command is used to specify the frequency of updates of the routing tables. The interval is specified in milliseconds. The default value is 500 ms.

3.2.14 XEQ

The XEQ command takes the following form:

XEQ <interval>

The XEQ command is used to specify the length of the experiment. The interval is

specified as the number of milliseconds the simulated network must work. There is no default value for the parameter. It must be specified. When this command is issued, the network simulator starts running. This must therefore be the last command.

3.3. Miscellaneous Commands

The following commands provide facilities not covered in the previous commands.

3.3.1 HELP

The HELP command takes the following form:

HELP <command.name>

The HELP command can be used to obtain assistance from the system. If no command name is specified, the system will type the list of available commands.

3.3.2 QUIT

The QUIT command does not take any parameters. Its use terminates the session and control returns to the PDP-10 monitor.

3.3.3 !

The ! command takes the following form:

! <text>

The ! command is used to insert comments. The Command Language Interpreter will ignore whatever follows the ! until the end of the line. This command is useful to provide users of command files (see the READ command) with commentaries and other information.

3.3.4 READ

The READ command takes the following form:

READ <filename>

The READ command can be used to execute predefined sequences of commands placed in a text or command file. The READ command in fact substitutes the user terminal with the command file. When all the commands in the file have been executed the user terminal again becomes the command language interface. The READ command can be used inside a command file, allowing the nesting of command files.

3.3.5 SAVE

The SAVE command takes the following form:

SAVE <filename>

The SAVE command can be used to save the current setting of the network and simulation parameters in a command file. The command file can later be read to set the parameters to the values they had at the time the SAVE command was issued. See the READ command.

3.3.6 SEED

The SEED command takes the following form:

SEED <integer>

The SEED command can be used to specify an integer to be used as the random number generator seed by the network simulator and the SIMULA-67 run time system. The default value is 10.

3.3.7 TRACE

The TRACE command takes the following form:

TRACE <filename>

The TRACE command is used to specify the name of the file where the trace data is to be written. The default file name is DSK:TRACE.TRC.

3.3.8 TITLE

The TITLE command takes the following form:

TITLE <text>

The TITLE command can be used to specify the first line of the trace file. It is useful to identify the experiment, the user, etc.

3.3.9 SHOW

The SHOW command takes the following form:

SHOW <parameter> <parameter> .....

The SHOW command is used to display on the user's terminal the setting of various network parameters. The following parameters can be specified:

NODES	Shows number of nodes in the network.
PROCNUM	Shows the number of processors in each node.
HOST <n>	Shows specifications of the host(s) attached to node n.
CONNECT	Shows the network connections.
FRAMETIME	Shows frame period in milliseconds.
TITLE	Shows the heading to be printed in the trace file.
MODE	Shows the current mode (voice, data, error).
MAXIN	Shows the memory allocated for the input queues.
MAXHOLD	Shows the memory allocated for the holding queues.

TOTLINES Shows the number of lines connected to each node.

FAIL Shows the schedule of failure/repair events of components.

LIMITS Shows all the variables for error control.

WMOVEDELAY Shows time needed to move a 16-bit word.

ALLOC Shows the current voice-slot allocation scheme.

VRATE If ALLOC is FFF, shows the voice rate employed. If ALLOC is VFF or VFV, shows the basic voice rate upon which the basic slot length is based.

VFRACTION Shows the percentage of voice traffic.

VDIST Shows the fraction of each voice rate for all allocation schemes except FFF.

If no arguments are specified, the SHOW command will display everything.

4. The Tracing Facilities

As packets are generated, processed, and terminated, a trace of these activities is kept in a trace file. The trace file not only contains information about the events in he life of a packet but it also contains information about the state of the nodes and lines of the network. The trace file contains a record of all significant event during a simulation run and can be processed by user defined analysis programs. This chapter describes the format of the trace file and the meaning of some of the events so recorded. Other events, specific to particular modes of operation (i.e. experiment type) will be described later.

4.1. Trace Entry Format

Regardless of the particular event being recorded, all trace file entries share a common format. A trace entry is a text line which contains the event time, the event identification, and a vector of event parameters:

<time> <event> <parameters>

The time information is always recorded in milliseconds from the start of the experiment. The event identification consists of one or two keywords that uniquely identify the type of event. The parameters provide the information necessary to identify the event and the number and type of parameters depends on the type of event.

4.2. Initialization Trace

When the network is being initialized a group of trace entries are generated. These entries describe the initialization of the network components. Although these entries do not provide information with regard to steady state events, they are nevertheless useful to verify that no errors were commited at the start of the experiment.

4.2.1 NEW NODE

The NEW NODE entries have the following form:

<time> NEW NODE <node>

The NEW NODE entry describes the initialization of a network node.

4.2.2 NEW PROC

The NEW PROC entries have the following form:

<time> NEW PROC <node> <processor>

The NEW PROC entry describes the initialization of a node processor.

4.2.3 NEW IHOST

The NEW IHOST entries have the following form:

<time> NEW IHOST <node>

The NEW IHOST entry describes the initialization of a node input host.

4.2.4 NEW VIHOST

The NEW VIHOST entries have the following form:

<time> NEW VIHOST <node>

The NEW VIHOST entry describes the initialization of a node PVD voice generator host.

4.2.5 NEW OHOST

The NEW OHOST entries have the following form:

<time> NEW OHOST <node>

The NEW OHOST entry describes the initialization of a node output host.

4.2.6 NEW VGEN

The NEW VGEN entries have the following form:

<time> NEW VGEN <node>

The NEW VGEN entry describes the initialization of a node call generator host. This entry only appears in SENET simulations. See NEW VIHOST.

4.3. Packet Trace Entries

In this section we describe the general format of the trace event entries associated with the normal (i.e. error free) transmission of data (type 3 and 4) packets from source host to destination host. We also include the acknowledgement (type 2) packet trace entries. Later chapters contain additional cases of trace entries.

4.3.1 CREATE PKT

The CREATE PKT entries have the following form:

<time> CREATE PKT <packet> <source> <destination> <type> <length>

The CREATE PKT entry describes the creation of a packet by a SENET host. The parameters identify the packet (a unique integer), the source node, the destination node, the type (3 or 4), and the length of the packet (in bits). When a packet is created, it is placed in the host-input queue of the source node for processing.

4.3.2 FRAME TRANS

The FRAME TRANS entries have the following form:

<time> FRAME TRANS <node1> <node2>

The FRAME TRANS describes the transmission of a frame between two adjacent nodes. This entry only describes the arrival of the frame transfer period and the

transmission of the Class I packet in a SENET experiment. The transmission of individual packets (PVD or otherwise) is indicated by entries of type SND PKT.

4.3.3 SND PKT

The SND PKT entries have the following form:

<time> SND PKT <packet> <source> <destination> <from> <to>

The SND PKT describes the transmission of a packet on a line (inside a frame in SENET, alone in PVD). The parameters describe the packet number (attached to the packet at creation time, see CREATE PKT), the packet initial source node, final destination node, sender node (the node sending it right now), and receiver (the node at the other end of the line).

4.3.4 END PKT

The END PKT entries have the following form:

<time> END PKT <packet> <source> <destination> <type> <length>

The END PKT entry describes the arrival of a packet to its final destination (an output host). The parameters are the same used in the CREATE PKT entry.

4.3.5 GEN CTLPKT (Acknowledgements)

Acknowledgements are control packets generated by a node when a newly processed packet has been placed in the holding queue. The generation of an acknowledgement has the following form:

<time> GEN CTLPKT <packet> <source> <destination> <sender> <receiver>

1 <pkt.ackd> 0 0 0 0 <length>

This entry identifies both the acknowledgement packet and the packet being acknowledged. The entry describes the control packet identification, the source and

destination nodes, the sender and receiver nodes (identical to the previous two for this type of entry), the control command (1 = acknowledgement), the identity of the packet being acknowledged, and the length of the control packet.

4.3.6 REC CTLPKT (Acknowledgement arrival)

The arrival of an acknowledgement control packet to a node is described by an entry of the following form:

<time> REC CTLPKT <packet> <source> <destination> <sender> <receiver>

1 <pkt.ackd> 0 0 0 0 <length>

The entry describes the packet number, the source and destination nodes, the immediate sender and receiver nodes (equal to the previous two), the control packet command (the entire vector), and the length.

Appendix A: Figures

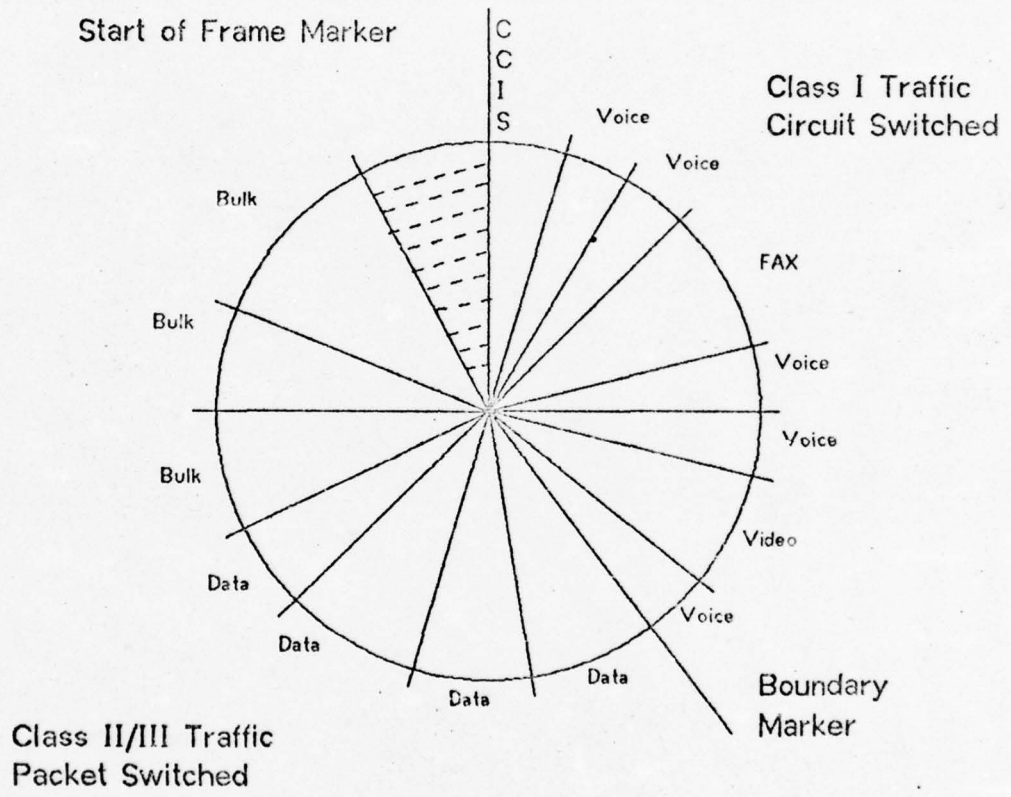


Figure 1.1 - An Integrated Network Frame

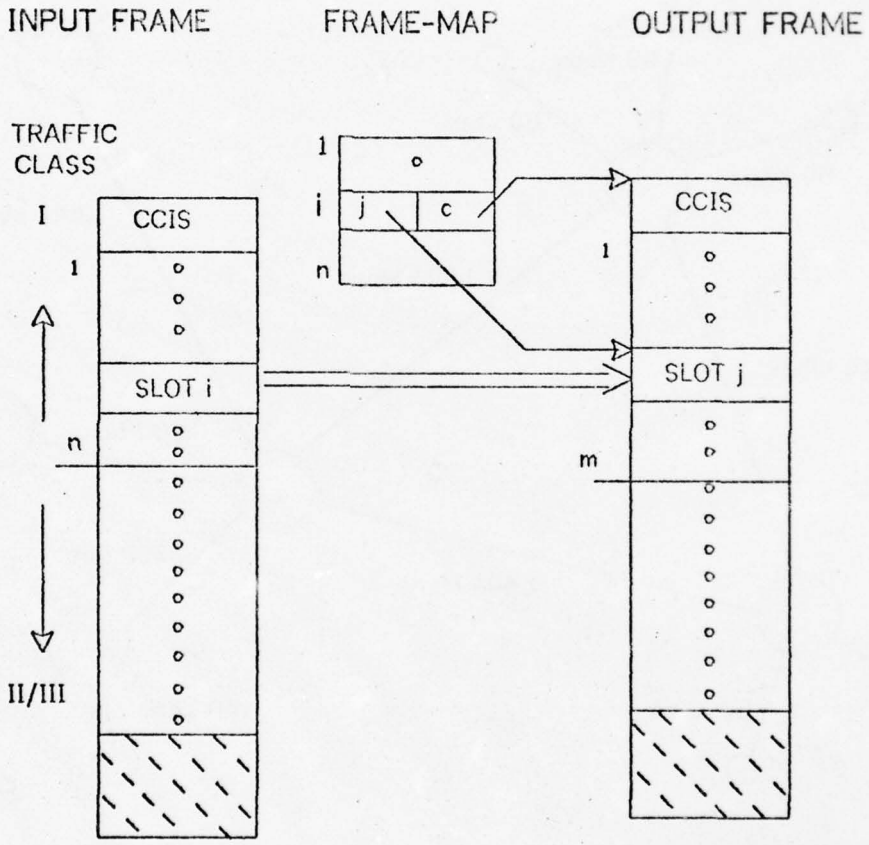


Figure 1.2 - Processing of Class I Traffic

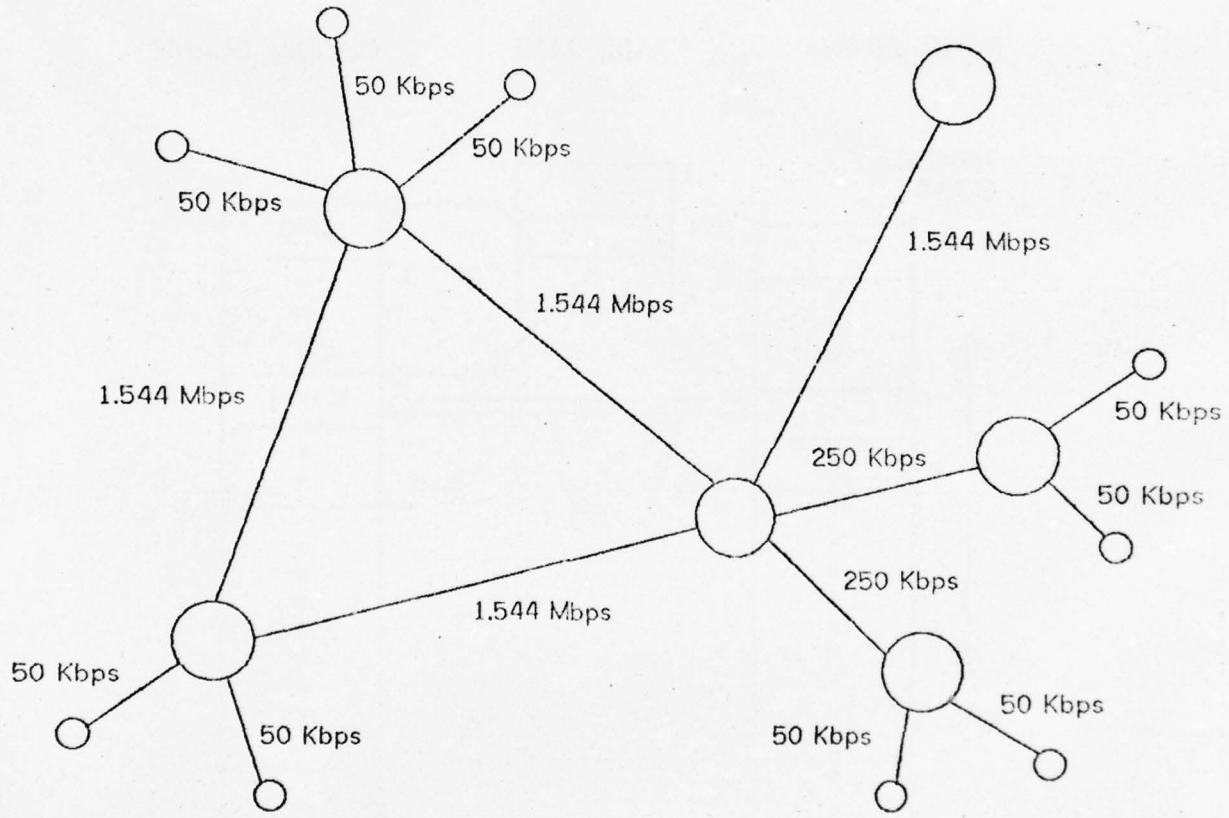


Figure 1.3 - Non-Homogeneous Links

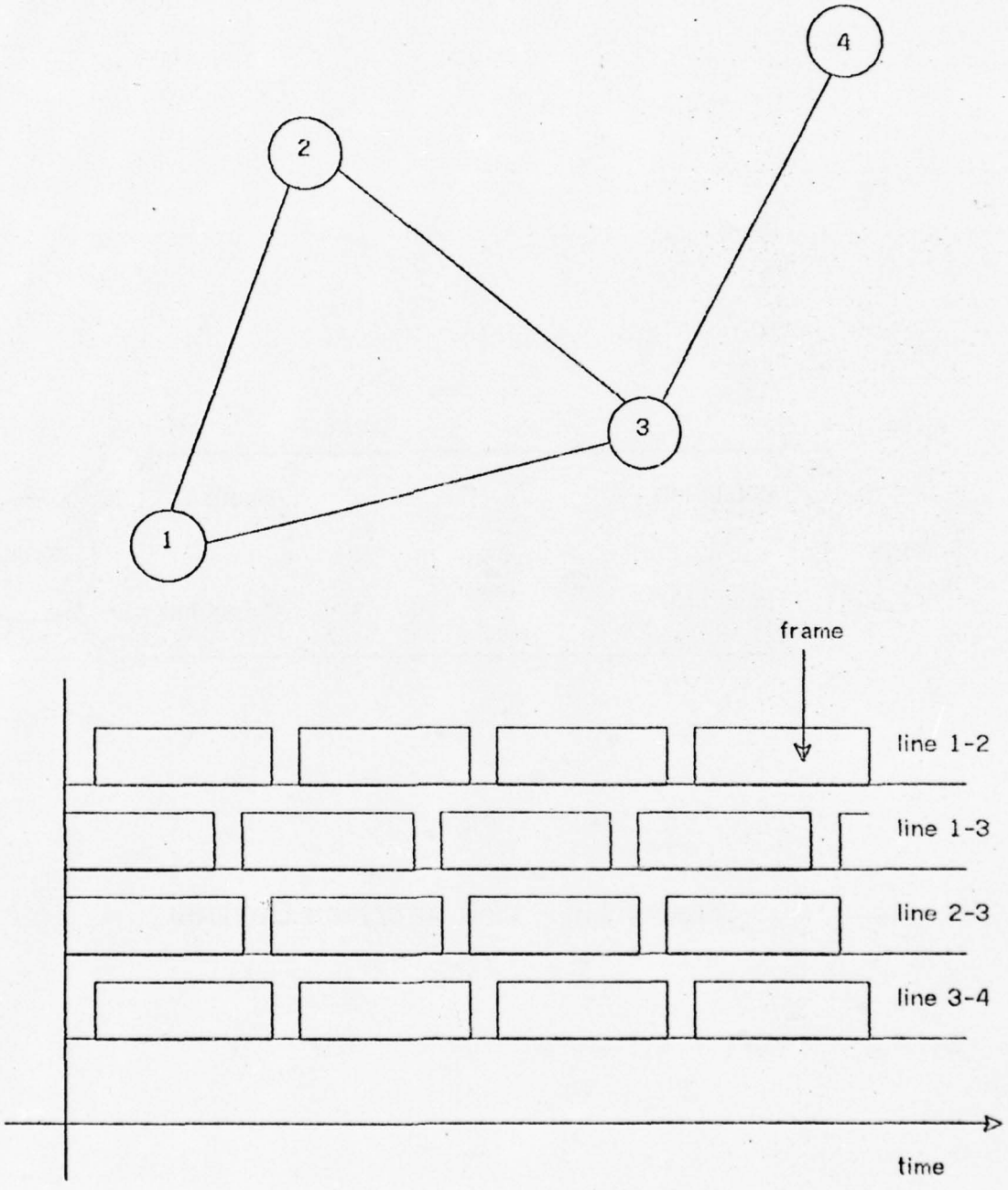


Figure 1.4 - Asynchronous Behavior of the Lines

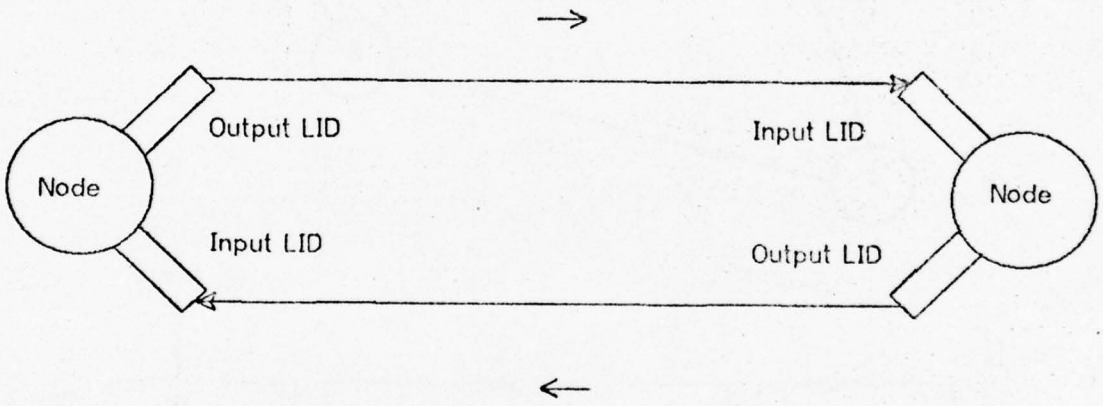


Figure 1.5 - Line Interface Devices

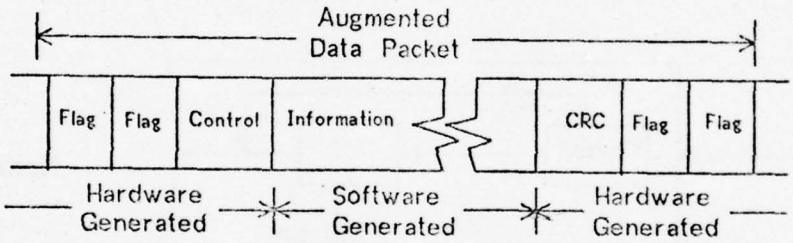
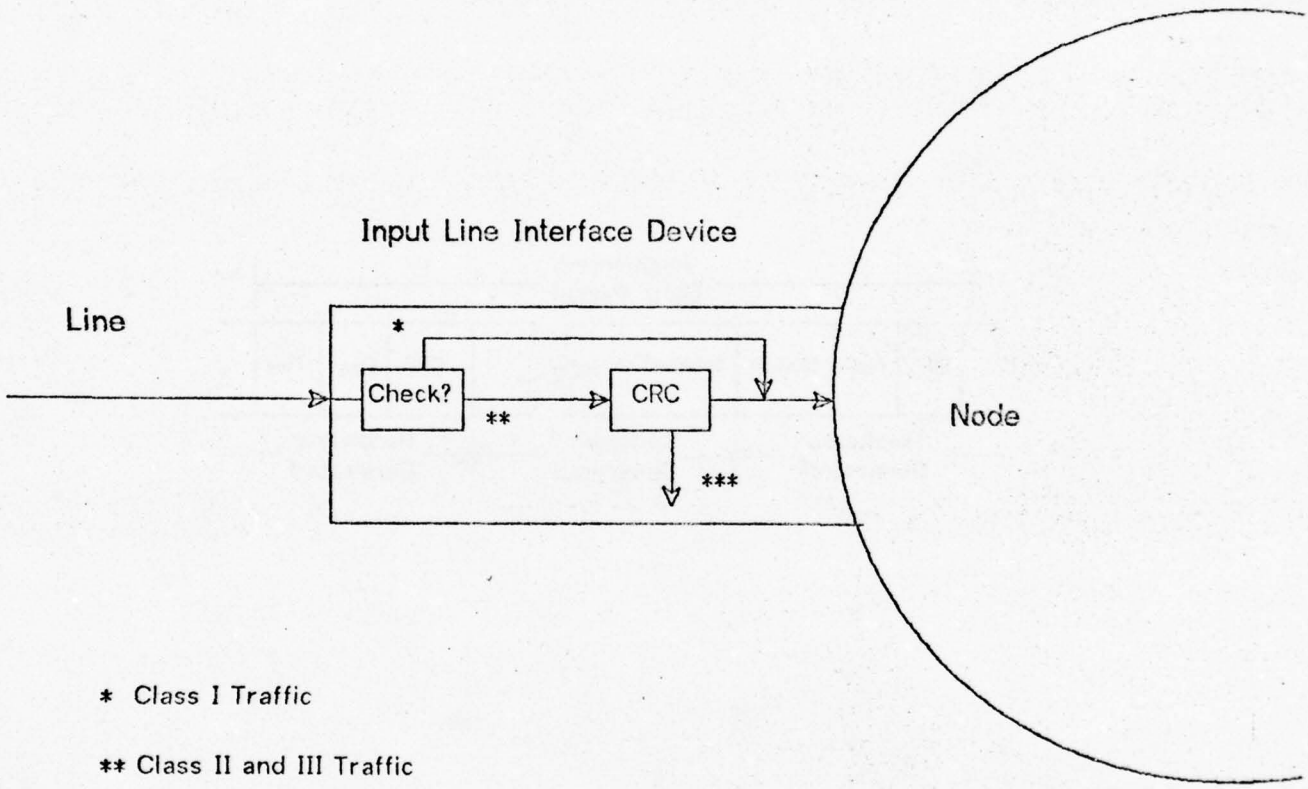


Figure 1.6 - Packet Transmission Format

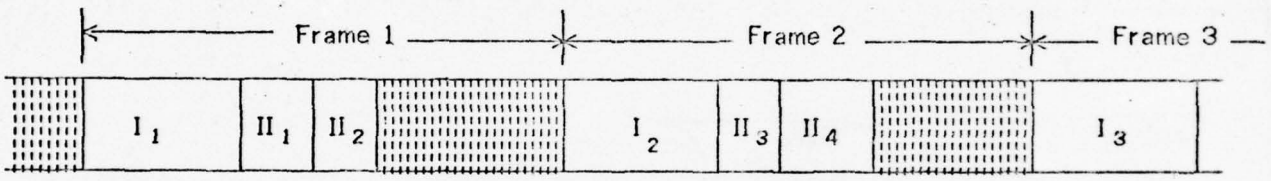


* Class I Traffic

** Class II and III Traffic

*** Class II and III Packets with Errors

Figure 1.7 - Handling of Class I Traffic by Line devices



← Direction of Flow

I_i = Class I Packets

II_j = Class II Packets

Shaded area = Idle characters

Figure 1.8 - Snapshot of Frame Transmission

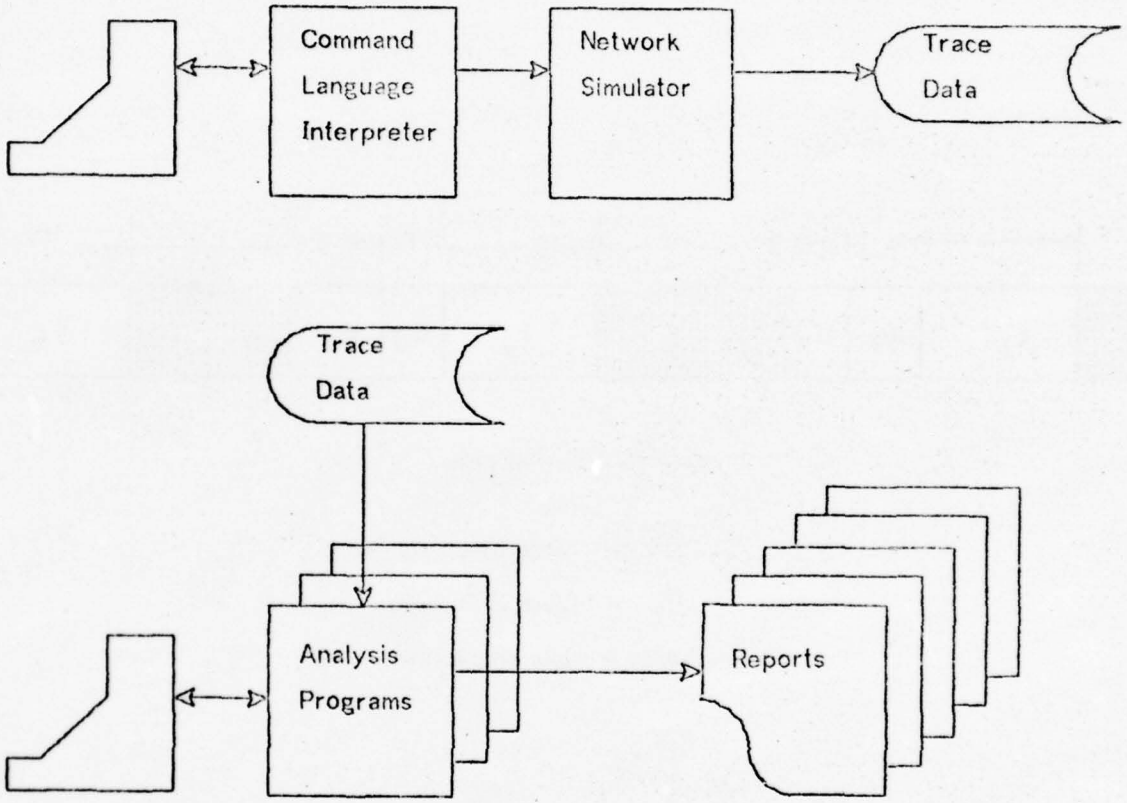


Figure 2.1 - Organization of the Network Simulation System

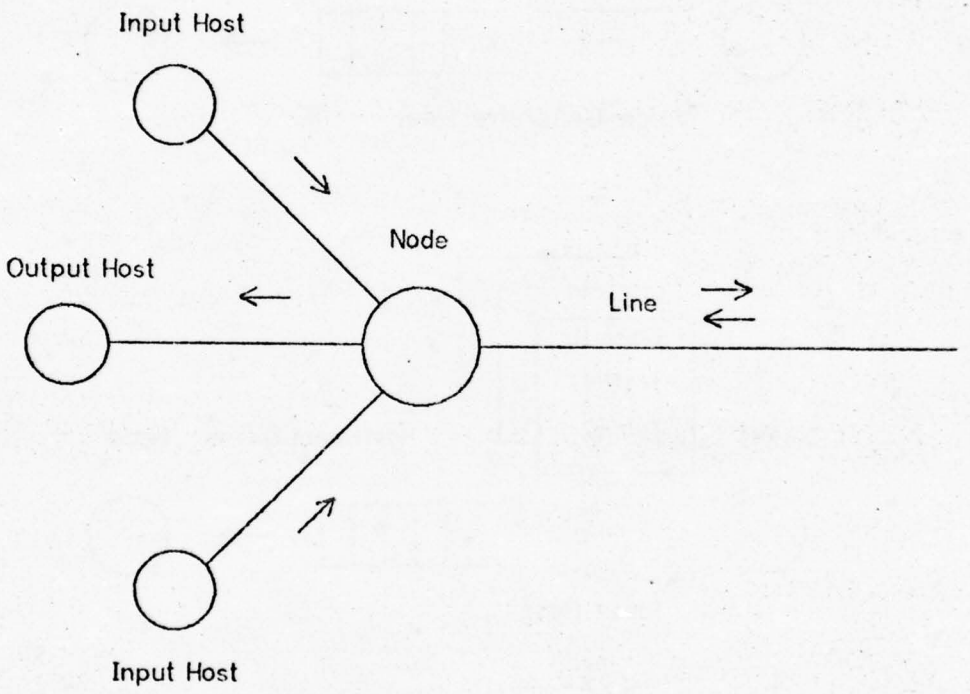


Figure 2.2 - Network Components

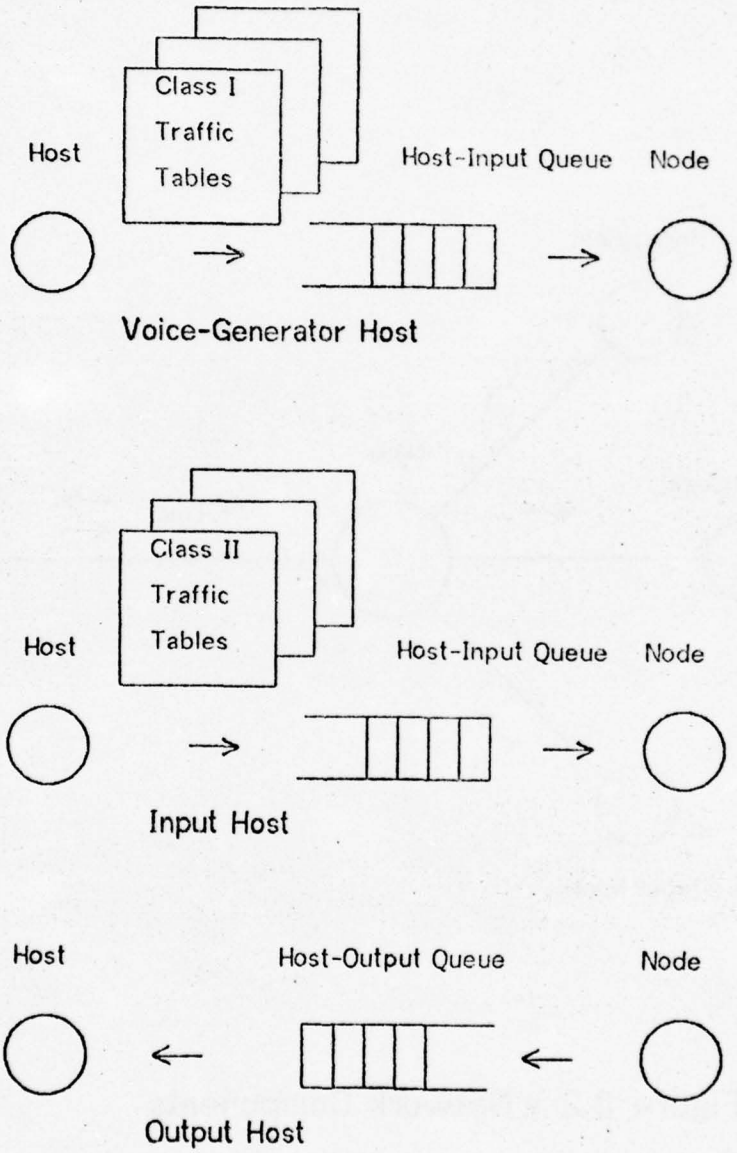


Figure 2.3 - Node Hosts

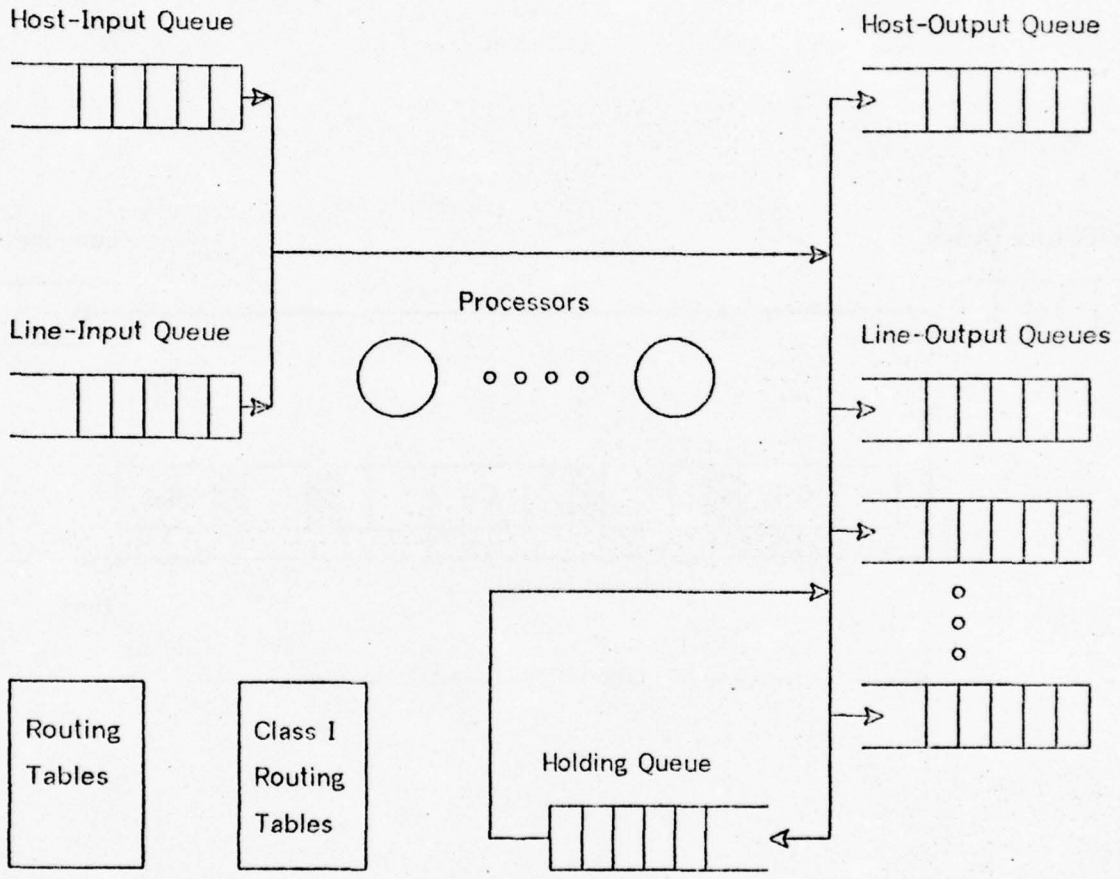


Figure 2.4 - Nodes

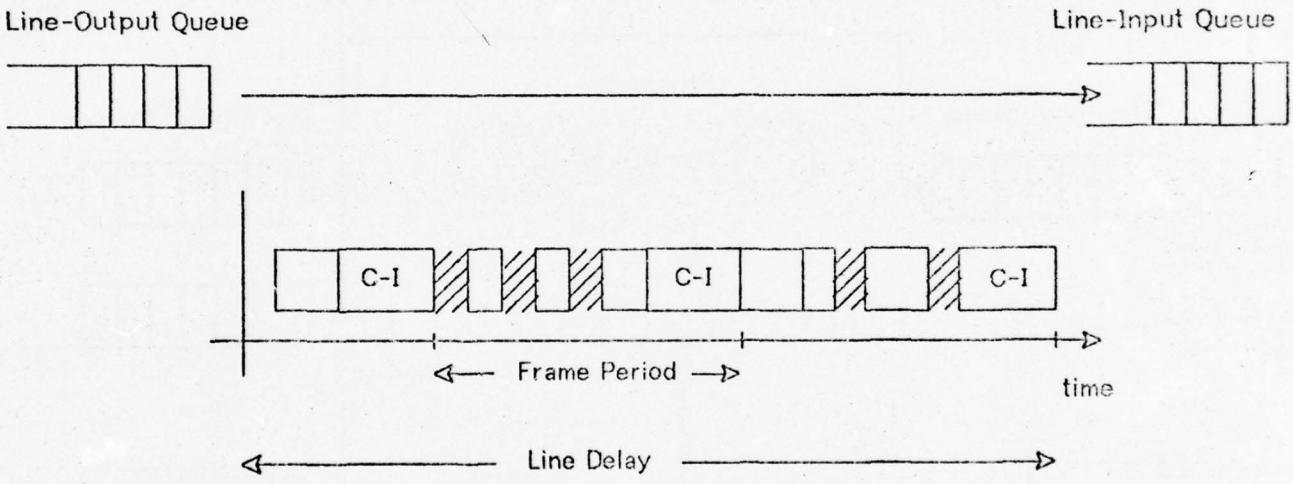


Figure 2.5 - Lines

Part II

Class I Traffic

Karem Sakallah
Mario R. Barbacci

1. Class I Traffic

This chapter describes the simulation of the synchronous, circuit-switched portion of the SENET master frame. As some projections indicate, Class I traffic will constitute the major part (approx. 87%) of the total traffic carried by an integrated network in 1985. It is, therefore, essential to pay particular attention to the design of an effective Class I protocol that is both simple and flexible.

In the following sections we describe the various types of Class I traffic, then indicate the major assumptions that we made in designing the Class I protocol. Next we discuss the protocol proper, and give some examples demonstrating its workability.

1.1. Class I Traffic Types

The integrated network will be required to accommodate and service a number of different Class I traffic types. The main categories are: voice, video, and facsimile. As far as the Class I protocol is concerned, the major differences among those types are:

1) different bandwidth requirements
2) one-way (e.g. facsimile) or two-way (voice) communication
3) different priorities

Furthermore, the voice traffic, which will constitute the bulk of the network traffic, will be generated by different vocoding techniques, leading to different bit rates. The network might have to provide facilities to connect subscribers with different rates and/or different vocoding schemes.

1.2. Implementation Issues

When we set out to implement the SENET scheme, we soon realized the need to

more precisely define a number of lower level details that were, out of necessity, left out in the original concept. In some cases compromises had to be made when a number of options presented themselves as possible solutions to a given problem. The following subsections attempt to put those considerations in perspective, laying down the background for the subsequent discussion of the protocol implemented in the simulator.

1.2.1 Class I Region Modification Protocol

As mentioned earlier, channel-control functions (adding and dropping Class I slots) will be carried out by special Class II control packets. This is necessary since the Class I region in the frame will not be checked for errors, whereas it is of the utmost importance to guarantee that the control information, that is used to decode the allocations in the Class I region, is error-free.

For proper operation, switches at both ends of a given link should have matching information about the Class I allocations in the frame. A fully interlocked protocol is, therefore, required to insure that changes in the Class I region are recognized simultaneously by both switches at the proper times.

Information about the active calls is kept in both switches in Class I Mapping Tables that are consulted on each frame arrival to determine where each incoming call should go (on which line, and on which position in the frame). Gaps that occur inside the Class I region are ignored by the receiving switch.

When the sending switch decides to alter the Class I region allocations by adding a new call or dropping an old one, it updates its Mapping Tables, sends a request for change to the receiving switch and waits for a positive acknowledgement before it considers that the receiving switch is aware of the change.

The receiving switch, in turn, ignores the unannounced change until it finally manages to process the control message that describes the nature of this change. Due to line errors, the time lapse between the occurrence of the change and its recognition by the receiving switch might be several frames (if the control packet was received in error, no acknowledgement is generated, and the sending switch times out and retransmits it). When the receiving switch updates its Mapping Tables appropriately, it sends an acknowledgement back to the sending switch. Receipt of this acknowledgement completes the transaction.

The above approach was favored to the other approaches that were suggested in [Bar76b]. The "time-absolute" approach requires a centralized master clock that provides a global time reference. It was mentioned earlier that reliability considerations advise against such a clock. The "time-relative" approach, on the other hand, can lead to "out-of phase" situations that might cause unpredictable delays in establishing the required change to the Class I region.

The reasoning that led to the development and adoption of the above protocol is quite simple. Realizing that the process of establishing a logical circuit between two subscribers is performed by a series of transactions between successive switches, it was concluded that the process can not be considered complete until all the transactions are successfully done. The suggested protocol insures this by requiring a new transaction to begin only after the previous one ended. The establishment of the logical circuit proceeds, therefore, one link at a time. Only when the whole path is clear end-to-end can it be used to carry the communication.

The only penalty in this technique is that a "dead space" is created from the time the sending switch incurs the change, until it finally receives an acknowledgement of

the change from the receiving switch. The slot involved in the change is unavailable during this period for either data transmission or other Class I allocations. Since this period will in general be on the order of a few frames, and since such a situation will only occur on call allocation and termination, the overhead that results will tend to be negligible.

1.2.2 Uni- vs. Bi-directional Control

The various channel control functions can be carried out in one of two ways: bidirectionally or unidirectionally.

In bidirectional control, those functions are performed, for two-way communications, simultaneously in both directions on a full-duplex link. This means that the forward path (from the call source to the call destination) and the backward path (from the destination to the source) are the same geographical route. Besides assuming full-duplex links (which might be a reasonable assumption to make), this method requires that all the switches on a given call path know whether the call is one- or two-way. Furthermore, some routing flexibility is lost because the forward and backward paths are constrained to follow the same route.

In unidirectional control, the channel functions proceed in one direction: from the call source to the call destination along the forward path, then (for two-way calls) from the destination to the source on the backward path, thus completing a closed loop. In many cases both paths might coincide, but it is conceivable that they might not. A heavy one-way load in one part of the network, or the existence of simplex links could lead to such a situation. Unidirectional control unifies the treatement of one- and two-way traffic since only the source and destination of a call need to know its type. In particular, the destination switch decides on whether to initiate reservation on the

backward path towards the source depending on the call type. This method allows also for greater routing flexibility than that provided by bidirectional control.

It should be obvious now that unidirectional control is both slower and and incurs more control overhead than bidirectional control. Nevertheless, we felt that those shortcomings are more than compensated for by the afore-mentioned advantages. Later we will show that the control overhead issue is irrelevant because it is less than 2% in any case. Similarly, if connection delays on the order of a few seconds are acceptable, then unidirectional control provides satisfactory performance. It is the method adopted in the simulator.

1.2.3 Routing Strategy

The routing of Class I traffic is carried out through the use of fixed Routing Tables that reflect the topology of the network. Each switch has a Routing Table whose entries indicate all the possible routes to all other switches, arranged in order of preference(i.e. primary route, then secondary route, and so on). When a switch tries to route a Class I call, it consults its Routing Table for enough capacity to carry the call on the primary route to the destination. If this fails, it tries the alternate routes successively.

The call path is determined one step at a time as the routing decision is passed from one switch to the next. Each switch tries to route the call to its final destination, regardless of where it is coming from. If after a path has been established from switch i to switch j, the latter switch fails to route the call, control reverts back to switch i and rerouting is attempted. If this fails too, blocking is indicated and the call is rejected. This limited routing capability was felt to be sufficient to provide adequate service. In later work, further routing considerations will be studied and compared, including adaptive routing as with packet-switched traffic.

1.2.4 Allocation/Compaction Methods

1.2.4.1 Compaction Problem

The Class I region modification protocol discussed above rests on the assumption that a change in the Class I allocations introduced by the sending node can be temporarily ignored by the receiving node. The implication of this assumption is that the sending node will not be allowed to drastically alter the Class I region in the frame in one operation. Rather, changes in this region should be performed gradually, one change at a time.

One immediate question arises: how can the Class I region compaction be performed when slots are dropped. If the constraint of one change at a time is not imposed, the obvious way to do compaction is to modify the starting position of all slots beyond the one being dropped. The Class I boundary is thus moved towards the beginning of the frame by an amount equal to the length of the dropped slot. Dropping and compaction are done simultaneously in this case.

This will not work for the proposed protocol. It creates a state of confusion for a period of time during which the switches at both ends of the link have inconsistent information. The solution to this problem is to do compaction by double transmission of the last slot in the Class I region. This is illustrated in Figure 1.1 for equal-length slots. The receiving switch will remain unaware of this double transmission until it receives and decodes the special "REPACK" control packet that describes the change that occurred. It, then, switches to the new slot position, updates its Mapping Tables, and responds with a positive acknowledgement. Upon receipt of this acknowledgement, the sending switch drops the old slot position and moves the Class I boundary towards the beginning of the frame appropriately.

This method works perfectly well for equal-length slots since any slot can exactly fit in the space created by dropping any other slot. With variable-length slots, however, the situation is more complicated. The existence of gaps in the Class I region can not be precluded completely because of partial fits. In addition, one compaction operation might not be sufficient to minimize the number and size of gaps. More will be said about this later.

1.2.4.2 Slot Quantization

Confronted with a Class I traffic consisting of a mixture of rates, we were naturally led to the following question: should we consider arbitrary slot lengths, or is it more advantageous to specify a "basic channel length" as the minimum transmission unit.

The management of arbitrary length slots tends to be cumbersome, especially if the previously described Class I region modification protocol is adopted. In addition, such a scheme would require bit addressability since the starting position of a slot can occur anywhere inside the Class I region.

We, therefore, considered the division of the Class I region into a number of equal-length transmission units, which will be called channels. A given call might require one or more contiguous channels depending on its rate. If the call requires a slot length that is not a multiple of the basic channel length, the source of the call appends a number of bits to the slot so that it fits in an integral number of channels. These extra bits add to the total over-head and should be recognized and removed by the destination switch. By properly choosing the basic channel length, the overhead introduced can be minimized.

1.2.4.3 Allocation Schemes Implemented

The simulator provides for three allocation options, that can be invoked by giving an

appropriate argument to the Command Language Interpreter command ALLOC. Other options and variations on the existing ones will be added later. A description of the current options is given below.

Allocation Scheme 1 (FFF)

This scheme handles Class I calls of a Fixed bit rate, and assigns them to Fixed-length channels(channel length corresponding to bit rate). The boundary of the Class I region is also Fixed. This scheme was the first to be incorporated in the simulator, and was used mainly for program debugging in the initial stages of the project. Because of the fixed bit rate restriction, this scheme is of no practical value.

Allocation Scheme 2 (VFF)

This scheme differs from the previous one in that it allows for an arbitrary Class I traffic composition (Variable bit rates). To allocate a given call the allocation algorithm searches for a number of contiguous channels that provide enough capacity to carry the call. Because of the fixed boundary, no compaction is performed. Upon dropping a given communication, the corresponding channels are flagged as available for other calls.

Allocation Scheme 3 (VFV)

This scheme allows the Class I region boundary to become flexible (Variable boundary). This brings in the compaction problem. In section 1.2.4.1 we illustrated how compaction can be done if all Class I calls require equal-length slots. We consider now arbitrary-length slots.

When compaction by double transmission is employed in this case, gaps will unavoidably appear in the Class I region. Allocation of new calls by appending them to the ongoing ones (as would be done if there are no gaps) will eventually push the

boundary to its limit, and cause further traffic to be rejected, even though the major portion of the Class I region is wasted in useless gaps. There are basically two ways around this problem: repeated compaction or a "hybrid allocation" algorithm.

In repeated compaction, dropping a slot starts a sequence of compactions that take the last slots in the Class I region in succession and fit them in the space freed by the dropped slot. This helps reduce the number and size of the gaps and, consequently, new calls can be always appended to the Class I region.

By "hybrid allocation" we mean that in trying to route a call, the allocation algorithm first tries to fit the call inside the Class I region (in one of the gaps) before it considers appending it to the region. It thus combines the features of a fixed-boundary scheme and a "pure" flexible-boundary scheme.

The simulator implements the "hybrid" scheme at present. The other option will be added to it later.

1.2.5 Service Restrictions

A number of service features have been left out of the simulator in this phase of the project. These include preemption policies, security considerations (encrypted voice, e.g.), and the possibility of connecting noncompatible subscribers(with different rates and/or vocoding methods).

In addition, the protocols are built on the assumption that the initiation of channel reservation and channel release is done by the caller, while channel allocation is started at the destination switch when the called party goes off-hook.

1.3. Class I Protocols

This section describes the various channel-control messages that were defined and implemented in the simulator.

1.3.1 Class I Call Phases

A Class I communication can be conveniently viewed as composed of three distinct phases: reservation, allocation and termination. A brief description of each of these phases follows.

1.3.1.1 Reservation Phase

During this phase a path for the call is established between the calling and the called parties. The path is determined as was described in section 1.2.3 . Reservation is initiated by the caller. If it is completed successfully, the source switch sends a ringing signal to the called party and a ringing tone to the calling party. If the called party line is engaged or the network is congested, so that further reservations cannot be carried through, all previous reservations are cancelled, and a busy tone is returned to the caller.

The Class I region is not disturbed during this phase. A successful reservation causes a RESERVATION NOTICE to be kept in the switch until allocation time. The NOTICE contains the following information:

- Call number
- Input line (line it is coming from)
- Output line (line it is leaving on)
- Output channel(s)

A value of -1 for output channel indicates that the call should be appended to the Class I region.

It is important to realize (for flexible Class I boundary allocation schemes) that until allocation is done the reserved channels are available for non-Class I traffic . This is also true for fixed boundary schemes if the gaps inside the Class I region are employed to carry non-Class I traffic.

1.3.1.2 Allocation Phase

This phase begins when the called party goes off-hook. The information kept in the RESERVATION NOTICES in the switches on the call path, is used during this phase to update the Mapping Tables and add the reserved channels to the Class I region. The NOTICES are deleted, then, and the Mapping Tables keep track of any later relocation of those channels (e.g. during compaction)

1.3.1.3 Termination Phase

When the caller decides to quit, he sends a message to his local switch requesting call termination (i.e. he hangs up).This starts the termination sequence. For fixed-boundary schemes this amounts to recovering the channels that were carrying the call to the pool of available channels. For variable- boundary schemes, compaction of the Class I region might be necessary.

1.4. Channel Control Messages

We define here the control packets that implement the Class I protocol described in previous sections. As remarked earlier, the information field that comprises the body of a control packet is represented in the simulator by an array called ctlmessage. Each control packet in the simulator has such an array as one of its attributes. The array has six elements, the first of which serves to identify a particular control action. The remaining five elements provide other information needed to carry out the specific action.

The ctlmessage fields for the eight Class I control packets are given below. The packets have been grouped according to function in correspondence with the above Classification of call phases.

CONTROL PACKET	ctlmessage[i]					
	[1]	[2]	[3]	[4]	[5]	[6]

A)Reservation

RESERVE	2	slot	rate	path	call	fwd slot
UNRESERVE	3	pkt	action	N/A	call	N/A
CANCEL	4	slot	N/A	N/A	call	N/A
RING	8	slot	rate	origin	call	N/A

B)Allocation

ALLOCATE	5	slot	rate	slotpos	call	N/A

C)Termination

RELEASE	6	slot	slotpos	N/A	call	N/A
REPACK	7	slot	newpos	oldpos	call	N/A
CONFIRM	1	pkt	N/A	N/A	call	N/A

Legend

pkt : request packet (RESERVE, CANCEL, ALLOCATE, RELEASE, REPACK) number
path: call path +2 Forward path, 2-way call
+1 Forward path, 1-way call
-1 Backward path, 2-way call
rate: bit rate, bits/sec
slot : unique slot number
action: +1 Reroute
-1 Cancel
slotpos: slot position in input frame
origin: node number of the call source
call : unique call number
newpos: new slot position in frame after repacking
oldpos: old slot position in frame before repacking
N/A: not applicable

Figures 1.3 to 1.9 display the control flow for each of the above packet types. The terminology used in those figures is defined in Figure 1.2.

1.5. Demonstration Facilities

A demonstration aid was developed to help illustrate and verify the various Class I protocols implemented in the simulator. The demonstration program takes the trace produced by the simulator as its input. It, then, shrinks it retaining only those events needed to follow the Class I call setup and breakdown. In addition, it gives snapshots

of the Class I region in the frame showing the allocation of the Class I channels to the ongoing calls, and displaying the dynamic boundary between Class I and Classes II and III.

1.5.1 Trace Messages

Most of the trace entries that the demonstration program retains have been described previously (see). A brief description of the remaining entries follows.

1.5.1.1 CREATE CTL

The CREATE CTL entries have the following form:

<time> CREATE CTL <packet> <source> <destination> <msg1> <msg2> <msg3> <msg4> <msg5> <msg6> <length>

This entry describes the event of a control-packet creation by a SENET host. For Class I simulations, the creation of such packets is the means by which the host requests the network's attention. The parameters identify the packet, its source and destination, the ctlmessage vector, and the packet length. Msg1 specifies the particular action that the host is requesting. For Class I traffic, these actions are: call initiation (RESERVE packet, msg1=2), call allocation (ALLOCATE packet, msg1=5), and call termination (RELEASE packet, msg1=6).

1.5.1.2 GEN CTLPKT

The GEN CTLPKT entries have the following form:

<time> GEN CTLPKT <packet> <source> <destination> <sender> <receiver> <msg1> <msg2> <msg3> <msg4> <msg5> <msg6> <length>

This entry describes the generation of a control packet by a SENET node in response to the receipt of another control packet. The other packet might be coming

from a host or from another node. The parameters identify the packet, its source and destination, its immediate sender and receiver, the ctlmessage vector, and the packet length. Msg1 identifies the particular control action the current node is requesting from the receiver of the packet.

1.5.1.3 NEW RESREV

The NEW RESERV entries have the following form:

<time> NEW RESERV <call> <slot> <from> <to>
<outputline> <output slot position>

This entry describes the successful completion of a reservation for a call between two nodes. The parameters identify the call and the slot(two unique numbers), the link on which the reservation has been made(by specifying the nodes at both of its ends). The last two parameters identify the output line number(lines connected to a certain node have local identification numbers that go from 1 to the total number of lines connected to the node) and the slot position on that line. A slot position of -1 indicates that the slot is to be appended to the Class I region at allocation time.

1.5.1.4 NEW ALLOC

The NEW ALLOC entries have the following format:

<time> NEW ALLOC <call> <slot> <from> <to>
<slot length> <inline> <inpos> <outline> <outpos>

This entry appears after the completion of an allocation. The parameters identify the call, and the slot, the link(<from> to <to>), the slot length (in bits). The last four parameters specify the input line number (its local identification number), the channel position of the slot on that line, the output line number, and the channel position of the slot on that line. Thus the last four parameters specify the mapping of the slot from the input to the output lines.

1.5.1.5 TALK

The TALK entries have the following form:

<time> TALK <call> <source> <destination>

This entry appears when allocation has been completed on the backward path of a call, and is starting on the forward path. It is issued after the source node has disconnected the ringing tone from the caller's line and indicated that the communication may start. The parameters identify the call, its source, and destination.

1.5.1.6 BLCK

The BLCK entries have the following form:

<time> BLCK <call> <source> <destination> <blocked at node>

This entry describes the event of blocking of a call. The parameters identify the call, its source and destination, and the node at which the blocking occurs.

1.5.2 Examples

Appendix A contains two examples of the Class I protocol. The first example, a two node network, illustrates all of the control functions except for routing. The second example illustrates routing and unidirectional control. In particular, it shows instances when the forward and backward paths of a call do not coincide.

2. Class I Performance Analysis

This chapter describes the statistical analysis that follows a simulation run to determine a number of performance measures for the Class I traffic. The data analysis program is described first. The latter part of the chapter describes the experiments that were conducted and the results obtained.

2.1. Data Analysis

As mentioned earlier, statistics are compiled subsequent to the simulation run. A record of the run is kept in the form of a trace file that can be examined (repeatedly if necessary) by data analysis programs. This section describes the above process for Class I traffic.

2.1.1 Class I Data Analysis Program

The Class I data analysis program reads successive records from a simulation trace file, processes them, and produces an output file containing the desired statistics.

The information contained in the trace, is compressed and summarized in the form of a list of all calls made during the simulation. Each entry in the list contains the following items:

1) Call number
2) Source
3) Destination
4) Bit rate
5) Time of initiation
6) Whether blocked or successfully completed
7) If blocked, at which node
8) Number of hops

9) Associated overhead

10) Connection time (length of reservation period)

11) Holding time.

As the program scans through the trace file records, it discards irrelevant events, and uses other events to update the call list. A new entry is added to the list each time a new call is initiated. (This is detected whenever a CREATE CTL message that refers to a RESERVE control packet is found). When all records have been read, the list is searched for unfinished calls(those still in the reservation phase), and the corresponding entries are deleted. The call list can then be analyzed and a number of statistics computed.

2.1.2 Estimation and Elimination of Initial Bias

A simulation run starts with a network in an idle state in which all Class I channels are free. Early calls, therefore, experience an unusually low blocking probability and cause the computed statistics to be biased. As the length of the run increases, the law of large numbers renders those early calls insignificant. It is desirable nevertheless to determine the length of this initial transient interval and to discard it as this tends to eliminate the need for a very long run.

To achieve this , the Data Analysis Program samples the number of busy channels in the class I region on all the lines at regular intervals. At the same sampling times, it examines the list of calls made and computes all the statistics of interest.After the trace file has been read, N samples of all the statistics are available.

The samples for the number of busy channels are used ,then, to calculate N sample means and their associated standard deviations, each based on the <u>first</u> i samples, where i assumes the values 1 to N. Those are tabulated as a function of time (which is

the same as tabulating them as a function of the number of samples since sampling is done at regular intervals). Furthermore, the program plots the mean number of busy channels as a function of time. By examining this curve, one can determine the onset of steady state. Better yet, if the standard deviation is plotted against time on log-log scales, the end of the transient period can be detected when the curve starts sloping down at a rate of -1/2 (corresponding to an inverse square root relation ; see [Gor69]).Either way,a point in time can be found which can be regarded as the end of the initial bias period. All statistics of interest have to exclude this period.

The N samples of each statistic are used in an analogous manner to those of the number of busy channels. N means are calculated. This time, however, each of these means is based on the last i samples, where i is 1 to N. In effect, each of these means discards an initial bias interval proportional to (N-i) .

With all this data available, a simple correlation between the curves for the number of busy channels, and the tables for the other statistics determines which set of statistics to quote. This procedure is illustrated in appendix B.

2.1.3 Statistics Gathered

We define here the various performance measures that are produced by the data analysis program.

2.1.3.1 Blocking Probabilities

Blocking (or Loss) probability PL is defined as follows:

PL = Number of blocked calls/Total number of attempted calls

A different PL is calculated for each of the Class I bit rates, and a composite PL is computed for all Class I traffic.

2.1.3.2 Allocation Scheme Efficiency

Efficiency Eff gives an indication of the amount of overhead associated with a certain allocation strategy. It is defined below:

Eff = Total information bits transmitted/(Tot info bits+Tot overhead bits)

The overhead associated with a Class I call can be broken down into the following four categories:

1) Control-packet overhead

2) Dead-space overhead (due to the step by step frame modification protocol)

3) Double-transmission overhead

4) Quantization overhead (the extra bits padded to a slot to make it fit an integral number of channels)

Everything eise, including silence gaps that might occur during a Class I communication, is considered pure information.

2.1.3.3 Class I Region Utilization

Utilization U is defined as follows:

U = Average number of busy channels/Total number of channels

2.1.3.4 Average Capacity available for Non-Class I Traffic

This is the capacity C (in bits/frame) that remains after the Class I requirements have been satisfied. More precisely,

C = Total bits in the frame minus

(avg Class I region length+avg Class I control packet requirements)

This measure assumes that Class I traffic has precedence over other traffic classes, and gets allocated first. The remaining capacity C is what is left for packet switched data and bulk traffic, and their associated control.

2.1.4 Output Report

The output of the data analysis program consists of the following parts:

2.1.4.1 Experiment Parameters

This part is basically a copy of the first page in the trace file. It includes a description of the network, and all the parameters that were used in the simulation run, including the simulation period (in milliseconds). The format of this part is the same as that produced by the Command Language Interpreter SHOW command described earlier. This part serves to identify the particular run for later reference and comparison with other runs.

2.1.4.2 List of Simulated Calls

This section provides an extensive listing of all the calls that occurred during the simulation. Each entry in this list has the following fields:

<call>	unique call number
<origin>	node number at which call was originated
<destination>	node number of call destination
<rate>	bit rate of call in KBPS
<connection time>	duration (in milliseconds) of the reservation phase
<holding time>	duration of the call in minutes
<number of hops>	average number of links traversed (this is one half the sum of the lengths of the forward and backward paths)
<blocked at node>	if the call is blocked, the node at which blocking occurred is indicated here

2.1.4.3 Node Activities

In this part, calls are classified by their source nodes and tabulated as follows:

<node > <total calls attempted> <completedcalls> <blocked calls>

2.1.4.4 Statistics

This section contains the statistics defined previously. They are tabulated for various initial-bias interval lengths. It also cotains a table of the mean number of busy channels in a frame, and its standard deviation for different sample sizes. Finally, it contains a plot of the mean number of busy channels as a function of time.

2.1.5 Sample Output Report

Appendix B illustrates a typical output report from the Data Analysis Program. It describes, too, the procedure used to determine the length of the initial bias interval.

2.2. Class I Experiments

This section describes the set of experiments that we conducted to evaluate the performance of the network for Class I traffic.

2.2.1 Data Base

The experiments are based on the following voice traffic mix [Sylvania76].

Trans. Rate	Percent	@[Remarks]
2400 bits/sec	10%	Vocoder
4000	10%	Linear Predictive Coding
8000	15%	Adaptive Predictive Coding
16000	50%	Continuous Voice Delta Modulation
32000	10%	Continuous Voice Delta Modulation
50000	5%	Secure Voice PCM

In the experiments the only Class I traffic that is simulated is voice traffic. All future references to Class I traffic are to interpreted accordingly.

2.2.2 Common Parameters

The following experiment parameters are shared by all the experiments described later.

2.2.2.1 Network Topology

The network used for those experiments consists of two nodes connected by a T1 carrier. Each node has one processor capable of moving a 16-bit word in 5 microseconds.

2.2.2.2 Frame Period

A 10-millisecond frame period is assumed throughout.

2.2.2.3 Traffic Parameters

The host of node 1 generates 60 Erlangs of voice traffic, destined to node 2. The average call duration is 5 minutes. The average responce time of the called party to answer a call is taken as 10 seconds. The host of node 2 generates no traffic.

2.2.2.4 Simulation Mode

The simulator is run in the VOICE mode. In this mode, only Class I traffic is generated and processed.

2.2.2.5 Basic Channel Length

The Class I channels are based on a 4 KBPS bit rate. This corresponds to 40-bit channels in a 10-millisecond master frame.

2.2.2.6 Control Packet Length

All control packets are assumed to be 100 bits long.

2.2.2.7 Simulation Length

The simulation length in all the experiments is 45 minutes. Several pilot runs were necessary to determine an appropriate length. The run has to be long enough so that steady state is reached.

2.2.3 Experiment Variables

The voice fraction in the master frame was varied from 0.5 to 0.9 in steps of 0.1 . This required five runs for each of two allocation methods: the fixed (VFF) and flexible (VFV) boundary schemes.

2.2.4 Results and Conclusions

2.2.4.1 Efficiency of the Allocation Schemes

Figures 2.1 and 2.2 show the variation of the allocation scheme efficiency Eff as a function of the voice fraction in the frame. Three conclusions can be drawn from those two figures:

1) The overhead due to control packets and slot quantization is less than 2%

2) The efficiency does not vary significantly with the allocation method employed

3) The efficiency does not vary significantly with the size of the Class I region in the frame

The above results, then, justify the use of quantization as opposed to arbitrary length slots. They also indicate that the control packet over head introduced by the frame modification protocol (which is based on unidirectional step-by-step control) is negligible.

2.2.4.2 Blocking Probabilities

Figures 2.3 to 2.5 show the variation of the various blocking probabilities as a function of the voice fraction in the frame. It is evident that more data points are needed in order to be able to measure a statistically significant blocking probabilities . Nevertheless, the following conclusions can be inferred:

1) As the voice rate increases, the probability of its being blocked increases. In particular, during the 45-minute runs no 2.4, 4 or 8 KBPS were blocked. This implies that "high-quality" voice is more

prone to being blocked and a preemption policy has to be developed to insure an adequate grade of service for it.

2) The results do not allow any inference to be made as to the effect of the allocation method on the blocking probabilities.

3) In general, the measured overall blocking probabilities are smaller than the approximate one obtained from the Erlang loss formula by assuming that all call connection requests require an "averge" channel-length as determined from the given traffic mix. for the specific mix at hand, this "normalized" channel has a length of 155.4 bits per 10 millisecond frame on a T1 carrier.

2.2.4.3 Class I Region Utilization

Figure 2.6 shows the variation of the utilization U as a function of the voice fraction in the frame.It displays for both allocation schemes the right trend of decreasing as the available capacity increases. A possible reason for the higher utilization observed for the flexible boundary scheme is double transmission during compaction. The utilization predicted by the Erlang formula (for the normalized voice channels) is plotted for comparison.

2.2.4.4 Average Capacity for Non-Class I traffic

Figure 2.7 shows how the capacity available for data and bulk traffic varies with the voice fraction in the frame. The straight line relationship for the fixed boundary scheme indicates that the class I control-packet overhead is extremely small.

For the flexible boundary scheme, the location of the boundary depends on two factors: the offered load, and the limit voice fraction. For small voice fractions, the class I region will be almost fully utilized (refer to Figure 2.6), and the location of the boundary will be determined by the voice fraction. This explains the initial linear portion of the curve. As the voice fraction increases the utilization of the class I region drops and the location of the boundary becomes governed by the class I load. Since we are assuming a fixed load, the curve bends and tends to a constant value. The particular value to which the boundary adjusts depends on the magnitude of the load.

This figure clearly shows the benefits of a flexible boundary scheme over a fixed boundary scheme.

Appendix A: Class I Protocol Examples

This appendix contains two detailed examples of the Class I protocol. Each example consists of a trace of the relevant events and snapshots of the frame maps showing the channel allocations. Additional comments are added to make it easier to follow the control sequence.

A.1. Example 1

This example illustrates the basic Class I protocol except for routing. It is based on a network of two nodes, in which node 1 is sending voice traffic to node 2. The link between the nodes is 1000 KBPS. The voice fraction is fixed at 0.1, producing a Class I region that can grow up to 1000 bits/10-millisecond frame. The basic channel rate is 20 KBPS. The maximum number of channels, therefore, is 5 (200 bits/channel). The voice load is equally divided between two rates: 20 KBPS (requiring one channel per call) and 30 KBPS (requiring two channels per call). The boundary is flexible, and no data traffic is generated.

The following comments should be correlated with the trace that follows.

TIME	EVENT
199823.453	Host 1 initiates call 1 (30 KBPS) by sending RESERVE packet # 1 to node 1
199823.484	The processor in node 1 attends to RESERVE packet # 1 after it is removed from the input queue. It decodes the control action and files a RESERVATION NOTICE for call 1 after it makes a reservation on line 1-2. It, then, sends RESERVE packet # 2 on this line to node 2.
199830.131	Node 2 receives RESERVE packet # 2, makes a reservation back on line 2-1. It acknowledges the receipt of packet # 2 by sending back CONFIRM packet # 3. It finally sends RESERVE packet # 4 on line 2-1 to node 1.
199840.131	Node 1 receives CONFIRM packet # 3, and responds by removing packet # 1 from its holding queue.
199840.230	Node 1 receives RESERVE packet # 4. It acknowledges by sending back

CONFIRM packet # 5. Recognizing that the reservation phase has been successfully completed, it sends RING packet # 6 to node 2.

199850.131 Node 2 receives CONFIRM packet # 5 and removes packet # 4from its holding queue.

199850.230 Node 2 receives RING packet # 6, confirms it (packet # 7), and sends a ringing tone to the called party.

202970.998 The called party goes off-hook, and sends ALLOCATE packet # 8 to node 2.

202971.029 Node 2 connects the called party to channels 1 and 2 on line 2-1. It, then, sends ALLOCATE packet # 9 on this line to node 1.

202980.531 Node 1 receives ALLOCATE packet # 9 and responds by connecting the calling party to channels 1 and 2 on line 1-2. It, then, sends ALLOCATE packet # 10 on line 1-2. Finally, it sends CONFIRM packet # 11 to node 2.

202990.531 Node 2 receives ALLOCATE packet # 10, and recognizing that the allocation phase is over, it merely acknowledges its receipt by sending CONFIRM packet # 12 back to node 1.

202990.631 Node 2 receives CONFIRM packet # 11 and responds by removing packet # 9 from its holding queue.

203000.531 Node 1 receives CONFIRM packet # 12, and responds by removing packet # 10 from its holding queue.

245690.465 The caller hangs up, and sends RELEASE packet # 13 to node 1.

245690.496 Node 1 receives RELEASE packet # 13, starts sending "silence" on channels 1 and 2, and sends RELEASE packet # 14 to node 2.

245700.531 Node 2 receives RELEASE packet # 14, starts sending "silence" on channels 1 and 2 on line 2-1.It then sends RELEASE packet # 15 and CONFIRM packet # 16 to node 1.

245710.531 Node 1 receives RELEASE packet # 15, and responds by sending CONFIRM packet # 17 to node 2.

245710.631 Node 1 receives CONFIRM packet # 16 acknowledging RELEASE packet # 14. It knows now that node 2 is aware of the fact that call 1 is over and is therefore ignoring channels 1 and 2 on line 1-2. Node 1 frees those channels now and moves the boundary of the Class I region towards the beginning of the frame (it happens that Class I region becomes empty now).

245720.131 Node 2 receives CONFIRM packet # 17 and responds by dropping channels 1 and 2 on line 2-1 and moving the boundary accordingly.

613145.532 Call 2 is initiated. The sequence of events for its reservation and allocation parallels the one just described for call 1.

710789.578 Call 3 is initiated. It is allocated on channels 3 on both the forward and the backward paths.

806262.313 Call 4 is initiated. It is allocated on channels 4 and 5 .

872723.727 Call 5 is initiated.

872723.758 Call 5 is blocked when node 1 finds that all the Class I capacity is currently allocated.

907486.320 The termination phase of call 3 is started.

907501.234 Node 1 receives CONFIRM packet # 58 acknowledging RELEASE packet # 56, and frees channel 3 on line 1-2 (this is indicated by a "*" in the frame map). This introduces a gap, which it fails to fill by double transmitting call 4 because call 4 requires two channels. No compaction is done, therefore.

1012868.664 The termination phase of call 2 begins.

1012881.234 Channels 1 and 2 are freed. Call 4 is double transmitted, and REPACK packet # 65 is sent to node 2.

1012901.234 Node 1 receives CONFIRM packet # 67 acknowledging REPACK packet # 65. It knows now that node 2 is receiving call 4 on the new channel positions (1 and 2). Therefore, it compacts the Class I region by dropping the old channel positions (4 and 5) and adjusting the boundary accordingly.

```
199823.453  CREATE CTL    1  1  2  2     1 30000     2     1     0   100
199823.484  REC CTLPKT    1  1  2  1  1  2     1 30000     2     1     0
199823.484  New Reserv    1     1  1  2  1    -1
199823.484  GEN CTLPKT    2  1  2  1  2  2     1 30000     2     1     0   100
199830.131  REC CTLPKT    2  1  2  1  2  2     1 30000     2     1     0
199830.131  New Reserv    1     2  2  1  1    -1
199830.131  GEN CTLPKT    3  2  1  2  1  1     2     0     0     1     0   100
199830.131  GEN CTLPKT    4  2  1  2  1  2     2 30000    -1     1     1   100
199840.131  REC CTLPKT    3  2  1  2  1  1     2     0     0     1     0
199840.230  REC CTLPKT    4  2  1  2  1  2     2 30000    -1     1     1
199840.230  GEN CTLPKT    5  1  2  1  2  1     4     0     0     1     0   100
199840.230  GEN CTLPKT    6  1  2  1  2  8     2 30000     1     1     0   100
199850.131  REC CTLPKT    5  1  2  1  2  1     4     0     0     1     0
199850.230  REC CTLPKT    6  1  2  1  2  8     2 30000     1     1     0
199850.230  GEN CTLPKT    7  2  1  2  1  1     6     0     0     1     0   100
199860.131  REC CTLPKT    7  2  1  2  1  1     6     0     0     1     0
202970.998  CREATE CTL    8  2  1  5     2 30000     0     1     0   100
202971.029  REC CTLPKT    8  2  1  2  2  5     2 30000     0     1     0
202971.029  New Alloc     1     2  2  1   400  0     0  1     1
```

FRAME MAP FOR LINE 2 - 1

```
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 1|
          -------
```

```
202971.029  GEN CTLPKT    9  2  1  2  1  5     2 30000     1     1     0   100
202980.531  REC CTLPKT    9  2  1  2  1  5     2 30000     1     1     0
202980.531  New Alloc     1     1  1  2   400  0     0  1     1
```

FRAME MAP FOR LINE 1 - 2

```
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 1|
          -------
```

```
202980.531  GEN CTLPKT   10  1  2  1  2  5     1 30000     1     1     0   100
202980.531  GEN CTLPKT   11  1  2  1  2  1     9     0     0     1     0   100
202980.531  TALK          1  1  2
202990.531  REC CTLPKT   10  1  2  1  2  5     1 30000     1     1     0
202990.531  GEN CTLPKT   12  2  1  2  1  1    10     0     0     1     0   100
202990.631  REC CTLPKT   11  1  2  1  2  1     9     0     0     1     0
203000.531  REC CTLPKT   12  2  1  2  1  1    10     0     0     1     0
245690.465  CREATE CTL   13  1  2  6     1     0     0     1     0   100
245690.496  REC CTLPKT   13  1  2  1  1  6     1     0     0     1     0
245690.496  GEN CTLPKT   14  1  2  1  2  6     1     1     0     1     0   100
245700.531  REC CTLPKT   14  1  2  1  2  6     1     1     0     1     0
245700.531  GEN CTLPKT   15  2  1  2  1  6     2     1     0     1     0   100
245700.531  GEN CTLPKT   16  2  1  2  1  1    14     0     0     1     0   100
245710.531  REC CTLPKT   15  2  1  2  1  6     2     1     0     1     0
245710.531  GEN CTLPKT   17  1  2  1  2  1    15     0     0     1     0   100
245710.631  REC CTLPKT   16  2  1  2  1  1    14     0     0     1     0
```

```
FRAME MAP FOR LINE   1 - 2
NO ONGOING CALLS AT PRESENT
     245720.131  REC CTLPKT    17  1  2  1  2  1    15     0     0     1     0


FRAME MAP FOR LINE   2 - 1
NO ONGOING CALLS AT PRESENT
     613145.523  CREATE CTL    18  1  2  2     3 30000     2     2     0   100
     613145.555  REC CTLPKT    18  1  2  1  1  2     3 30000     2     2     0
     613145.555  New Reserv     2     3  1  2  1    -1
     613145.555  GEN CTLPKT    19  1  2  1  2  2     3 30000     2     2     0   100
     613150.133  REC CTLPKT    19  1  2  1  2  2     3 30000     2     2     0
     613150.133  New Reserv     2     4  2  1  1    -1
     613150.133  GEN CTLPKT    20  2  1  2  1  1    19     0     0     2     0   100
     613150.133  GEN CTLPKT    21  2  1  2  1  2     4 30000    -1     2     3   100
     613160.133  REC CTLPKT    20  2  1  2  1  1    19     0     0     2     0
     613160.234  REC CTLPKT    21  2  1  2  1  2     4 30000    -1     2     3
     613160.234  GEN CTLPKT    22  1  2  1  2  1    21     0     0     2     0   100
     613160.234  GEN CTLPKT    23  1  2  1  2  8     4 30000     1     2     0   100
     613170.133  REC CTLPKT    22  1  2  1  2  1    21     0     0     2     0
     613170.234  REC CTLPKT    23  1  2  1  2  8     4 30000     1     2     0
     613170.234  GEN CTLPKT    24  2  1  2  1  1    23     0     0     2     0   100
     613180.133  REC CTLPKT    24  2  1  2  1  1    23     0     0     2     0
     627092.930  CREATE CTL    25  2  1  5     4 30000     0     2     0   100
     627092.961  REC CTLPKT    25  2  1  2  2  5     4 30000     0     2     0
     627092.961  New Alloc      2     4  2  1   400  0     0  1     1


FRAME MAP FOR LINE   2 - 1
            -------
CHANNEL     | 1| 2|
            -------
CALL        | 2| 2|
            -------

     627092.961  GEN CTLPKT    26  2  1  2  1  5     4 30000     1     2     0   100
     627100.531  REC CTLPKT    26  2  1  2  1  5     4 30000     1     2     0
     627100.531  New Alloc      2     3  1  2   400  0     0  1     1


FRAME MAP FOR LINE   1 - 2
            -------
CHANNEL     | 1| 2|
            -------
CALL        | 2| 2|
            -------

     627100.531  GEN CTLPKT    27  1  2  1  2  5     3 30000     1     2     0   100
     627100.531  GEN CTLPKT    28  1  2  1  2  1    26     0     0     2     0   100
     627100.531  TALK           2  1  2
     627110.531  REC CTLPKT    27  1  2  1  2  5     3 30000     1     2     0
     627110.531  GEN CTLPKT    29  2  1  2  1  1    27     0     0     2     0   100
     627110.633  REC CTLPKT    28  1  2  1  2  1    26     0     0     2     0
     627120.531  REC CTLPKT    29  2  1  2  1  1    27     0     0     2     0
     710789.578  CREATE CTL    30  1  2  2     5 20000     2     3     0   100
     710789.609  REC CTLPKT    30  1  2  1  1  2     5 20000     2     3     0
     710789.609  New Reserv     3     5  1  2  1    -1
```

```
710789.609  GEN CTLPKT   31  1  2  1  2  2     5 20000     2     3     0   100
710790.531  REC CTLPKT   31  1  2  1  2  2     5 20000     2     3     0
710790.531  New Reserv    3     6  2  1  1    -1
710790.531  GEN CTLPKT   32  2  1  2  1  1    31     0     0     3     0   100
710790.531  GEN CTLPKT   33  2  1  2  1  2     6 20000    -1     3     5   100
710800.531  REC CTLPKT   32  2  1  2  1  1    31     0     0     3     0
710800.633  REC CTLPKT   33  2  1  2  1  2     6 20000    -1     3     5
710800.633  GEN CTLPKT   34  1  2  1  2  1    33     0     0     3     0   100
710890.633  GEN CTLPKT   35  1  2  1  2  8     6 20000     1     3     0   100
710810.531  REC CTLPKT   34  1  2  1  2  1    33     0     0     3     0
710810.633  REC CTLPKT   35  1  2  1  2  8     6 20000     1     3     0
710810.633  GEN CTLPKT   36  2  1  2  1  1    35     0     0     3     0   100
710820.531  REC CTLPKT   36  2  1  2  1  1    35     0     0     3     0
748411.594  CREATE CTL   37  2  1  5     6 20000     0     3     0   100
748411.625  REC CTLPKT   37  2  1  2  2  5     6 20000     6     3     0
748411.625  New Alloc     3     6  2  1   200  0     0  1     3
```

FRAME MAP FOR LINE 2 - 1

```
          ----------
CHANNEL   | 1| 2| 3|
          ----------
CALL      | 2| 2| 3|
          ----------
```

```
748411.625  GEN CTLPKT   38  2  1  2  1  5     6 20000     3     3     0   100
748420.734  REC CTLPKT   38  2  1  2  1  5     6 20000     3     3     0
748420.734  New Alloc     3     5  1  2   200  0     0  1     3
```

FRAME MAP FOR LINE 1 - 2

```
          ----------
CHANNEL   | 1| 2| 3|
          ----------
CALL      | 2| 2| 3|
          ----------
```

```
748420.734  GEN CTLPKT   39  1  2  1  2  5     5 20000     3     3     0   100
748420.734  GEN CTLPKT   40  1  2  1  2  1    38     0     0     3     0   100
748420.734  TALK          3  1  2
748430.734  REC CTLPKT   39  1  2  1  2  5     5 20000     3     3     0
748430.734  GEN CTLPKT   41  2  1  2  1  1    39     0     0     3     0   100
748430.836  REC CTLPKT   40  1  2  1  2  1    38     0     0     3     0
748440.734  REC CTLPKT   41  2  1  2  1  1    39     0     0     3     0
806262.313  CREATE CTL   42  1  2  2     7 30000     2     4     0   100
806262.344  REC CTLPKT   42  1  2  1  1  2     7 30000     2     4     0
806262.344  New Reserv    4     7  1  2  1    -1
806262.344  GEN CTLPKT   43  1  2  1  2  2     7 30000     2     4     0   100
806270.734  REC CTLPKT   43  1  2  1  2  2     7 30000     2     4     0
806270.734  New Reserv    4     8  2  1  1    -1
806270.734  GEN CTLPKT   44  2  1  2  1  1    43     0     0     4     0   100
806270.734  GEN CTLPKT   45  2  1  2  1  2     8 30000    -1     4     7   100
806280.734  REC CTLPKT   44  2  1  2  1  1    43     0     0     4     0
806280.836  REC CTLPKT   45  2  1  2  1  2     8 30000    -1     4     7
806280.836  GEN CTLPKT   46  1  2  1  2  1    45     0     0     4     0   100
806280.836  GEN CTLPKT   47  1  2  1  2  8     8 30000     1     4     8   100
806290.734  REC CTLPKT   46  1  2  1  2  1    45     0     0     4     0
```

```
      806290.836  REC CTLPKT    47  1  2  1  2  8     8 30000     1     4     0
      806290.836  GEN CTLPKT    48  2  1  2  1  1    47     0     0     4     0   100
      806300.734  REC CTLPKT    48  2  1  2  1  1    47     0     0     4     0
      808107.750  CREATE CTL    49  2  1  5     8 30000     0     4     0   100
      808107.781  REC CTLPKT    49  2  1  2  2  5     8 30000     0     4     0
      808107.781  New Alloc      4     8  2  1   400  0     0  1     4

FRAME MAP FOR LINE   2 -  1
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 2| 2| 3| 4| 4|
          ----------------

      808107.781  GEN CTLPKT    50  2  1  2  1  5     8 30000     4     4     0   100
      808111.133  REC CTLPKT    50  2  1  2  1  5     8 30000     4     4     0
      808111.133  New Alloc      4     7  1  2   400  0     0  1     4

FRAME MAP FOR LINE   1 -  2
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 2| 2| 3| 4| 4|
          ----------------

      808111.133  GEN CTLPKT    51  1  2  1  2  5     7 30000     4     4     0   100
      808111.133  GEN CTLPKT    52  1  2  1  2  1    50     0     0     4     0   100
      808111.133  TALK           4  1  2
      808121.133  REC CTLPKT    51  1  2  1  2  5     7 30000     4     4     0
      808121.133  GEN CTLPKT    53  2  1  2  1  1    51     0     0     4     0   100
      808121.234  REC CTLPKT    52  1  2  1  2  1    50     0     0     4     0
      808131.133  REC CTLPKT    53  2  1  2  1  1    51     0     0     4     0
      872723.727  CREATE CTL    54  1  2  2     9 30000     2     5     0   100
      872723.758  REC CTLPKT    54  1  2  1  1  2     9 30000     2     5     0
      872723.758  BLCK           5  1  2  1
      907486.328  CREATE CTL    55  1  2  6     5     0     0     3     0   100
      907486.352  REC CTLPKT    55  1  2  1  1  6     5     0     0     3     0
      907486.352  GEN CTLPKT    56  1  2  1  2  6     5     3     0     3     0   100
      907491.133  REC CTLPKT    56  1  2  1  2  6     5     3     0     3     0
      907491.133  GEN CTLPKT    57  2  1  2  1  6     6     3     0     3     0   100
      907491.133  GEN CTLPKT    58  2  1  2  1  1    56     0     0     3     0   100
      907501.133  REC CTLPKT    57  2  1  2  1  6     6     3     0     3     0
      907501.133  GEN CTLPKT    59  1  2  1  2  1    57     0     0     3     0   100
      907501.234  REC CTLPKT    58  2  1  2  1  1    56     0     0     3     0

FRAME MAP FOR LINE   1 -  2
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 2| 2| *| 4| 4|
          ----------------

      907511.133  REC CTLPKT    59  1  2  1  2  1    57     0     0     3     0
```

```
FRAME MAP FOR LINE   2 - 1
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 2| 2| *| 4| 4|
          ----------------

    1012868.664  CREATE CTL   60  1  2  6     3     0     0     2     0   100
    1012868.695  REC CTLPKT   60  1  2  1  1  6     3     0     0     2     0
    1012868.695  GEN CTLPKT   61  1  2  1  2  6     3     1     0     2     0   100
    1012871.133  REC CTLPKT   61  1  2  1  2  6     3     1     0     2     0
    1012871.133  GEN CTLPKT   62  2  1  2  1  6     4     1     0     2     0   100
    1012871.133  GEN CTLPKT   63  2  1  2  1  1    61     0     0     2     0   100
    1012881.133  REC CTLPKT   62  2  1  2  1  6     4     1     0     2     0
    1012881.133  GEN CTLPKT   64  1  2  1  2  1    62     0     0     2     0   100
    1012881.234  REC CTLPKT   63  2  1  2  1  1    61     0     0     2     0

FRAME MAP FOR LINE   1 - 2
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | *| *| *| 4| 4|
          ----------------

    1012881.234  New Alloc     4     7  1  2   400  0     0  1     1

FRAME MAP FOR LINE   1 - 2
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 4| 4| *| 4| 4|
          ----------------

    1012881.234  GEN CTLPKT   65  1  2  1  2  7     7     1     4     4     0   100
    1012891.133  REC CTLPKT   64  1  2  1  2  1    62     0     0     2     0

FRAME MAP FOR LINE   2 - 1
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | *| *| *| 4| 4|
          ----------------

    1012891.133  New Alloc     4     8  2  1   400  0     0  1     1

FRAME MAP FOR LINE   2 - 1
          ----------------
CHANNEL   | 1| 2| 3| 4| 5|
          ----------------
CALL      | 4| 4| *| 4| 4|
          ----------------
```

```
    1012891.133  GEN CTLPKT   65  2  1  2  1  7     8     1     4     4     0   100
    1012891.234  REC CTLPKT   65  1  2  1  2  7     7     1     4     4     0
    1012891.234  GEN CTLPKT   67  2  1  2  1  1    65     0     0     4     0   100
    1012901.133  REC CTLPKT   66  2  1  2  1  7     8     1     4     4     0
    1012901.133  GEN CTLPKT   68  1  2  1  2  1    66     0     0     4     0   100
    1012901.234  REC CTLPKT   67  2  1  2  1  1    65     0     0     4     0


FRAME MAP FOR LINE   1 - 2
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 4| 4|
          -------

    1012910.531  REC CTLPKT   68  1  2  1  2  1    66     0     0     4     0


FRAME MAP FOR LINE   2 - 1
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 4| 4|
          -------

    1071122.531  CREATE CTL   69  1  2  6     7     0     0     4     0   100
    1071122.563  REC CTLPKT   69  1  2  1  1  6     7     0     0     4     0
    1071122.563  GEN CTLPKT   70  1  2  1  2  6     7     1     0     4     0   100
    1071130.531  REC CTLPKT   70  1  2  1  2  6     7     1     0     4     0
    1071130.531  GEN CTLPKT   71  2  1  2  1  6     8     1     0     4     0   100
    1071130.531  GEN CTLPKT   72  2  1  2  1  1    70     0     0     4     0   100
    1071140.531  REC CTLPKT   71  2  1  2  1  6     8     1     0     4     0
    1071140.531  GEN CTLPKT   73  1  2  1  2  1    71     0     0     4     0   100
    1071140.625  REC CTLPKT   72  2  1  2  1  1    70     0     0     4     0


FRAME MAP FOR LINE   1 - 2
NO ONGOING CALLS AT PRESENT
    1071150.125  REC CTLPKT   73  1  2  1  2  1    71     0     0     4     0


FRAME MAP FOR LINE   2 - 1
NO ONGOING CALLS AT PRESENT
    1490803.281  CREATE CTL   74  1  2  2    10 20000     2     6     0   100
    1490803.313  REC CTLPKT   74  1  2  1  1  2    10 20000     2     6     0
    1490803.313  New Reserv    6    10  1  2  1    -1
    1490803.313  GEN CTLPKT   75  1  2  1  2  2    10 20000     2     6     0   100
    1490810.125  REC CTLPKT   75  1  2  1  2  2    10 20000     2     6     0
    1490810.125  New Reserv    6    11  2  1  1    -1
    1490810.125  GEN CTLPKT   76  2  1  2  1  1    75     0     0     6     0   100
    1490810.125  GEN CTLPKT   77  2  1  2  1  2    11 20000    -1     6    10   100
    1490820.125  REC CTLPKT   76  2  1  2  1  1    75     0     0     6     0
    1490820.219  REC CTLPKT   77  2  1  2  1  2    11 20000    -1     6    10
    1490820.219  GEN CTLPKT   78  1  2  1  2  1    77     0     0     6     0   100
    1490820.219  GEN CTLPKT   79  1  2  1  2  8    11 20000     1     6     0   100
    1490830.125  REC CTLPKT   78  1  2  1  2  1    77     0     0     6     0
    1490830.219  REC CTLPKT   79  1  2  1  2  8    11 20000     1     6     0
```

```
    1490830.219  GEN CTLPKT   80  2  1  2  1  1    79     0     0     6     0   100
    1490840.125  REC CTLPKT   80  2  1  2  1  1    79     0     0     6     0
    1518336.359  CREATE CTL   81  2  1  5    11 20000     0     6     0   100
    1518336.391  REC CTLPKT   81  2  1  2  2  5    11 20000     0     6     0
    1518336.391  New Alloc     6    11  2  1   200  0     0  1     1

FRAME MAP FOR LINE   2 - 1
          ----
CHANNEL   | 11|
          ----
CALL      | 61|
          ----

    1518336.391  GEN CTLPKT   82  2  1  2  1  5    11 20000     1     6     0   100
    1518340.328  REC CTLPKT   82  2  1  2  1  5    11 20000     1     6     0
    1518340.328  New Alloc     6    10  1  2   200  0     0  1     1

FRAME MAP FOR LINE   1 - 2
          ----
CHANNEL   | 11|
          ----
CALL      | 61|
          ----

    1518340.328  GEN CTLPKT   83  1  2  1  2  5    10 20000     1     6     0   100
    1518340.328  GEN CTLPKT   84  1  2  1  2  1    82     0     0     6     0   100
    1518340.328  TALK          6  1  2
    1518350.328  REC CTLPKT   83  1  2  1  2  5    10 20000     1     6     0
    1518350.328  GEN CTLPKT   85  2  1  2  1  1    83     0     0     6     0   100
    1518350.422  REC CTLPKT   84  1  2  1  2  1    82     0     0     6     0
    1518360.328  REC CTLPKT   85  2  1  2  1  1    83     0     0     6     0
    1522650.016  CREATE CTL   86  1  2  6    10     0     0     6     0   100
    1522650.047  REC CTLPKT   86  1  2  1  1  6    10     0     0     6     0
    1522650.047  GEN CTLPKT   87  1  2  1  2  6    10     1     0     6     0   100
    1522660.328  REC CTLPKT   87  1  2  1  2  6    10     1     0     6     0
    1522660.328  GEN CTLPKT   88  2  1  2  1  6    11     1     6     6     0   100
    1522660.328  GEN CTLPKT   89  2  1  2  1  1    87     0     0     6     0   100
    1522670.328  REC CTLPKT   88  2  1  2  1  6    11     1     0     6     0
    1522670.328  GEN CTLPKT   90  1  2  1  2  1    88     0     0     6     0   100
    1522670.422  REC CTLPKT   89  2  1  2  1  1    87     0     0     6     0

FRAME MAP FOR LINE   1 - 2
NO ONGOING CALLS AT PRESENT
    1522680.125  REC CTLPKT   90  1  2  1  2  1    88     0     0     6     0

FRAME MAP FOR LINE   2 - 1
NO ONGOING CALLS AT PRESENT
```

A.2. Example 2

Figure A.2 shows the network for this example, together with the routing table for each of the nodes. There is only one voice rate (20 KBPS) that requires 200-bit slots. Again the voice fraction in the frame is 0.1 . This time, though, lines of different speeds are used to simulate the effect of unequal loading on lines that have the same speed. The maximum number of channels on each of the lines is as follows:

Lines 1-2 and 2-1:	5 channels
Lines 1-3 and 3-1:	1 channel
Lines 2-3 and 3-2:	2 channels
Lines 2-4 and 4-2:	0 channels
Lines 3-4 and 4-3:	5 channels

All calls are from node 1 to node 4.

TIME	EVENT
85168.784	Call 1 is initiated.
85168.815	Node 1 makes a reservation for call 1 on its primary route to node 4 (line 1-2)
851170.131	Node 2 makes a reservation for call 1 on its secondary route to node 4 (line 2-3) because the capacity on the primary route (line 2-4) was not sufficient to carry the call.
85180.281	Node 3 makes a reservation for call 1 on its primary route to node 4 (line 3-4). This completes the forward reservation for call 1.
85190.131	Node 4 makes a reservation for call 1 on its primary route to node 1 (line 4-3).
85200.230	Node 3 makes a reservation for call 1 on its primary route to node 1 (line 3-1). This completes the reservation phase for call 1. Note that because of "congestion" on line 2-4, the forward and backward paths for call 1 did not coincide (forward: 1-2-3-4 ; backward: 4-3-1).
127878.757	Reservation for call 2 begins at node 1. Its forward path is the same as that for call 1.

127900.431 Node 4 makes a reservation for call 2 on its primary route to node 1 (line 4-3).

127910.431 Node 3 makes a reservation for call 2 on its secondary route to node 1 (line 3-2) since its primary route (line 3-1) is congested (call 1 is still going on). In this case the forward and backward paths coincided.

```
85168.784  CREATE CTL     1  1  4  2     1 20000     2     1     0   100
85168.815  REC CTLPKT     1  1  4  1  1  2     1 20000     2     1     0
85168.815  New Reserv     1     1  1  2  1    -1
85168.815  GEN CTLPKT     2  1  4  1  2  2     1 20000     2     1     0   100
85170.131  REC CTLPKT     2  1  4  1  2  2     1 20000     2     1     0
85170.131  New Reserv     1     1  2  3  2    -1
85170.131  GEN CTLPKT     3  2  1  2  1  1     2     0     0     1     0   100
85170.131  GEN CTLPKT     4  1  4  2  3  2     1 20000     2     1     0   100
85180.131  REC CTLPKT     3  2  1  2  1  1     2     0     0     1     0
85180.281  REC CTLPKT     4  1  4  2  3  2     1 20000     2     1     0
85180.281  New Reserv     1     1  3  4  3    -1
85180.281  GEN CTLPKT     5  3  2  3  2  1     4     0     0     1     0   100
85180.281  GEN CTLPKT     6  1  4  3  4  2     1 20000     2     1     0   100
85190.131  REC CTLPKT     6  1  4  3  4  2     1 20000     2     1     0
85190.131  New Reserv     1     2  4  3  2    -1
85190.131  GEN CTLPKT     7  4  3  4  3  1     6     0     0     1     0   100
85190.131  GEN CTLPKT     8  4  1  4  3  2     2 20000    -1     1     1   100
85190.281  REC CTLPKT     5  3  2  3  2  1     4     0     0     1     0
85200.131  REC CTLPKT     7  4  3  4  3  1     6     0     0     1     0
85200.230  REC CTLPKT     8  4  1  4  3  2     2 20000    -1     1     1
85200.230  New Reserv     1     2  3  1  1    -1
85200.230  GEN CTLPKT     9  3  4  3  4  1     8     0     0     1     0   100
85200.230  GEN CTLPKT    10  4  1  3  1  2     2 20000    -1     1     1   100
85210.131  REC CTLPKT     9  3  4  3  4  1     8     0     0     1     0
85210.531  REC CTLPKT    10  4  1  3  1  2     2 20000    -1     1     1
85210.531  GEN CTLPKT    11  1  3  1  3  1    10     0     0     1     0   100
85210.531  GEN CTLPKT    12  1  4  1  2  8     2 20000     1     1     0   100
85220.131  REC CTLPKT    12  1  4  1  2  8     2 20000     1     1     0
85220.131  GEN CTLPKT    13  2  1  2  1  1    12     0     0     1     0   100
85220.131  GEN CTLPKT    14  1  4  2  4  8     2 20000     1     1     0   100
85220.531  REC CTLPKT    11  1  3  1  3  1    10     0     0     1     0
85230.131  REC CTLPKT    13  2  1  2  1  1    12     0     0     1     0
85231.031  REC CTLPKT    14  1  4  2  4  8     2 20000     1     1     0
85231.031  GEN CTLPKT    15  4  2  4  2  1    14     0     0     1     0   100
85241.031  REC CTLPKT    15  4  2  4  2  1    14     0     0     1     0
89815.089  CREATE CTL    16  4  1  5     2 20000     0     1     0   100
89815.120  REC CTLPKT    16  4  1  4  4  5     2 20000     0     1     0
89815.120  New Alloc      1     2  4  3   200  0     0  2     1
```

FRAME MAP FOR LINE 4 - 3

```
          ----
CHANNEL   | 1|
          ----
CALL      | 1|
          ----
```

```
89815.120  GEN CTLPKT    17  4  1  4  3  5     2 20000     1     1     0   100
89820.331  REC CTLPKT    17  4  1  4  3  5     2 20000     1     1     0
89820.331  New Alloc      1     2  3  1   200  3     1  1     1
```

FRAME MAP FOR LINE 3 - 1

```
          ----
CHANNEL   | 1|
          ----
CALL      | 1|
          ----
```

```
      89820.331  GEN CTLPKT   16  4  1  3  1  5     2 20000     1     1     0   100
      89820.331  GEN CTLPKT   19  3  4  3  4  1    17     0     0     1     0   100
      89830.131  REC CTLPKT   19  3  4  3  4  1    17     0     0     1     0
      89831.531  REC CTLPKT   18  4  1  3  1  5     2 20000     1     1     0
      89831.531  New Alloc     1     1  1  2   200  0     0  1     1


FRAME MAP FOR LINE   1 - 2
          ----
CHANNEL   | 11
          ----
CALL      | 11
          ----

      89831.531  GEN CTLPKT   20  1  4  1  2  5     1 20000     1     1     0   100
      89831.531  GEN CTLPKT   21  1  3  1  3  1    18     0     0     1     0   100
      89831.531  TALK          1  1  4
      89840.331  REC CTLPKT   20  1  4  1  2  5     1 20000     1     1     0
      89840.331  New Alloc     1     1  2  3   200  1     1  2     1


FRAME MAP FOR LINE   2 - 3
          ----
CHANNEL   | 11
          ----
CALL      | 11
          ----

      89840.331  GEN CTLPKT   22  1  4  2  3  5     1 20000     1     1     0   100
      89840.331  GEN CTLPKT   23  2  1  2  1  1    20     0     0     1     0   100
      89840.531  REC CTLPKT   21  1  3  1  3  1    18     0     0     1     0
      89850.131  REC CTLPKT   23  2  1  2  1  1    20     0     0     1     0
      89850.781  REC CTLPKT   22  1  4  2  3  5     1 20000     1     1     0
      89850.781  New Alloc     1     1  3  4   200  2     1  3     1


FRAME MAP FOR LINE   3 - 4
          ----
CHANNEL   | 11
          ----
CALL      | 11
          ----

      89850.781  GEN CTLPKT   24  1  4  3  4  5     1 20000     1     1     0   100
      89850.781  GEN CTLPKT   25  3  2  3  2  1    22     0     0     1     0   100
      89860.281  REC CTLPKT   25  3  2  3  2  1    22     0     0     1     0
      89860.331  REC CTLPKT   24  1  4  3  4  5     1 20000     1     1     0
      89860.331  GEN CTLPKT   26  4  3  4  3  1    24     0     0     1     0   100
      89870.331  REC CTLPKT   26  4  3  4  3  1    24     0     0     1     0
     127878.726  CREATE CTL   27  1  4  2     3 20000     2     2     0   100
     127878.757  REC CTLPKT   27  1  4  1  1  2     3 20000     2     2     0
     127878.757  New Reserv    2     3  1  2  1    -1
     127878.757  GEN CTLPKT   28  1  4  1  2  2     3 20000     2     2     0   100
     127880.331  REC CTLPKT   28  1  4  1  2  2     3 20000     2     2     0
     127880.331  New Reserv    2     3  2  3  2    -1
     127880.331  GEN CTLPKT   29  2  1  2  1  1    28     0     0     2     0   100
     127880.331  GEN CTLPKT   30  1  4  2  3  2     3 20000     2     2     0   100
```

```
127890.131  REC CTLPKT   29  2  1  2  1  1    28     0     0     2     0
127890.781  REC CTLPKT   30  1  4  2  3  2     3 20000     2     2     0
127890.781  New Reserv    2     3  3  4  3    -1
127890.781  GEN CTLPKT   31  3  2  3  2  1    30     0     0     2     0   100
127890.781  GEN CTLPKT   32  1  4  3  4  2     3 20000     2     2     0   100
127900.281  REC CTLPKT   31  3  2  3  2  1    30     0     0     2     0
127900.331  REC CTLPKT   32  1  4  3  4  2     3 20000     2     2     0
127900.331  New Reserv    2     4  4  3  2    -1
127900.331  GEN CTLPKT   33  4  3  4  3  1    32     0     0     2     0   100
127900.331  GEN CTLPKT   34  4  1  4  3  2     4 20000    -1     2     3   100
127910.331  REC CTLPKT   33  4  3  4  3  1    32     0     0     2     0
127910.431  REC CTLPKT   34  4  1  4  3  2     4 20000    -1     2     3
127910.431  New Reserv    2     4  3  2  2    -1
127910.431  GEN CTLPKT   35  3  4  3  4  1    34     0     0     2     0   100
127910.431  GEN CTLPKT   36  4  1  3  2  2     4 20000    -1     2     3   100
127920.281  REC CTLPKT   36  4  1  3  2  2     4 20000    -1     2     3
127920.281  New Reserv    2     4  2  1  1    -1
127920.281  GEN CTLPKT   37  2  3  2  3  1    36     0     0     2     0   100
127920.281  GEN CTLPKT   38  4  1  2  1  2     4 20000    -1     2     3   100
127920.331  REC CTLPKT   35  3  4  3  4  1    34     0     0     2     0
127930.131  REC CTLPKT   38  4  1  2  1  2     4 20000    -1     2     3
127930.131  GEN CTLPKT   39  1  2  1  2  1    38     0     0     2     0   100
127930.131  GEN CTLPKT   40  1  4  1  2  8     4 20000     1     2     0   100
127930.781  REC CTLPKT   37  2  3  2  3  1    36     0     0     2     0
127940.331  REC CTLPKT   39  1  2  1  2  1    38     0     0     2     0
127940.431  REC CTLPKT   40  1  4  1  2  8     4 20000     1     2     0
127940.431  GEN CTLPKT   41  2  1  2  1  1    40     0     0     2     0   100
127940.431  GEN CTLPKT   42  1  4  2  4  8     4 20000     1     2     0   100
127950.131  REC CTLPKT   41  2  1  2  1  1    40     0     0     2     0
127951.031  REC CTLPKT   42  1  4  2  4  8     4 20000     1     2     0
127951.031  GEN CTLPKT   43  4  2  4  2  1    42     0     0     2     0   100
127961.031  REC CTLPKT   43  4  2  4  2  1    42     0     0     2     0
128540.208  CREATE CTL   44  4  1  5     4 20000     0     2     0   100
128540.239  REC CTLPKT   44  4  1  4  4  5     4 20000     0     2     0
128540.239  New Alloc     2     4  4  3   200  0     0  2     2


FRAME MAP FOR LINE   4 - 3
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 2|
          -------

128540.239  GEN CTLPKT   45  4  1  4  3  5     4 20000     2     2     0   100
128550.531  REC CTLPKT   45  4  1  4  3  5     4 20000     2     2     0
128550.531  New Alloc     2     4  3  2   200  3     2  2     1


FRAME MAP FOR LINE   3 - 2
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

128550.531  GEN CTLPKT   46  4  1  3  2  5     4 20000     1     2     0   100
```

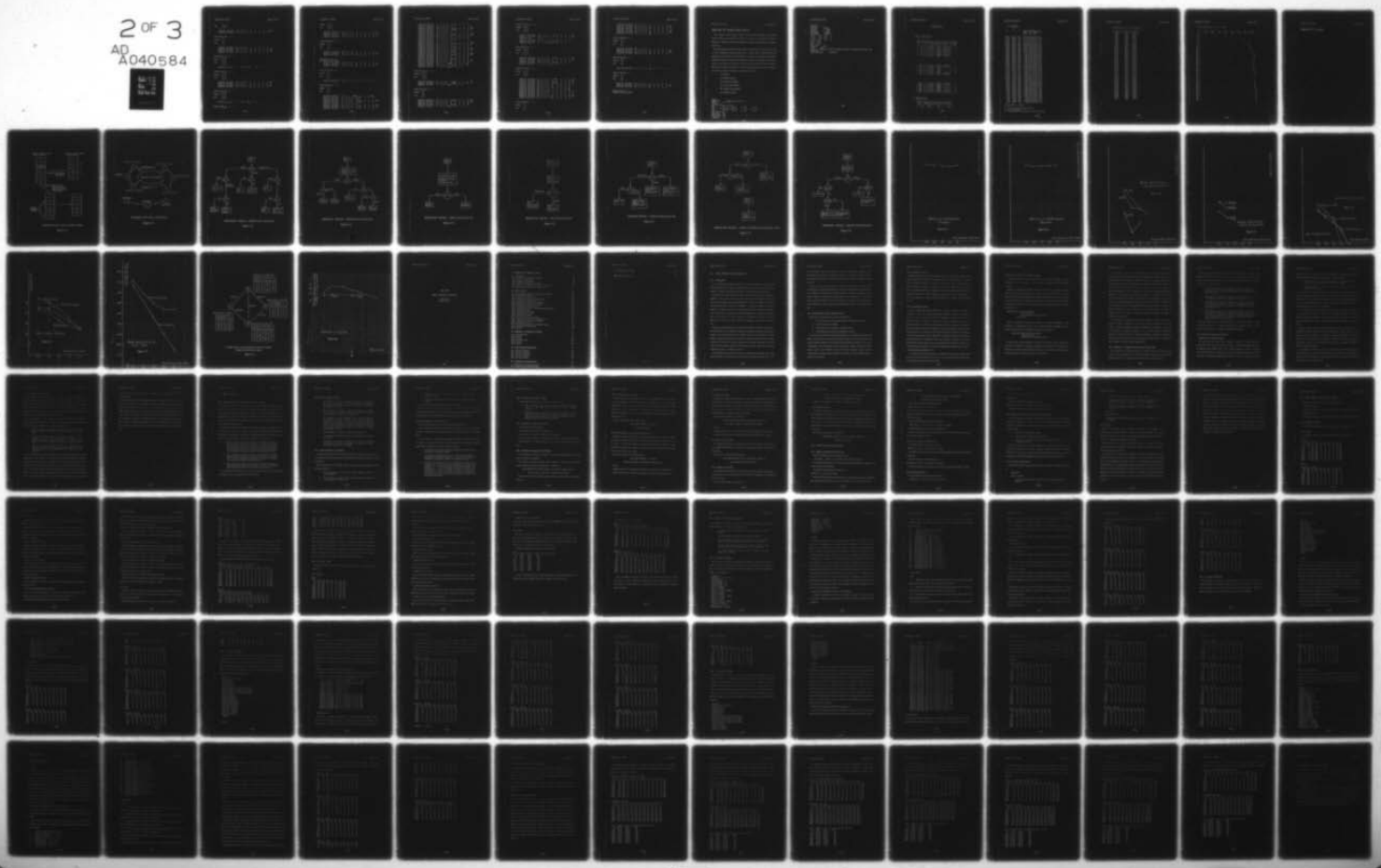
```
128550.531  GEN CTLPKT   47  3  4  3  4  1    45     0     0     2     0  100
128560.331  REC CTLPKT   47  3  4  3  4  1    45     0     0     2     0
128560.781  REC CTLPKT   46  4  1  3  2  5     4 20000     1     2     0
128560.781  New Alloc     2     4  2  1   200  2     1  1     1

FRAME MAP FOR LINE   2 - 1
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

128560.781  GEN CTLPKT   48  4  1  2  1  5     4 20000     1     2     0  100
128560.781  GEN CTLPKT   49  2  3  2  3  1    46     0     0     2     0  100
128570.331  REC CTLPKT   48  4  1  2  1  5     4 20000     1     2     0
128570.331  New Alloc     2     3  1  2   200  0     0  1     2

FRAME MAP FOR LINE   1 - 2
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 2|
          -------

128570.331  GEN CTLPKT   50  1  4  1  2  5     3 20000     2     2     0  100
128570.331  GEN CTLPKT   51  2  2  1  2  1    48     0     0     2     0  100
128570.331  TALK          2  1  4
128570.781  REC CTLPKT   49  2  3  2  3  1    46     0     0     2     0
128580.531  REC CTLPKT   50  1  4  1  2  5     3 20000     2     2     0
128580.531  New Alloc     2     3  2  3   200  1     2  2     2

FRAME MAP FOR LINE   2 - 3
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 2|
          -------

128580.531  GEN CTLPKT   52  1  4  2  3  5     3 20000     2     2     0  100
128580.531  GEN CTLPKT   53  2  1  2  1  1    50     0     0     2     0  100
128580.631  REC CTLPKT   51  1  2  1  2  1    48     0     0     2     0
128590.331  REC CTLPKT   53  2  1  2  1  1    50     0     0     2     0
128591.281  REC CTLPKT   52  1  4  2  3  5     3 20000     2     2     0
128591.281  New Alloc     2     3  3  4   200  2     2  3     2

FRAME MAP FOR LINE   3 - 4
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 1| 2|
          -------
```

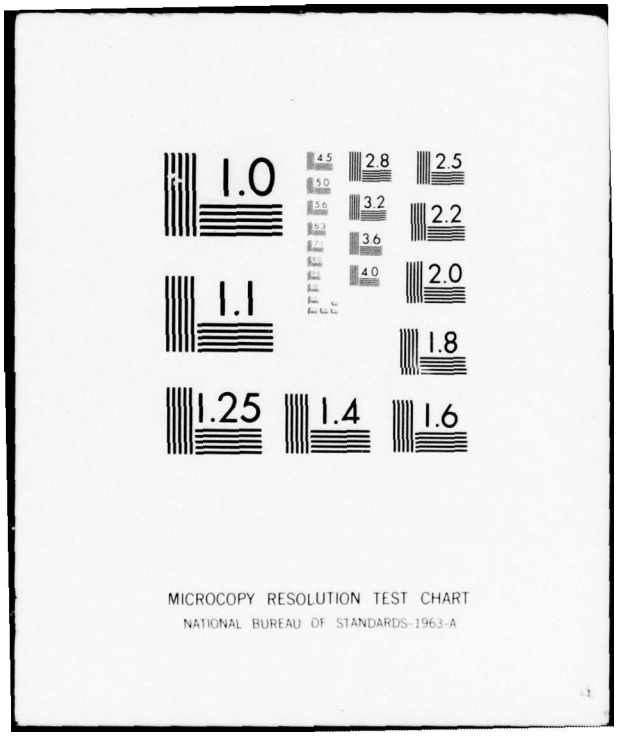
```
     128591.281  GEN CTLPKT    54  1  4  3  4  5     3 20000     2     2     0   100
     128591.281  GEN CTLPKT    55  3  2  3  2  1    52     0     0     2     0   100
     128600.531  REC CTLPKT    54  1  4  3  4  5     3 20000     2     2     0
     128600.531  GEN CTLPKT    56  4  3  4  3  1    54     0     0     2     0   100
     128600.781  REC CTLPKT    55  3  2  3  2  1    52     0     0     2     0
     128610.531  REC CTLPKT    56  4  3  4  3  1    54     0     0     2     0
     131542.137  CREATE CTL    57  1  4  6     1     0     0     1     0   100
     131542.168  REC CTLPKT    57  1  4  1  1  6     1     0     0     1     0
     131542.168  GEN CTLPKT    58  1  4  1  2  6     1     1     0     1     0   100
     131550.531  REC CTLPKT    58  1  4  1  2  6     1     1     0     1     0
     131550.531  GEN CTLPKT    59  1  4  2  3  6     1     1     0     1     0   100
     131550.531  GEN CTLPKT    60  2  1  2  1  1    58     0     0     1     0   100
     131560.330  REC CTLPKT    60  2  1  2  1  1    58     0     0     1     0

FRAME MAP FOR LINE   1 - 2
          --------
CHANNEL   | 1| 2|
          --------
CALL      | *| 2|
          --------

     131560.330  New Alloc      2     3  1  2   200  0     0  1     1

FRAME MAP FOR LINE   1 - 2
          --------
CHANNEL   | 1| 2|
          --------
CALL      | 2| 2|
          --------

     131560.330  GEN CTLPKT    61  1  2  1  2  7     3     1     2     2     0   100
     131561.281  REC CTLPKT    59  1  4  2  3  6     1     1     0     1     0
     131561.281  GEN CTLPKT    62  1  4  3  4  6     1     1     0     1     0   100
     131561.281  GEN CTLPKT    63  3  2  3  2  1    59     0     0     1     0   100
     131570.531  REC CTLPKT    61  1  2  1  2  7     3     1     2     2     0
     131570.531  GEN CTLPKT    64  2  1  2  1  1    61     0     0     2     0   100
     131570.531  REC CTLPKT    62  1  4  3  4  6     1     1     0     1     0
     131570.531  GEN CTLPKT    65  4  1  4  3  6     2     1     0     1     0   100
     131570.531  GEN CTLPKT    66  4  3  4  3  1    62     0     0     1     0   100
     131570.781  REC CTLPKT    63  3  2  3  2  1    59     0     0     1     0

FRAME MAP FOR LINE   2 - 3
          --------
CHANNEL   | 1| 2|
          --------
CALL      | *| 2|
          --------

     131570.781  New Alloc      2     3  2  3   200  1     1  2     1

FRAME MAP FOR LINE   2 - 3
          --------
CHANNEL   | 1| 2|
```

AD-A040 584 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 17/2
THE SIMULATION OF AN INTEGRATED VOICE/DATA COMMUNICATIONS NETWO--ETC(U)
MAY 77 M R BARBACCI DCA100-76-C-0058

UNCLASSIFIED DCA-100-76-C-0058 NL

2 OF 3
AD
A040584




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
          -------
CALL      | 2| 2|
          -------

     131570.781  GEN CTLPKT   67  2  3  2  3  7    3    1    2    2    0   100
     131580.330  REC CTLPKT   64  2  1  2  1  1   61    0    0    2    0


FRAME MAP FOR LINE   1 - 2
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     131580.531  REC CTLPKT   65  4  1  4  3  6    2    1    0    1    0
     131580.531  GEN CTLPKT   68  4  1  3  1  6    2    1    0    1    0   100
     131580.531  GEN CTLPKT   69  3  4  3  4  1   65    0    0    1    0   100
     131580.631  REC CTLPKT   66  4  3  4  3  1   62    0    0    1    0


FRAME MAP FOR LINE   3 - 4
          -------
CHANNEL   | 1| 2|
          -------
CALL      | *| 2|
          -------

     131580.631  New Alloc     2    3  3  4   200  2    2  3    1


FRAME MAP FOR LINE   3 - 4
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 2| 2|
          -------

     131580.631  GEN CTLPKT   70  3  4  3  4  7    3    1    2    2    0   100
     131581.281  REC CTLPKT   67  2  3  2  3  7    3    1    2    2    0
     131581.281  GEN CTLPKT   71  3  2  3  2  1   67    0    0    2    0   100
     131590.531  REC CTLPKT   69  3  4  3  4  1   65    0    0    1    0


FRAME MAP FOR LINE   4 - 3
          -------
CHANNEL   | 1| 2|
          -------
CALL      | *| 2|
          -------

     131590.531  New Alloc     2    4  4  3   200  0    0  2    1


FRAME MAP FOR LINE   4 - 3
          -------
```

```
CHANNEL   | 1| 2|
          -------
CALL      | 2| 2|
          -------

     131590.531  GEN CTLPKT   72  4  3  4  3  7    4    1    2    2    0  100
     131590.631  REC CTLPKT   70  3  4  3  4  7    3    1    2    2    0
     131590.631  GEN CTLPKT   73  4  3  4  3  1   70    0    0    2    0  100
     131590.781  REC CTLPKT   71  3  2  3  2  1   67    0    0    2    0


FRAME MAP FOR LINE   2 - 3
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     131591.531  REC CTLPKT   68  4  1  3  1  6    2    1    0    1    0
     131591.531  GEN CTLPKT   74  1  3  1  3  1   68    0    0    1    0  100
     131600.531  REC CTLPKT   74  1  3  1  3  1   68    0    0    1    0


FRAME MAP FOR LINE   3 - 1
NO ONGOING CALLS AT PRESENT
     131600.563  REC CTLPKT   72  4  3  4  3  7    4    1    2    2    0
     131600.563  GEN CTLPKT   75  3  4  3  4  1   72    0    0    2    0  100
     131600.631  REC CTLPKT   73  4  3  4  3  1   70    0    0    2    0


FRAME MAP FOR LINE   3 - 4
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     131610.330  REC CTLPKT   75  3  4  3  4  1   72    0    0    2    0


FRAME MAP FOR LINE   4 - 3
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     513646,863  CREATE CTL   76  1  4  2     5 20000    2    3    0  100
     513646.895  REC CTLPKT   76  1  4  1  1  2    5 20000    2    3    0
     513646.895  New Reserv    3     5  1  2  1   -1
     513646.895  GEN CTLPKT   77  1  4  1  2  2    5 20000    2    3    0  100
     513650.332  REC CTLPKT   77  1  4  1  2  2    5 20000    2    3    0
     513650.332  New Reserv    3     5  2  3  2   -1
     513650.332  GEN CTLPKT   78  2  1  2  1  1   77    0    0    3    0  100
     513650.332  GEN CTLPKT   79  1  4  2  3  2    5 20000    2    3    0  100
     513660.332  REC CTLPKT   78  2  1  2  1  1   77    0    0    3    0
```

```
513660.781  REC CTLPKT   79  1  4  2  3  2     5 20000     2     3     0
513660.781  New Reserv    3     5  3  4  3    -1
513660.781  GEN CTLPKT   80  3  2  3  2  1    79     0     0     3     0   100
513660.781  GEN CTLPKT   81  1  4  3  4  2     5 20000     2     3     0   100
513670.332  REC CTLPKT   81  1  4  3  4  2     5 20000     2     3     0
513670.332  New Reserv    3     6  4  3  2    -1
513670.332  GEN CTLPKT   82  4  3  4  3  1    81     0     0     3     0   100
513670.332  GEN CTLPKT   83  4  1  4  3  2     6 20000    -1     3     5   100
513670.781  REC CTLPKT   80  3  2  3  2  1    79     0     0     3     0
513680.332  REC CTLPKT   82  4  3  4  3  1    81     0     0     3     0
513680.434  REC CTLPKT   83  4  1  4  3  2     6 20000    -1     3     5
513680.434  New Reserv    3     6  3  1  1    -1
513680.434  GEN CTLPKT   84  3  4  3  4  1    83     0     0     3     0   100
513680.434  GEN CTLPKT   85  4  1  3  1  2     6 20000    -1     3     5   100
513690.332  REC CTLPKT   84  3  4  3  4  1    83     0     0     3     0
513690.531  REC CTLPKT   85  4  1  3  1  2     6 20000    -1     3     5
513690.531  GEN CTLPKT   86  1  3  1  3  1    85     0     0     3     0   100
513690.531  GEN CTLPKT   87  1  4  1  2  8     6 20000     1     3     0   100
513700.332  REC CTLPKT   87  1  4  1  2  8     6 20000     1     3     0
513700.332  GEN CTLPKT   88  2  1  2  1  1    87     0     0     3     0   100
513700.332  GEN CTLPKT   89  1  4  2  4  8     6 20000     1     3     0   100
513700.531  REC CTLPKT   86  1  3  1  3  1    85     0     0     3     0
513710.332  REC CTLPKT   88  2  1  2  1  1    87     0     0     3     0
513711.031  REC CTLPKT   89  1  4  2  4  8     6 20000     1     3     0
513711.031  GEN CTLPKT   90  4  2  4  2  1    89     0     0     3     0   100
513721.031  REC CTLPKT   90  4  2  4  2  1    89     0     0     3     0
534525.258  CREATE CTL   91  4  1  5     6 20000     0     3     0   100
534525.289  REC CTLPKT   91  4  1  4  4  5     6 20000     0     3     0
534525.289  New Alloc     3     6  4  3  200  0     0  2     2
```

```
FRAME MAP FOR LINE   4 - 3
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 2| 3|
          -------
```

```
534525.289  GEN CTLPKT   92  4  1  4  3  5     6 20000     2     3     0   100
534530.531  REC CTLPKT   92  4  1  4  3  5     6 20000     2     3     0
534530.531  New Alloc     3     6  3  1  200  3     2  1     1
```

```
FRAME MAP FOR LINE   3 - 1
          ----
CHANNEL   | 1|
          ----
CALL      | 3|
          ----
```

```
534530.531  GEN CTLPKT   93  4  1  3  1  5     6 20000     1     3     0   100
534530.531  GEN CTLPKT   94  3  4  3  4  1    92     0     0     3     0   100
534540.336  REC CTLPKT   94  3  4  3  4  1    92     0     0     3     0
534541.531  REC CTLPKT   93  4  1  3  1  5     6 20000     1     3     0
534541.531  New Alloc     3     5  1  2  200  0     0  1     2
```

```
FRAME MAP FOR LINE   1 -  2
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 2| 3|
          -------

     534541.531  GEN CTLPKT    95  1  4  1  2  5     5 20000     2     3     0   100
     534541.531  GEN CTLPKT    96  1  3  1  3  1    93     0     0     3     0   100
     534541.531  TALK           3  1  4
     534550.531  REC CTLPKT    96  1  3  1  3  1    93     0     0     3     0
     534550.531  REC CTLPKT    95  1  4  1  2  5     5 20000     2     3     0
     534550.531  New Alloc      3     5  2  3   200  1     2  2     2

FRAME MAP FOR LINE   2 -  3
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 2| 3|
          -------

     534550.531  GEN CTLPKT    97  1  4  2  3  5     5 20000     2     3     0   100
     534550.531  GEN CTLPKT    98  2  1  2  1  1    95     0     0     3     0   100
     534560.336  REC CTLPKT    98  2  1  2  1  1    95     0     0     3     0
     534561.281  REC CTLPKT    97  1  4  2  3  5     5 20000     2     3     0
     534561.281  New Alloc      3     5  3  4   200  2     2  3     2

FRAME MAP FOR LINE   3 -  4
          -------
CHANNEL   | 1| 2|
          -------
CALL      | 2| 3|
          -------

     534561.281  GEN CTLPKT    99  1  4  3  4  5     5 20000     2     3     0   100
     534561.281  GEN CTLPKT   100  3  2  3  2  1    97     0     0     3     0   100
     534570.531  REC CTLPKT    99  1  4  3  4  5     5 20000     2     3     0
     534570.531  GEN CTLPKT   101  4  3  4  3  1    99     0     0     3     0   100
     534576.781  REC CTLPKT   100  3  2  3  2  1    97     0     0     3     0
     534580.531  REC CTLPKT   101  4  3  4  3  1    99     0     0     3     0
     664195.852  CREATE CTL   102  1  4  6     5     0     0     3     0   100
     664195.883  REC CTLPKT   102  1  4  1  1  6     5     0     0     3     0
     664195.883  GEN CTLPKT   103  1  4  1  2  6     5     2     0     3     0   100
     664200.531  REC CTLPKT   103  1  4  1  2  6     5     2     0     3     0
     664200.531  GEN CTLPKT   104  1  4  2  3  6     5     2     0     3     0   100
     664200.531  GEN CTLPKT   105  2  1  2  1  1   103     0     0     3     0   100
     664210.336  REC CTLPKT   105  2  1  2  1  1   103     0     0     3     0

FRAME MAP FOR LINE   1 -  2
          ----
CHANNEL   | 1| 1|
          ----
CALL      | 2| 1|
          ----
```

```
     664211.281  REC CTLPKT   104  1  4  2  3  6    5    2    0    3    0
     664211.281  GEN CTLPKT   106  1  4  3  4  6    5    2    0    3    0  100
     664211.281  GEN CTLPKT   107  3  2  3  2  1  104    0    0    3    0  100
     664220.531  REC CTLPKT   106  1  4  3  4  6    5    2    0    3    0
     664220.531  GEN CTLPKT   108  4  1  4  3  6    6    2    0    3    0  100
     664220.531  GEN CTLPKT   109  4  3  4  3  1  106    0    0    3    0  100
     664220.781  REC CTLPKT   107  3  2  3  2  1  104    0    0    3    0

FRAME MAP FOR LINE   2 -  3
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     664230.531  REC CTLPKT   108  4  1  4  3  6    6    2    0    3    0
     664230.531  GEN CTLPKT   110  4  1  3  1  6    6    1    0    3    0  100
     664230.531  GEN CTLPKT   111  3  4  3  4  1  108    0    0    3    0  100
     664230.633  REC CTLPKT   109  4  3  4  3  1  106    0    0    3    0

FRAME MAP FOR LINE   3 -  4
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     664240.336  REC CTLPKT   111  3  4  3  4  1  108    0    0    3    0

FRAME MAP FOR LINE   4 -  3
          ----
CHANNEL   | 1|
          ----
CALL      | 2|
          ----

     664241.531  REC CTLPKT   110  4  1  3  1  6    6    1    0    3    0
     664241.531  GEN CTLPKT   112  1  3  1  3  1  110    0    0    3    0  100
     664250.531  REC CTLPKT   112  1  3  1  3  1  110    0    0    3    0

FRAME MAP FOR LINE   3 -  1
NO ONGOING CALLS AT PRESENT
```

Appendix B: Sample Output Report

This appendix shows a typical output of the Data Analysis Program. The various sections of this output have been described previously, and it is not difficult to follow through. An attempt will be made here, though, to explain how the initial bias period is estimated.

By examining the curve for the mean number of busy channels versus time , it can be easily established that the steady state is on after about 11 minutes from the beginning of the simulation. To consolidate this estimate, the standard deviation entries tabulated alongside the means can be plotted on log-log scales against time. This has been done in Figure B.1. It confirms our estimate for the length of the bias interval. Having determined this, the entries in the first table corresponding to a bias interval of 11 minutes are chosen. In this case the entries chosen are:

- U = 85.36 %
- C = 8134 bits/frame
- PL = 9.33 % for 16KBPS
- PL = 27.12 % for 32 KBPS
- PL = 28.191 % for 50 KBPS
- PL = 9.999 % overall

```
NEWNET  2
CONNECTIONS      1- 2/1544.0 / 0.0 / 0.0 / 4.0
PROCNUM        1- 1    2- 1
FRAMETIME  10
HOST  1   H 2:1.00   V 2:1.00
HOST  1         A: 0.00       D:0.00       L: 60.00       T:  5.00
HOST  2   H 1:1.00   V 1:1.00
HOST  2         A: 0.00       D:0.00       L:  0.00       T:100.00
TRACE  DSK:AV1.T2
ROUTEUPDATE    500
NOPKTINTERVAL  500
TIMEOUTDELAY   125
LINEQUALITY     10
```

```
HELLOLIMIT        2
RETXLIMIT         3
WMOVEDELAY         0.00500
MAXHOLDQ           1  128000
MAXHOLDQ           2  128000
MAXINFRAMEQ        1 1280000
MAXINFRAMEQ        2 1280000
VS 1/L:     0 R:     0 S:      0
VS 2/L:     0 R:     0 S:      0
VS 3/L:     0 R:     0 S:      0
VS 4/L:     0 R:     0 S:      0
Pktlen  2000
SEED         10
MODE    VOICE
VFRACTION         0.500
VRATE          4000
VDISTRIB    2400/ 0.100  4000/ 0.100  8000/ 0.150 16000/ 0.500 32000/ 0.100 50000/ 0.050
ALLOC  VFV
SIMULATION PERIOD          2700000.000
```

SIMULATION RESULTS
****

I. LIST OF SIMULATED CALLS

CALL	ORIG	DEST	RATE (KBPS)	CONCTTIME (MILLISEC)	HOLDTIME (MIN)	HOPS	BLOCKED AT NODE
1	1	2	16.0	19.778	8.71	1.0	
2	1	2	32.0	11.173	8.91	1.0	
3	1	2	32.0	13.772	18.80	1.0	
4	1	2	4.0	13.089	14.93	1.0	
5	1	2	8.0	12.392	9.74	1.0	
6	1	2	2.4	19.985	17.99	1.0	
7	1	2	50.0	16.937	9.67	1.0	
. . . . .							
85	1	2	16.0	17.816	8.94	1.0	
87	1	2	16.0	21.137	18.69	1.0	
88	1	2	32.0	0.031			1
89	1	2	16.0	0.027			1
90	1	2	16.0	0.035			1
91	1	2	16.0	17.801	1.36	1.0	
92	1	2	32.0	21.320	9.96	1.0	
. . . . .							
452	1	2	4.0	23.031	0.16	1.0	
453	1	2	16.0	17.938	5.41	1.0	
454	1	2	32.0	0.063			1
455	1	2	4.0	19.156	0.57	1.0	
456	1	2	16.0	0.031			1
457	1	2	2.4	15.813	7.08+	1.0	

II. NODE ACTIVITIES

NODE	TOTAL CALLS	CMPLTD CALLS	BLKD CALLS
1	457	413	44
2	0	0	0

III. STATISTICS

B	U	C	Blocking Probabilities (%) 16000 KBPS	32000 KBPS	50000 KBPS	Overall
44.00	82.38	7717	11.888	40.909	50.000	12.406
43.00	88.60	7717	11.951	40.222	53.000	12.264
42.00	90.67	7730	11.777	38.481	52.794	12.104
41.00	90.03	7857	11.488	37.835	52.690	11.964
40.00	90.57	7901	11.304	37.110	52.152	11.842
39.00	89.81	7984	11.201	36.781	51.355	11.741
38.00	90.38	7952	10.895	36.546	50.368	11.639
37.00	90.16	7988	10.637	35.906	49.219	11.533
36.00	89.98	8016	10.335	35.284	47.917	11.422
35.00	89.33	7998	10.124	34.568	46.458	11.310
34.00	89.02	7969	9.964	33.982	44.832	11.202
33.00	89.51	7951	9.847	33.301	43.477	11.098
32.00	89.52	7970	9.766	32.937	41.908	11.001
31.00	89.82	7986	9.693	32.436	40.563	10.908
30.00	88.95	8064	9.656	32.002	39.525	10.822
29.00	88.15	8132	9.638	31.622	38.760	10.743
28.00	87.20	8164	9.660	31.287	38.084	10.672
27.00	87.02	8148	9.701	30.998	37.483	10.607
26.00	86.91	8142	9.758	30.723	36.946	10.549
25.00	86.74	8137	9.798	30.387	36.462	10.497
24.00	86.82	8125	9.850	30.182	35.678	10.449
23.00	87.19	8107	9.857	29.843	34.966	10.404
22.00	87.16	8104	9.765	29.534	34.315	10.360
21.00	86.23	8136	9.693	29.295	33.811	10.318
20.00	86.09	8153	9.636	29.176	33.348	10.278
19.00	85.47	8181	9.609	29.122	32.920	10.240
18.00	85.09	8174	9.602	29.133	32.523	10.205
17.00	84.60	8168	9.613	28.807	31.808	10.171
16.00	84.35	8162	9.639	28.552	31.143	10.139
15.00	84.34	8157	9.688	28.315	30.521	10.109
14.00	84.76	8143	9.741	28.093	29.997	10.081
13.00	85.02	8135	9.677	27.885	29.506	10.054
12.00	85.34	8127	9.532	27.689	29.045	10.027
11.00	85.36	8134	9.330	27.120	28.191	9.999
10.00	85.30	8139	9.149	26.605	27.385	9.971
9.00	85.26	8144	8.986	26.118	26.625	9.943
8.00	85.16	8163	8.845	25.682	25.905	9.915
7.00	85.36	8164	8.612	25.007	25.223	9.888
6.00	85.39	8172	8.391	24.365	24.577	9.860
5.00	85.45	8180	8.181	23.756	23.962	9.832
4.00	85.38	8200	7.982	23.177	23.378	9.804
3.00	85.37	8215	7.792	22.625	22.821	9.777
2.00	84.68	8271	7.611	22.099	22.290	9.750
1.00	83.28	8392	7.438	21.597	21.784	9.725

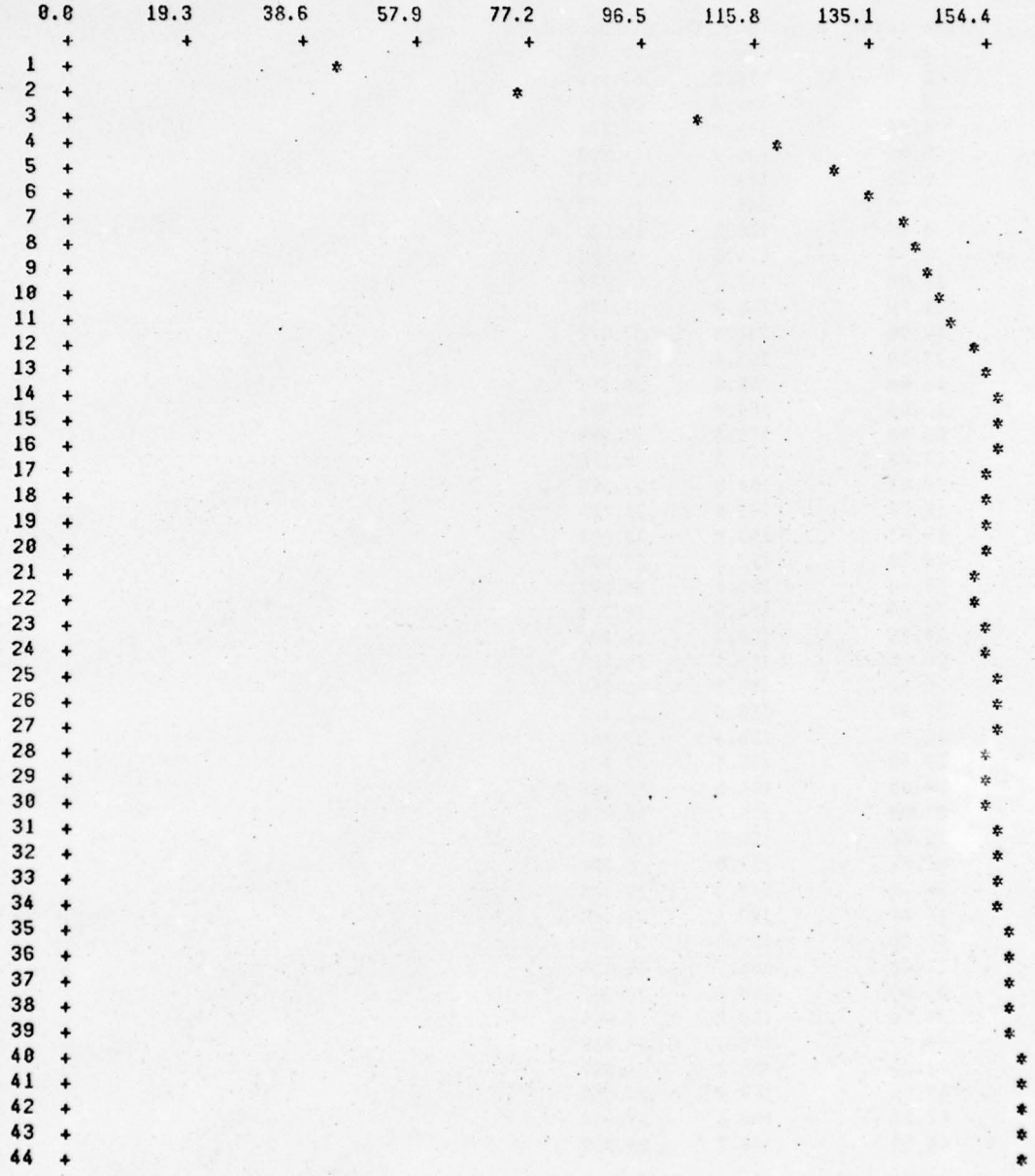
VOICE TXMN EFF: 98.506 %

B is the initial bias period in minutes
U is the utilization
C is the bits per frame available for non-class I traffic

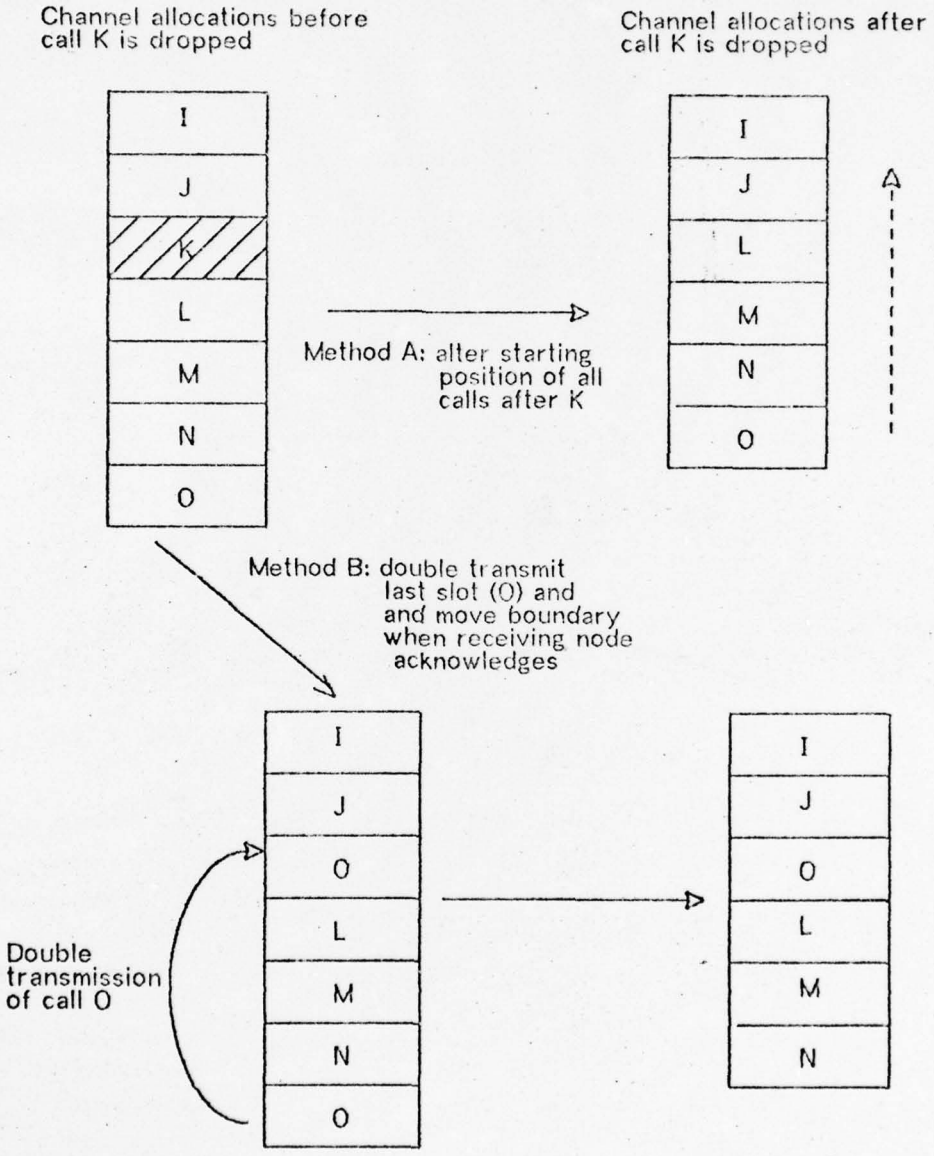
Determination of the initial bias interval:

Time (min)	Avg Busy Ch	Standard Dev
1.00	44.0	0.000
2.00	76.0	32.000
3.00	105.3	49.026
4.00	118.8	48.401
5.00	129.0	47.900
6.00	135.3	45.963
7.00	141.6	45.213
8.00	143.5	42.688
9.00	145.6	40.582
10.00	147.1	38.777
11.00	148.8	37.370
12.00	151.8	37.077
13.00	153.9	36.409
14.00	156.4	36.165
15.00	156.7	34.968
16.00	156.3	33.989
17.00	155.2	33.176
18.00	154.6	32.325
19.00	153.6	31.766
20.00	153.9	30.993
21.00	152.5	30.866
22.00	153.2	30.307
23.00	154.5	30.258
24.00	155.2	29.804
25.00	155.4	29.225
26.00	155.7	28.705
27.00	156.0	28.193
28.00	155.4	27.863
29.00	155.1	27.421
30.00	154.8	26.990
31.00	155.7	26.951
32.00	156.2	26.697
33.00	157.0	26.687
34.00	157.3	26.336
35.00	157.4	25.964
36.00	157.8	25.698
37.00	158.1	25.439
38.00	158.7	25.367
39.00	158.9	25.066
40.00	159.4	24.949
41.00	159.7	24.697
42.00	160.2	24.658
43.00	160.8	24.610
44.00	160.7	24.330

TIME VARIATION OF MEAN OF BUSY CHANNELS

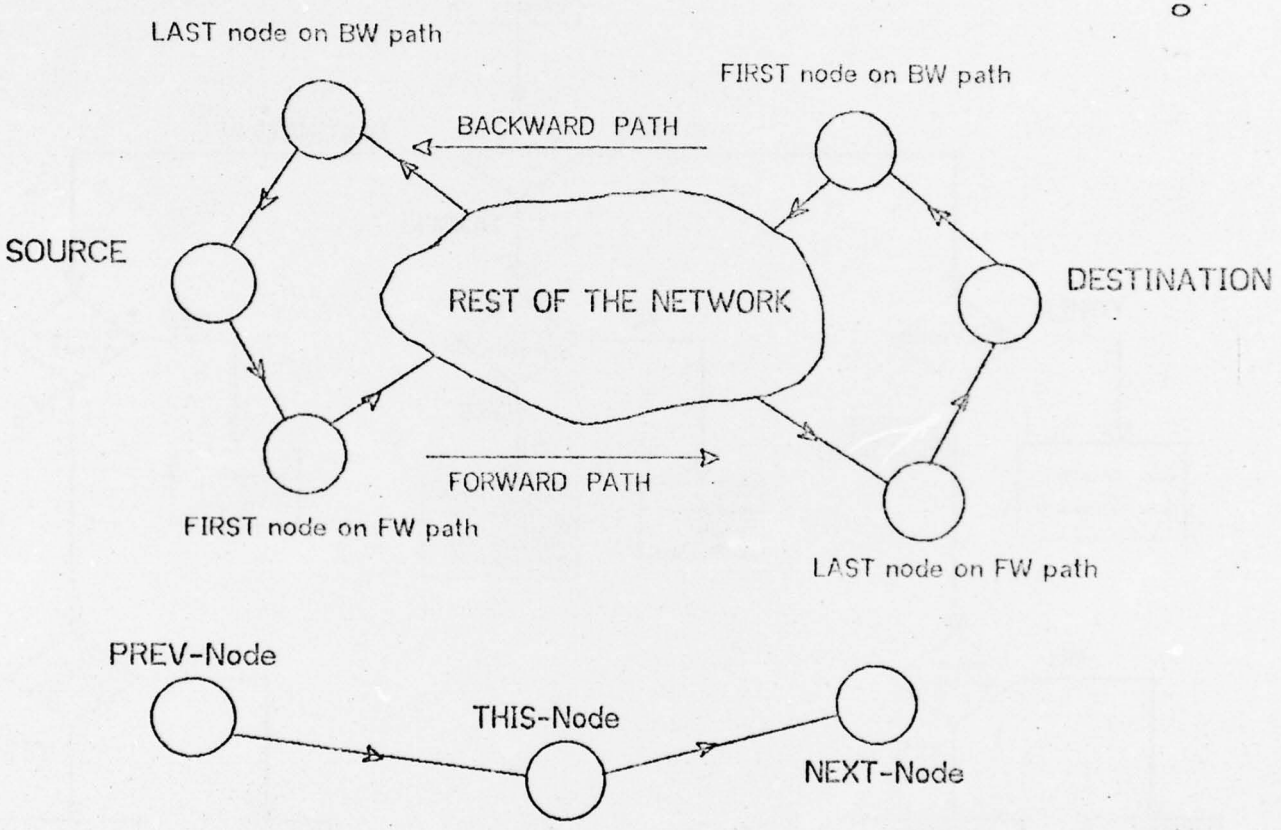


Appendix C: Figures



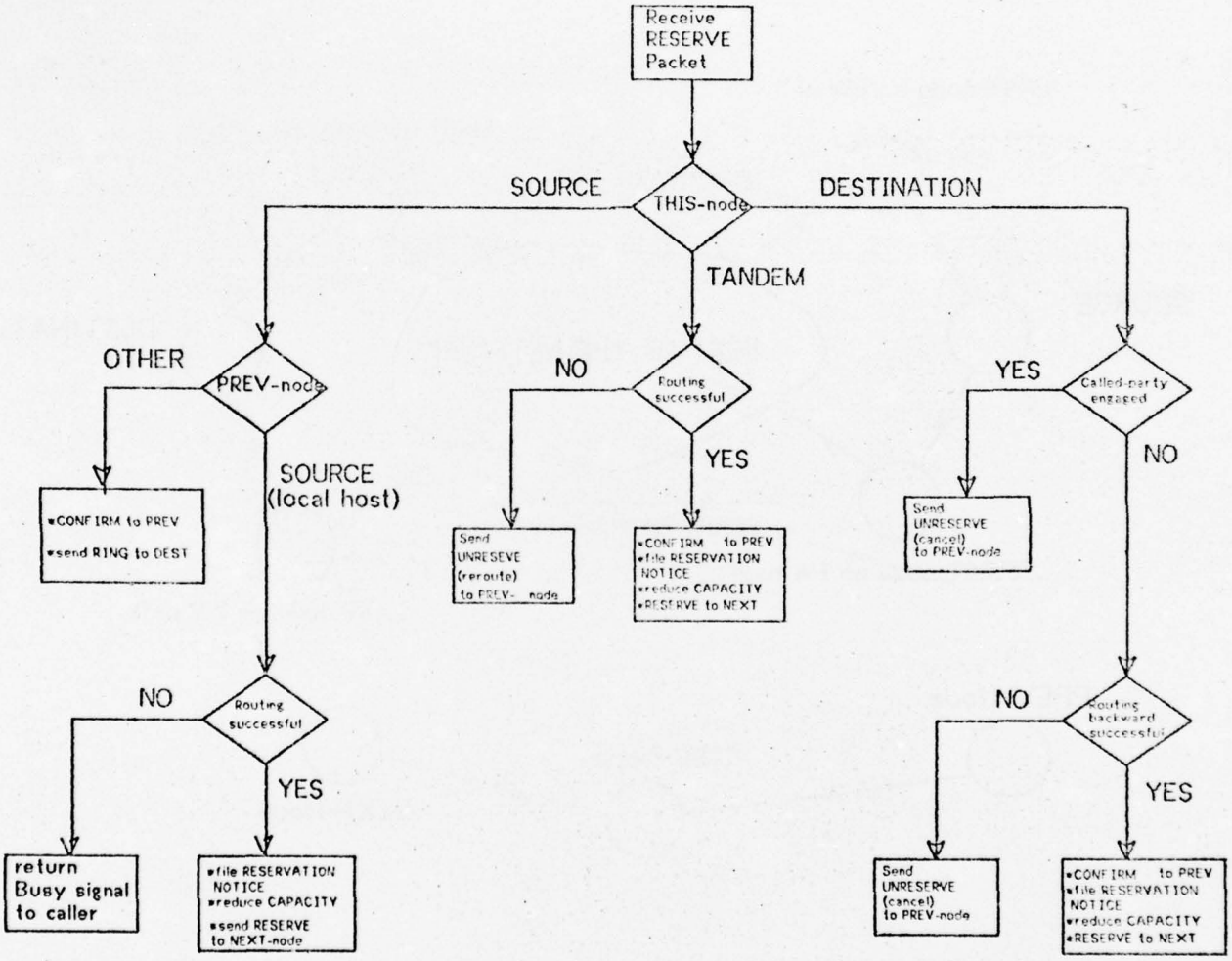
Comparison of class I region compaction methods

Figure 1.1



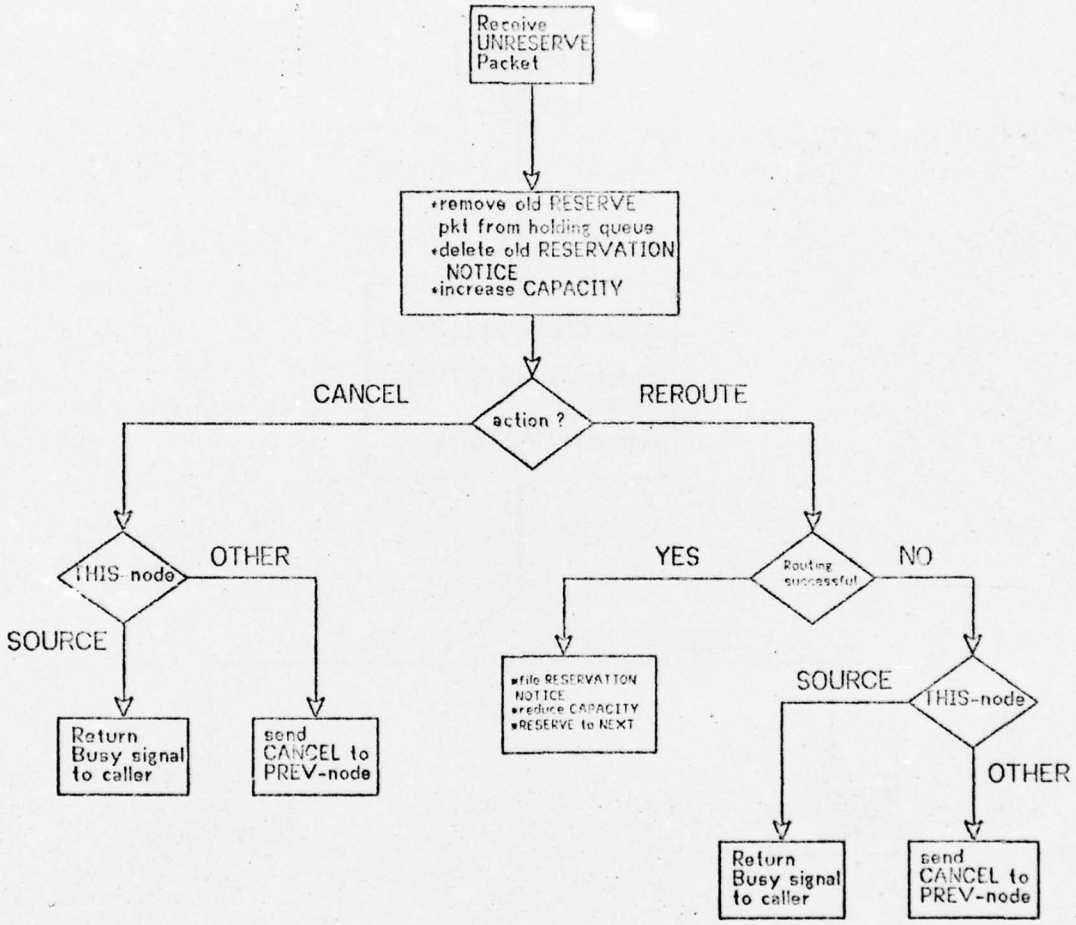
Terminology used for Class I communications

Figure 1.2



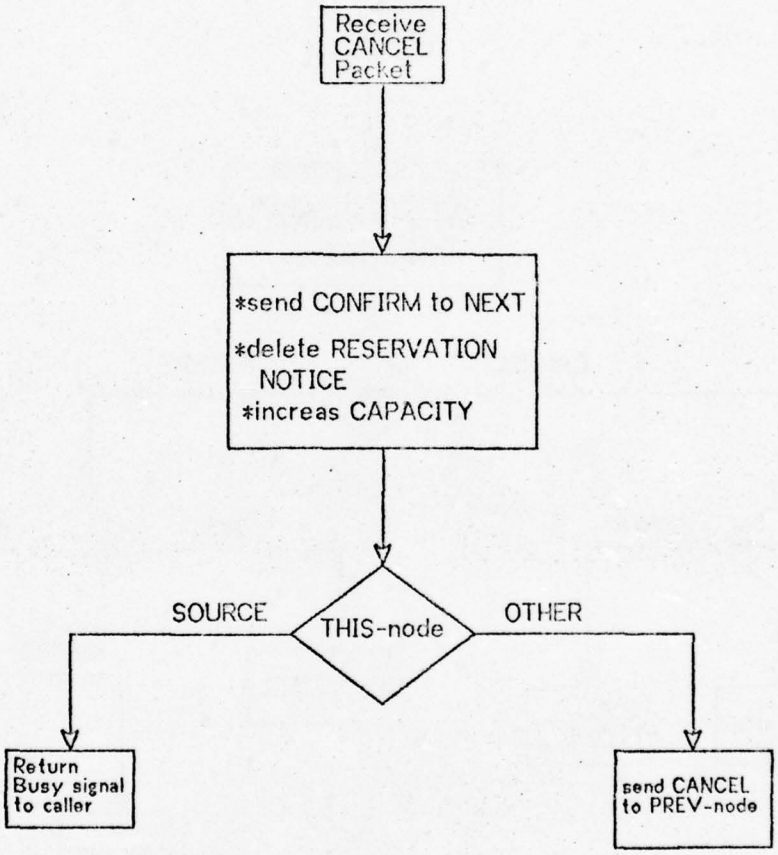
RESERVATION PROTOCOL - RESERVE Packet Control Flow

Figure 1.3



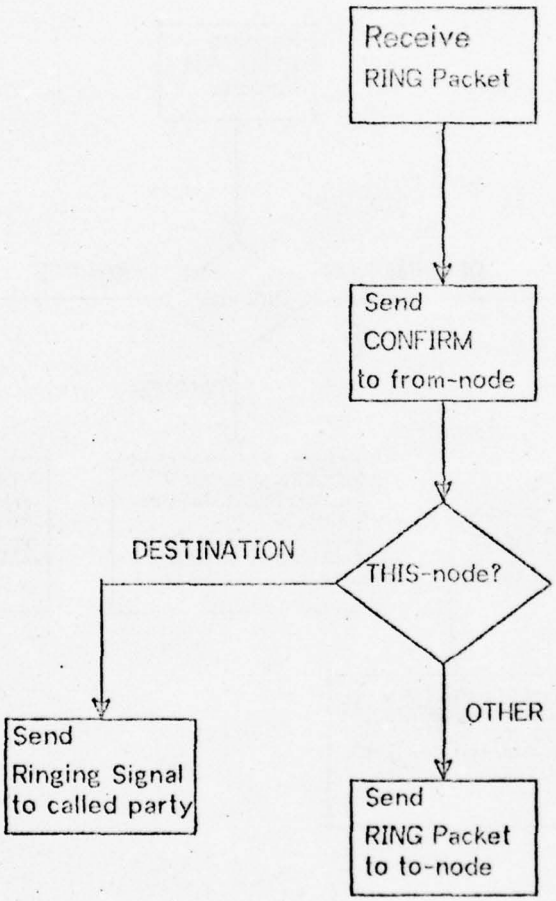
RESERVATION PROTOCOL - UNRESERVE Packet Control Flow

Figure 1.4



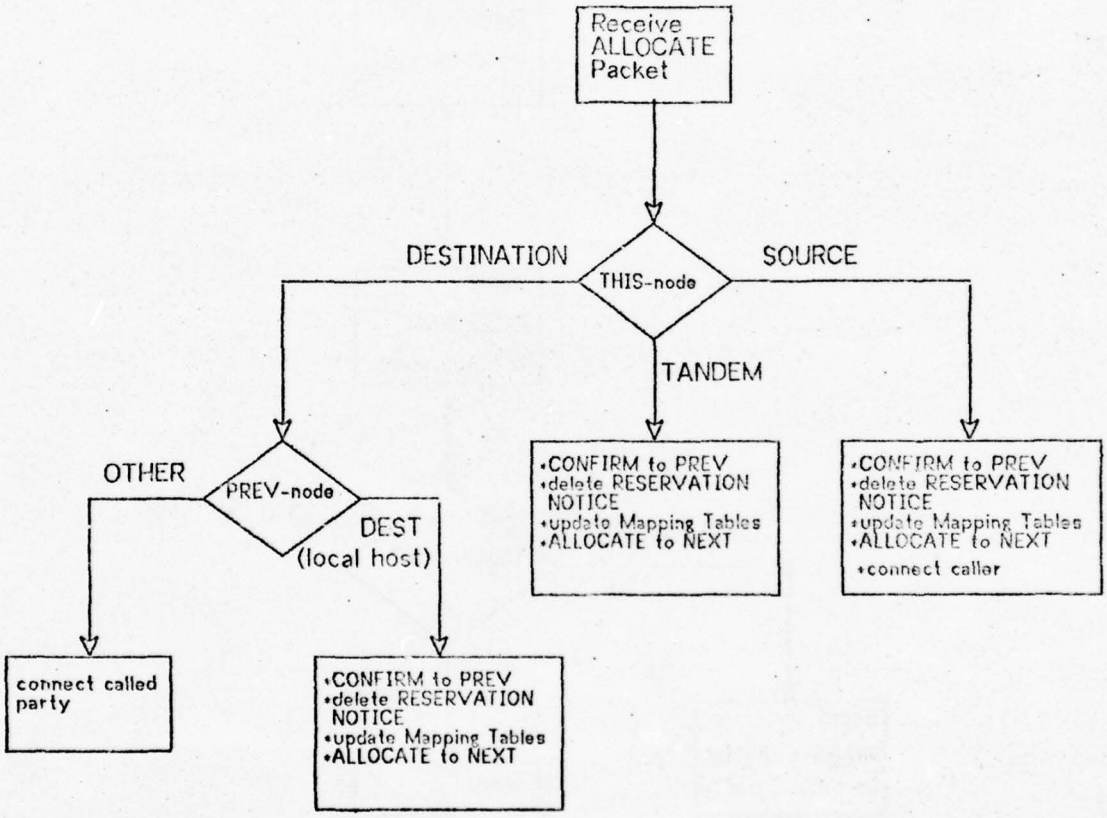
RESERVATION PROTOCOL - CANCEL Packet Control Flow

Figure 1.5



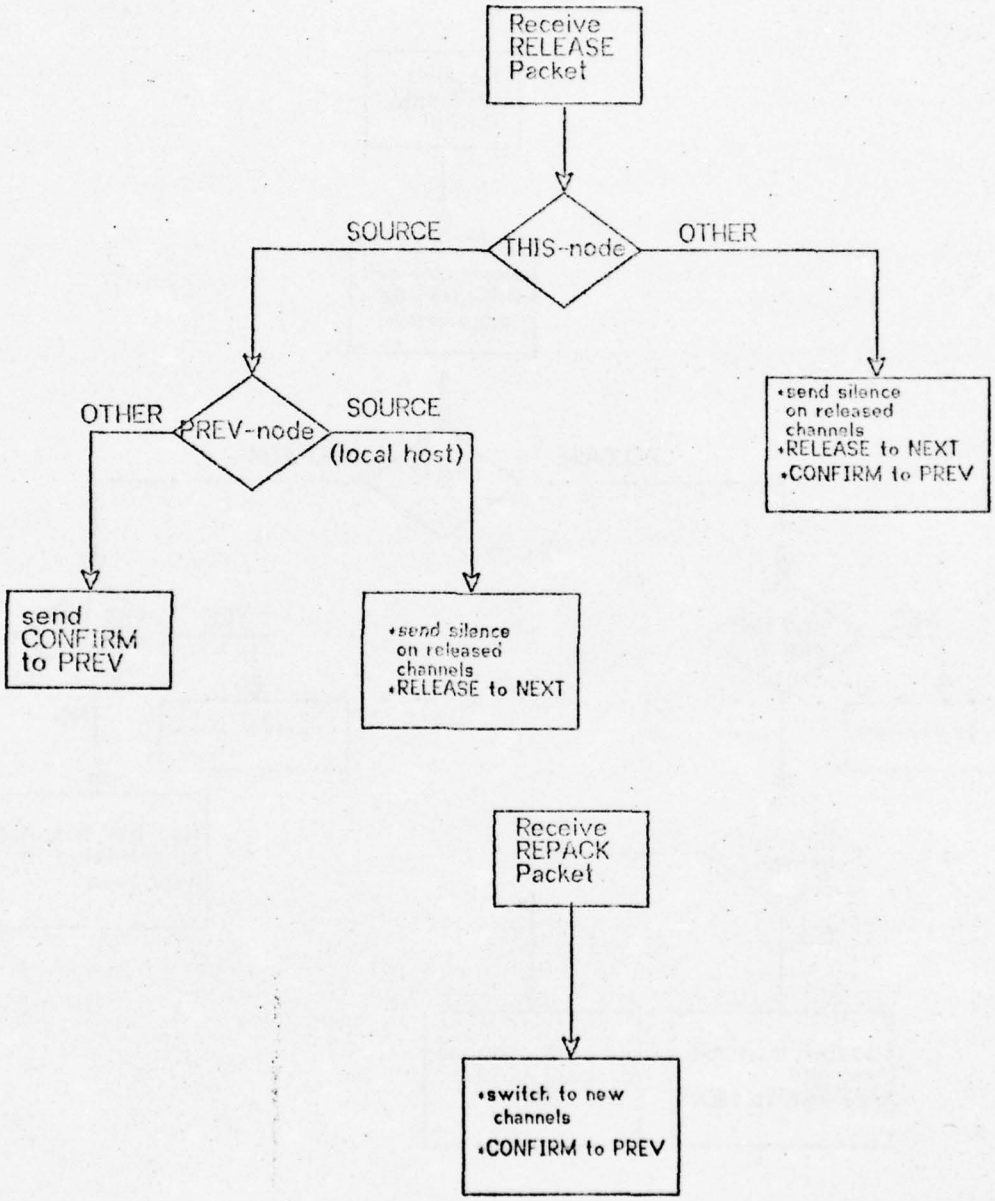
RESERVATION PROTOCOL - RING Packet Control Flow

Figure 1.6



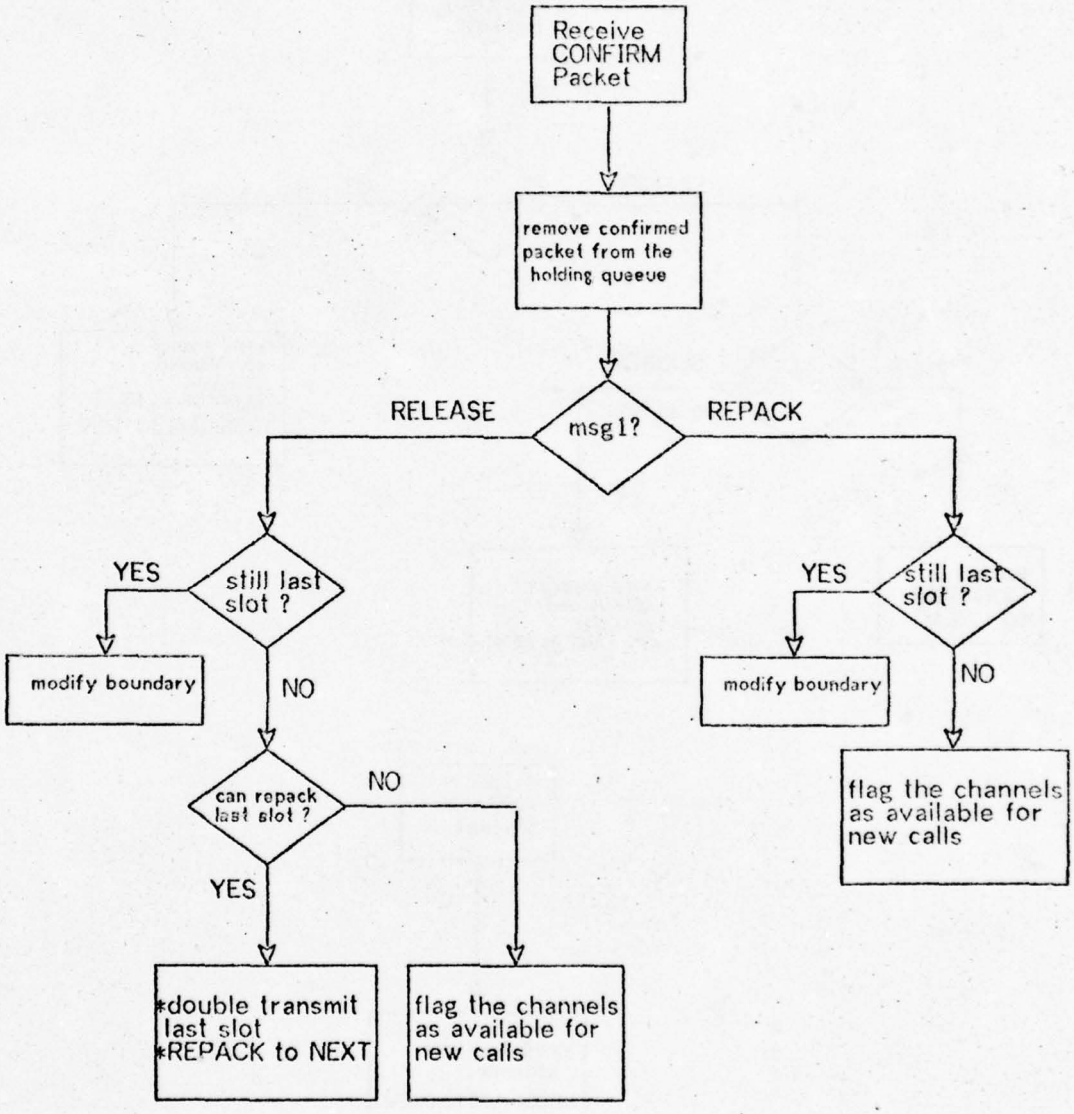
ALLOCATION PROTOCOL - ALLOCATE Packet Control Flow

Figure 1.7



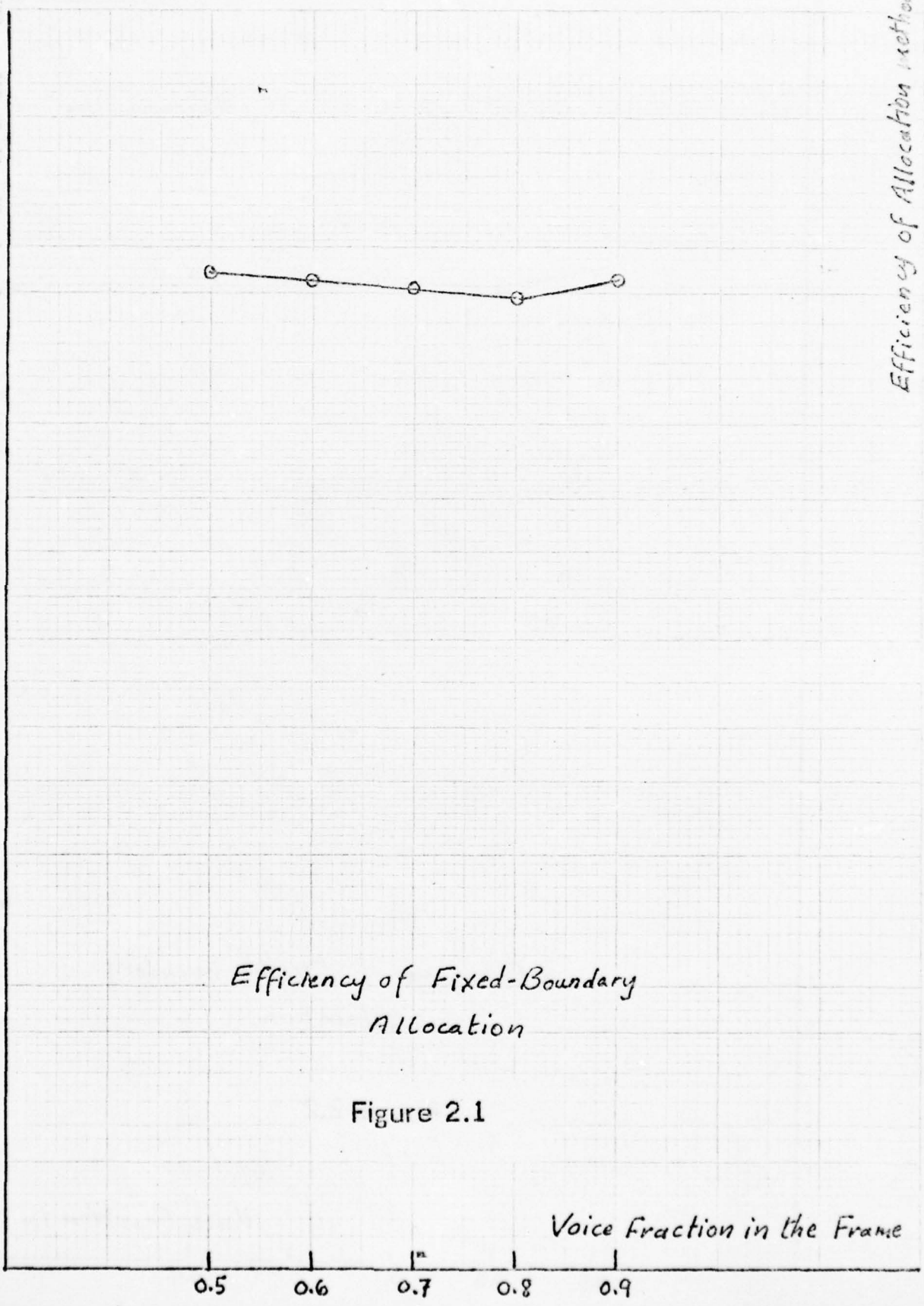
TERMINATION PROTOCOL - RELEASE and REPACK Packets CONTROL FLOWS

Figure 1.8



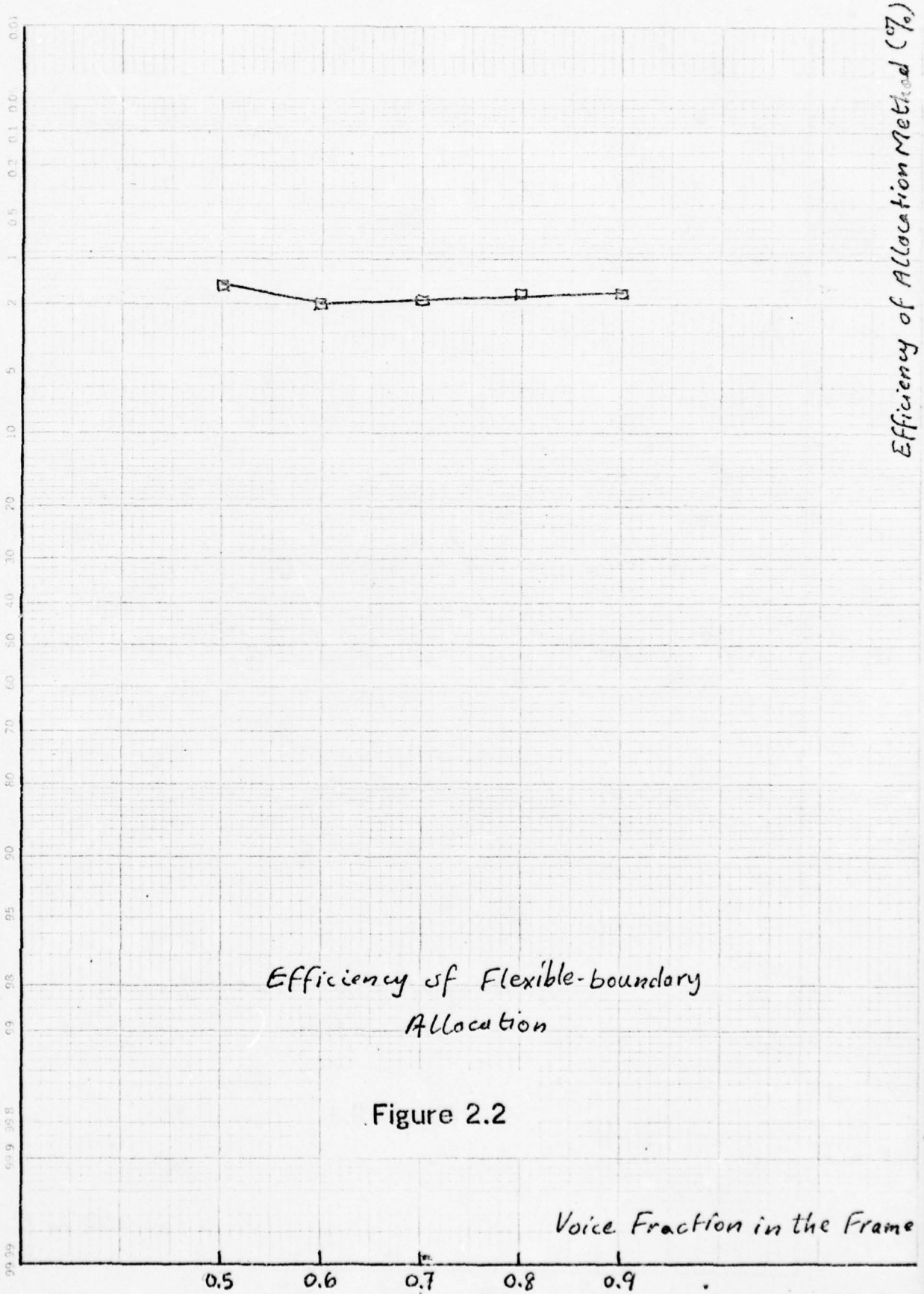
TERMINATION PROTOCOL - CONFIRM Packet Control Flow

Figure 1.9



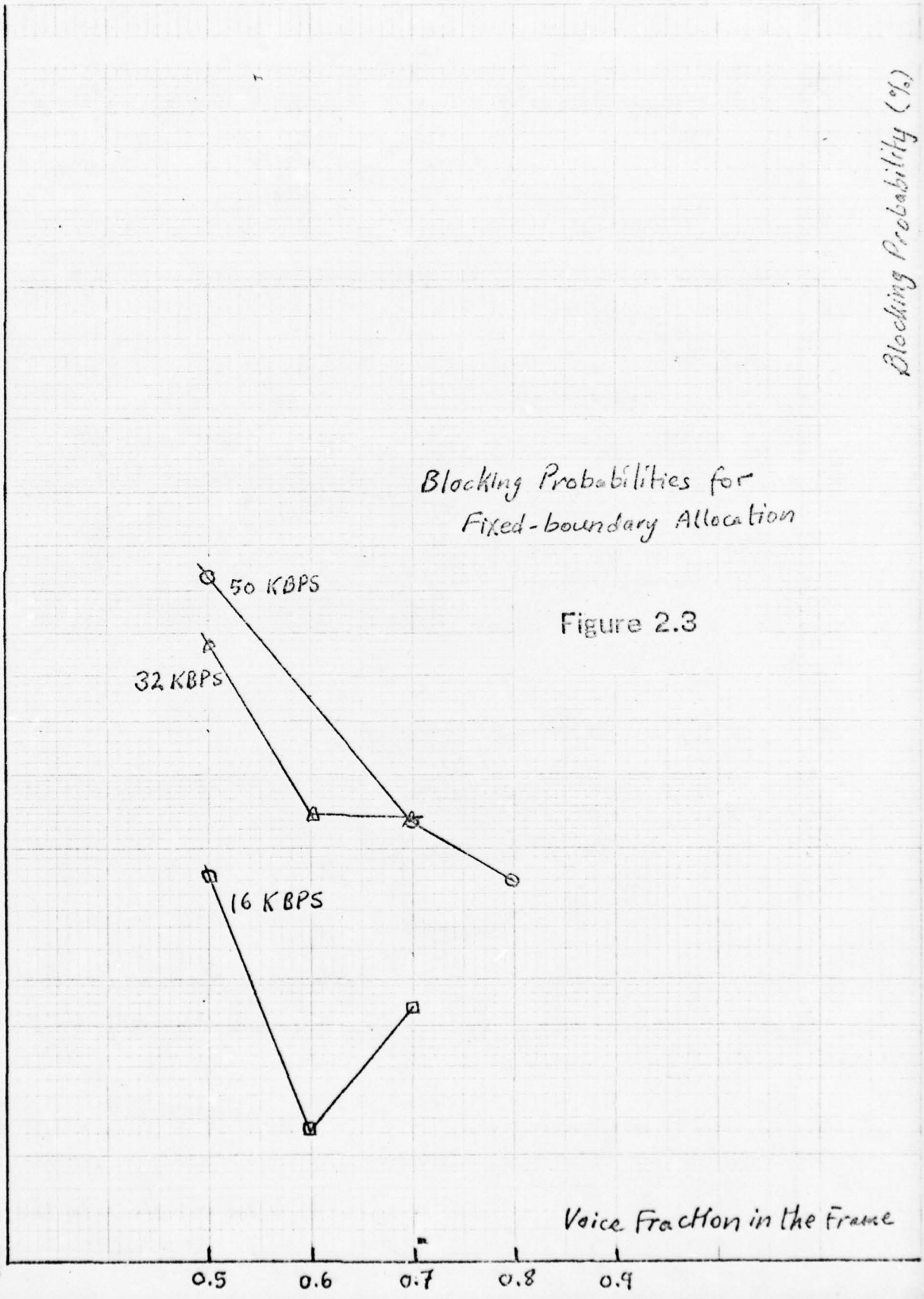
Efficiency of Fixed-Boundary Allocation

Figure 2.1



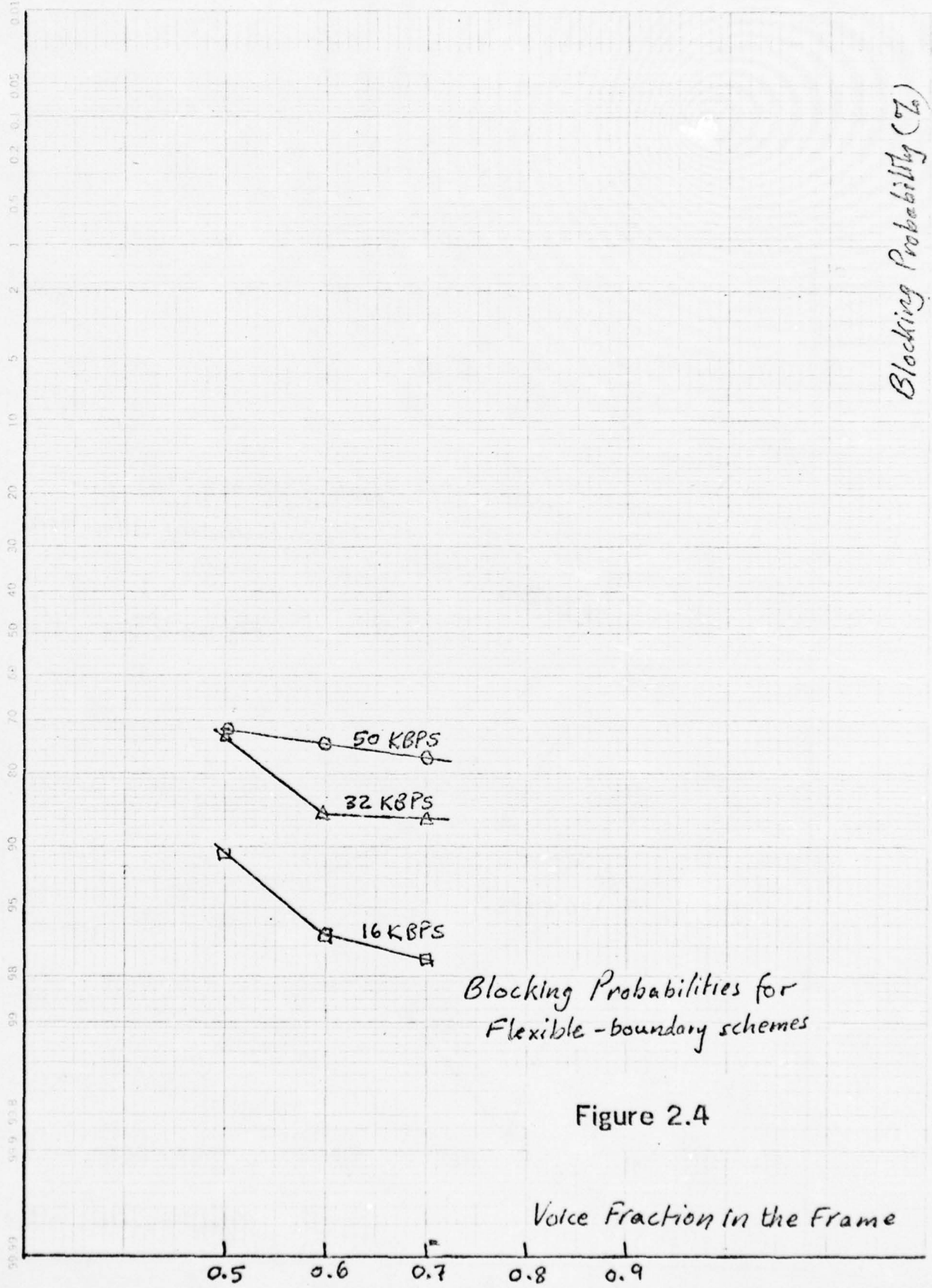
Efficiency of Flexible-boundary Allocation

Figure 2.2



Blocking Probabilities for Fixed-boundary Allocation

Figure 2.3



Blocking Probabilities for Flexible-boundary schemes

Figure 2.4

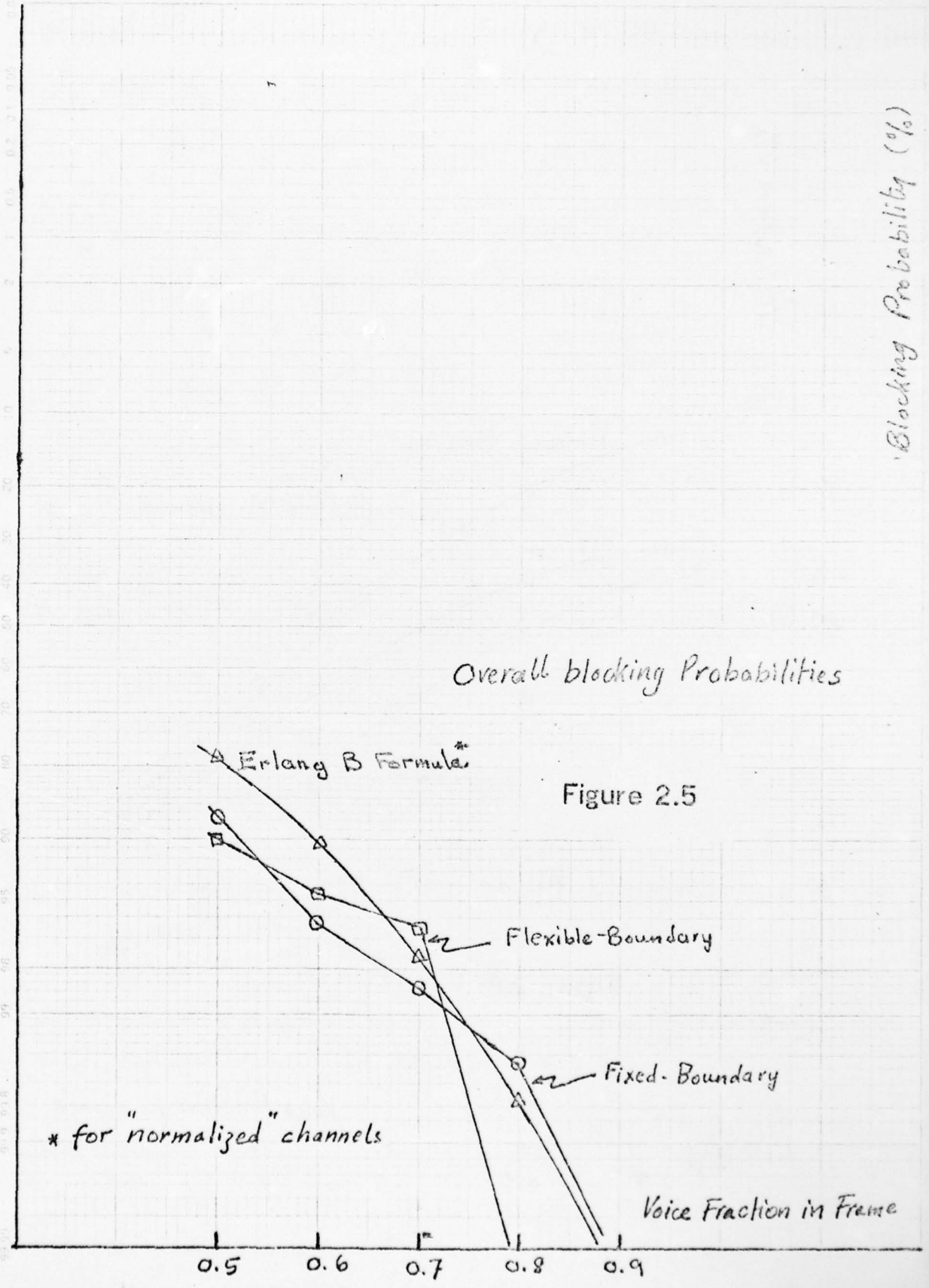

Blocking Probability (%)
Overall blocking Probabilities
Erlang B Formula*
Flexible-Boundary
Fixed-Boundary
* for "normalized" channels
Voice Fraction in Frame
0.5
0.6
0.7
0.8
0.9

Figure 2.5

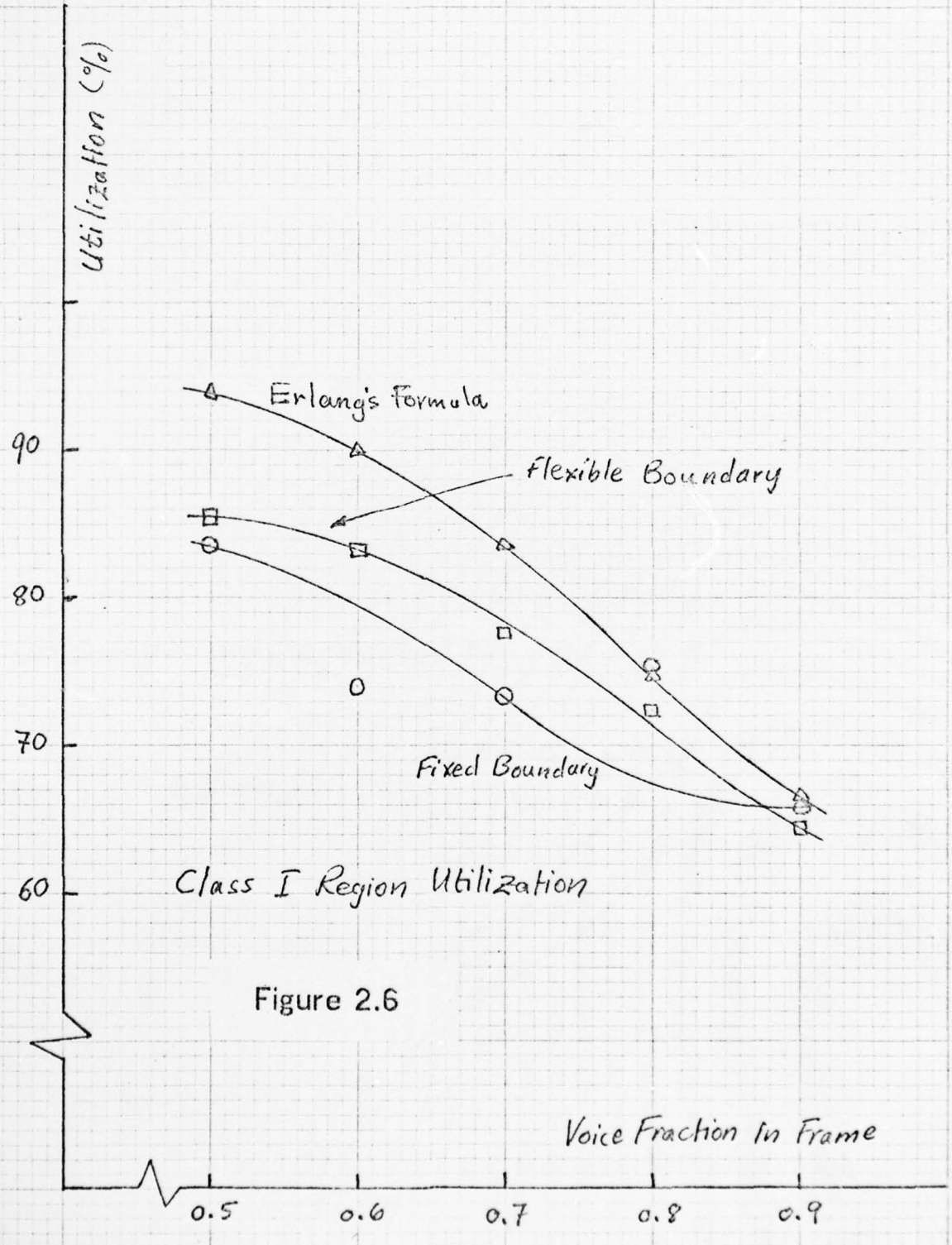

Utilization (%)
90
80
70
60
Erlang's Formula
Flexible Boundary
Fixed Boundary
Class I Region Utilization
Voice Fraction in Frame
0.5
0.6
0.7
0.8
0.9

Figure 2.6

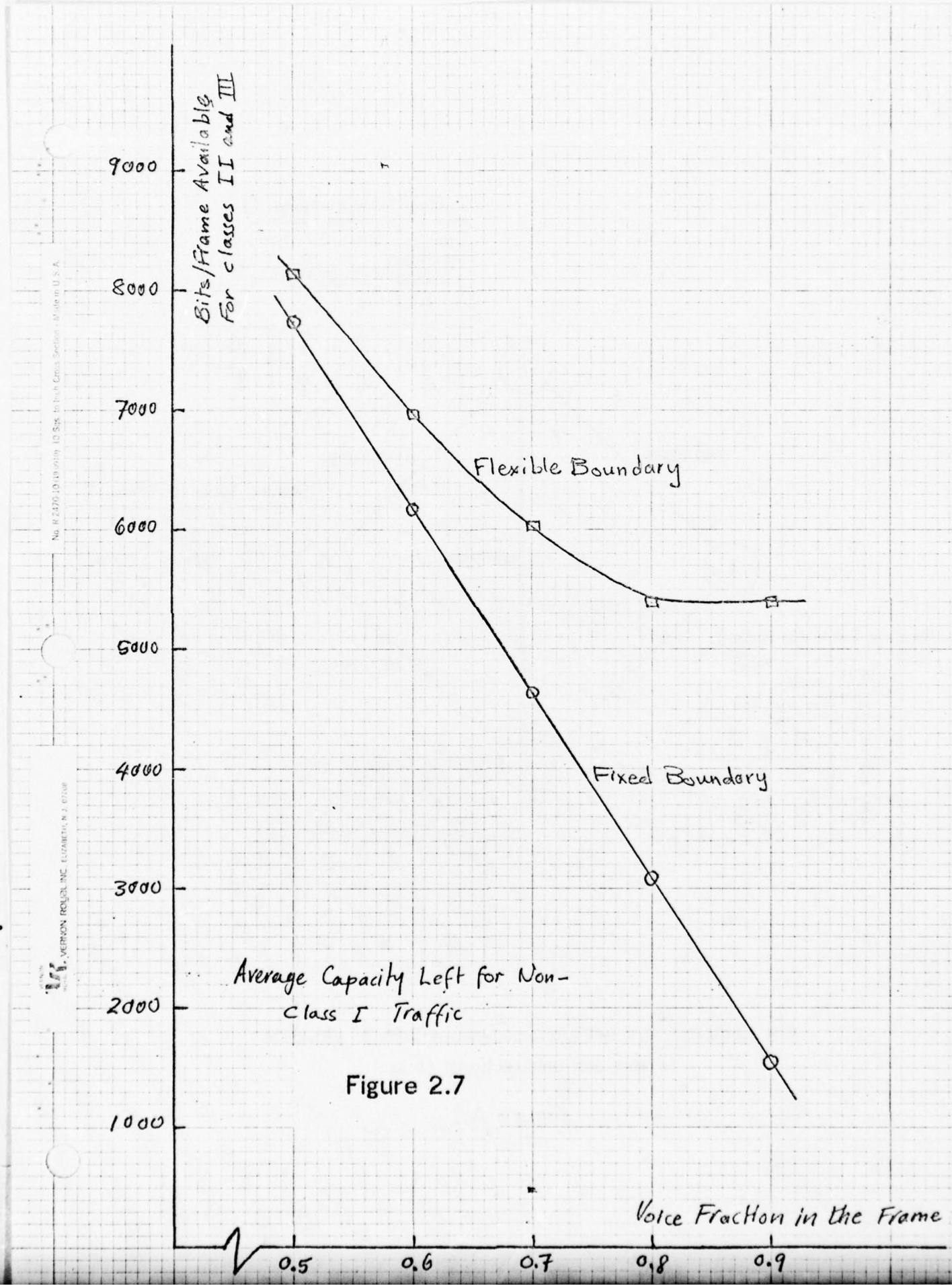
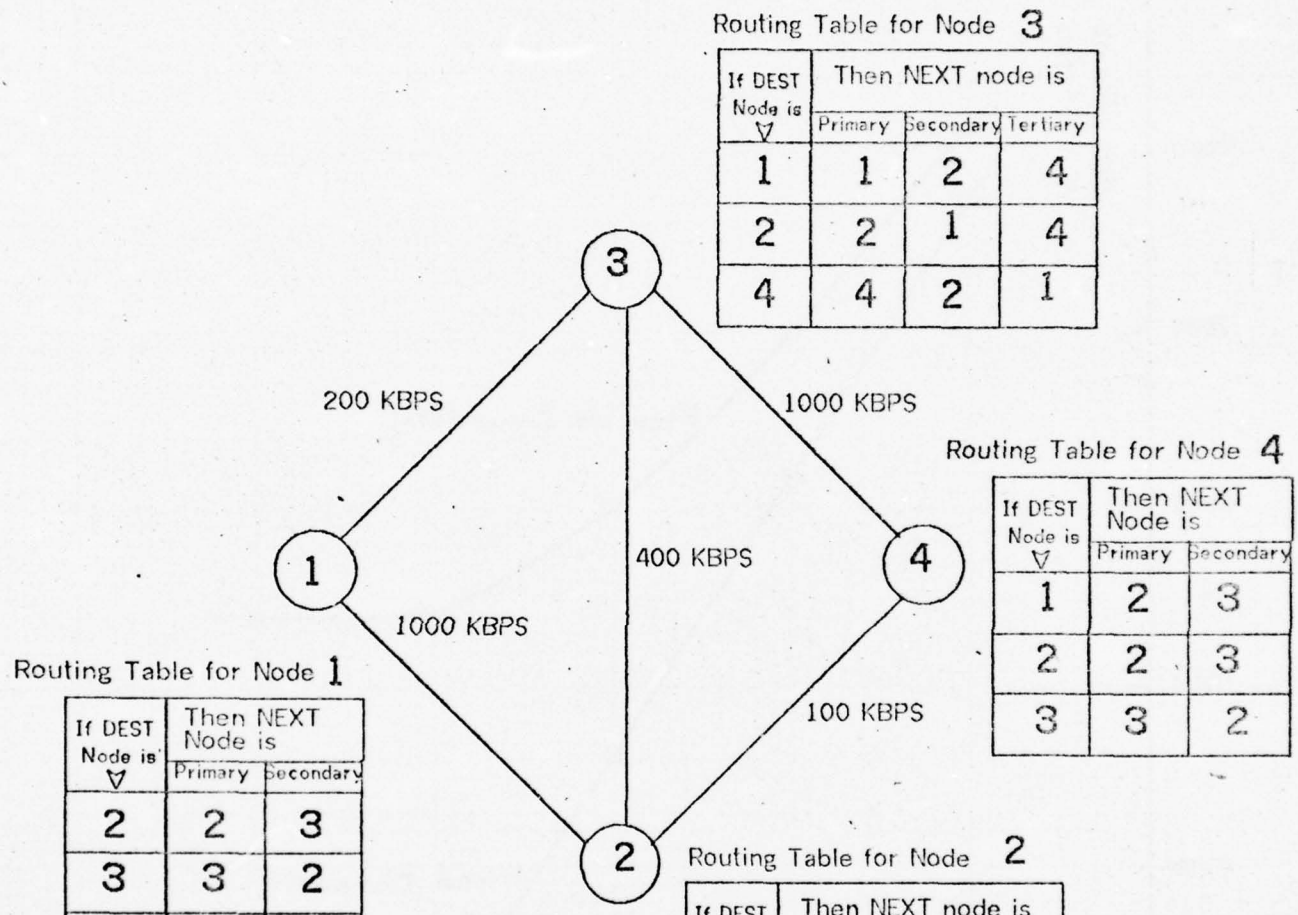

Bits/Frame Available
For classes II and III
9000
8000
7000
6000
5000
4000
3000
2000
1000
Flexible Boundary
Fixed Boundary
Average Capacity Left for Non-
Class I Traffic
Voice Fraction in the Frame
0.5
0.6
0.7
0.8
0.9

Figure 2.7



4 - Node network with inhomogeneous links to illustrate routing and unidirectional control

Figure A.1

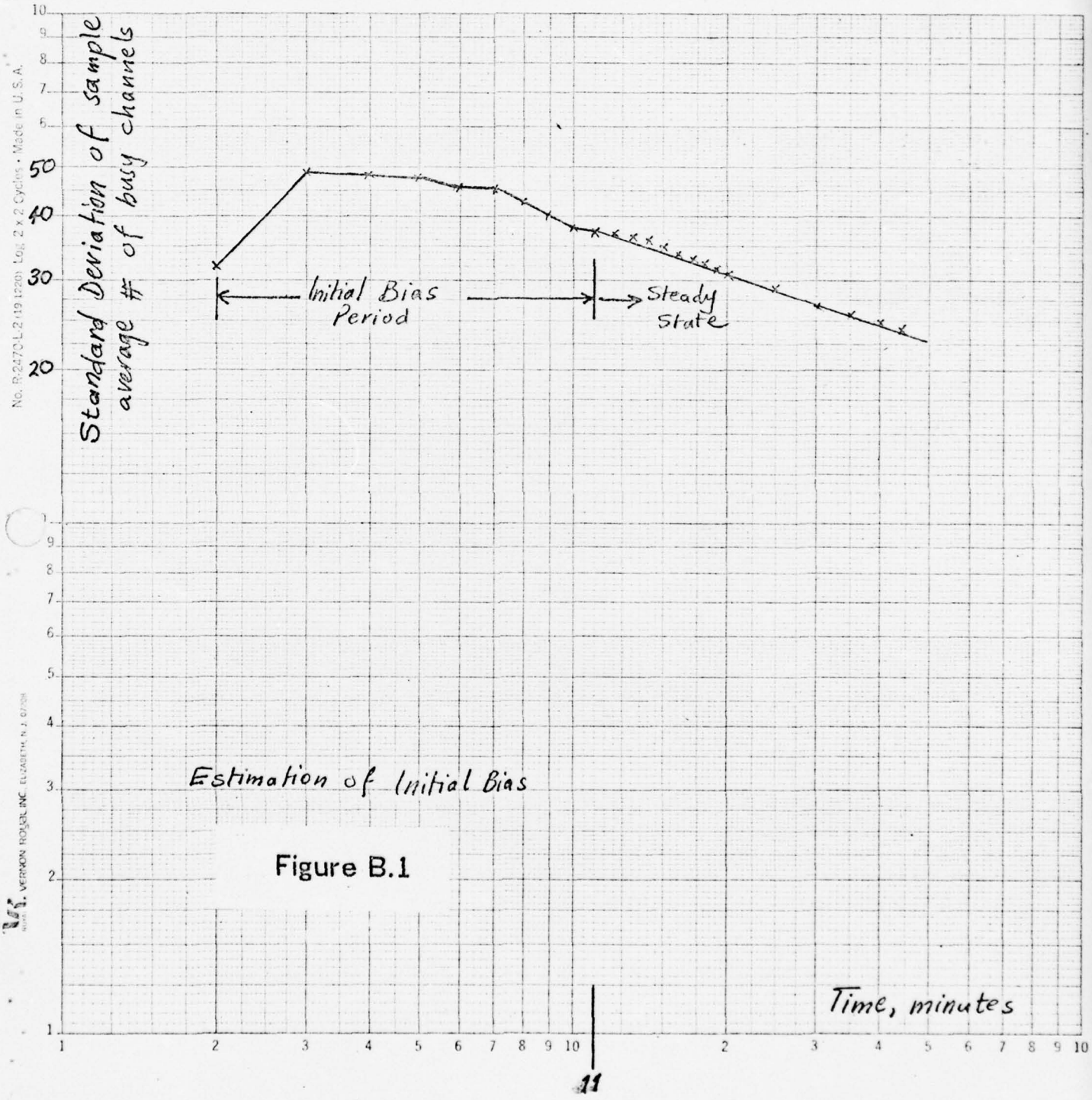

Standard Deviation of sample
average # of busy channels
Initial Bias
Period
Steady
State
Time, minutes
Estimation of Initial Bias

Figure B.1

Part III

Flow and Error Control

William Wu
Mario R. Barbacci

1. Flow Control and Congestion

1.1. Introduction

In any communication network, there is a limit to the traffic it can carry. If the traffic demand exceeds certain established limit, some of the traffic would have to be rejected. The state in which the network has to reject some traffic is congestion.

Congestion is a type of network failure which would demand more and more attention as the network and amount of traffic grows as it could degrade the network performance severely. The present ARPANET flow control works quite well. (A brief description of control schemes with ARPANET as an example could be found in ?) However, it is observed that it may be only because most of the time the channel is under utilized. The busiest channel in ARPANET [KLE74] has an average utilization of only 20 % and peak of 50 % only. Excluding the overhead, the peak utilization is only about 25 %. A discussion of the source of congestion could be found in section 2.7.

Besides, the ARPANET flow control scheme is not without inefficiency. Since it lacks the instantaneous current global knowledge of the state of the network transmission, it cannot make the best and the fairest routing and flow control decision. Consider another example, a node with two routes to its destination. With the present routing scheme only one preferred route is being used, if two routes could be used at the same time the maximum amount of traffic between the two nodes could have been doubled. (see 6 for an simulation example)

Using a SENET concept with dynamic boundary between voice and data region , adds extra flexibility and a new dimension to the already complicated control scheme. The

use of multiprocessors adds reliability to the network since graceful degradation (with reduced processing power) at a node is possible. To make best use of the new ability, the number of processor at a node should be taken into account during flow control decision.

As there is no real analytic way to design a flow control scheme and most of the scheme has been approached [FRA75] in an ad hoc fashion, the network characteristics were studied (with no global flow control) for future reference in designing a control scheme. There are two classes of flow control - local and global. (see 1.2). We are at present interested in local control only and hence no global control is implemented here. In other words, the network wide congestion is assumed improbable here.

1.2. An Overview of Flow Control Scheme

There are four major objectives in flow control [FRA73]:

1 To avoid severe performance degradation resulting from congestion.

2 To prevent crippling deadlock.

3 To maximize resource utilization and minimize delay.

4 To provide fair allocation of communication resources.

There are two classes of flow control mechanism that can be distinguished -local and global. Local control is concerned with decision making on the basis of information available within a specific node or obtained from its immediate neighbors. It is a direct consequence of the limited buffer space in each node. Adaptive routing is the local control used in the ARPANET. The global control mechanisms would required widespread knowledge of network activity to put some limit on the total number of packets in the network.

1.2.1 Arpanet Local Control

Adaptive routing is the major technique used in preventing local congestion. About twice every second, a node (IMP) exchanges with its immediate neighbor its current routing information[GER75]. The information received by a node is combined with its own knowledge to compute the routing table which gives the best output line on which to place a packet bound for each destination. So packets could be routed from locally congested area, over a whole region of network instead of a fixed route (see 6 for an example). A more detailed description of the scheme could be found in 1.3.

1.2.2 Arpanet Global Control

Source / Destination Control [KAH71] It is concerned with the prevention of congestion in the store- and-forward subnetwork. Buffer storage for a particular message at the destination node has to be allocated first before the message is allowed to enter the network. The source no issues an allocation request to the destination, only when storage has been reserved at the destination and allocation control message returned in reply does the source node accepts the remainder of the message from the host and forward it through the network. To allow high throughput, subsequent messages to the same destination are allowed to bypass the handshaking described above. The destination node, upon completely receiving the message, automatically allocates buffer storage for another message and returns notice of the allocation with the "request for next message" (RFNM) acknowledgement to the source.

Host to Host Flow Control [KAH71]

It is concerned with fair distribution of available host buffer space among several users. The receiving host controls message flow by periodically authorizing the sender to send a given number of messages or bits.

1.3. Present Local Flow Control Scheme

A local control scheme quite similar to the ARPANET adaptive routing scheme is implemented in the simulation. The scheme could be explained with the following example.

Consider a network with N nodes, each node i has an N * A_i matrix (A_i is the number of links connected to node i) called delaytable. Each of the entry of the delaytable i (k,l) is the estimated minimum delay from node k to node l. From this table an N-dimensional vector called the delayvec is obtained. Each of the entry of this vector delayvec [l] of node i is the calculated minimum delay from node i to node l. Mathematically,

delayvec[l] of i:= MIN (delaytable[k,l] +
delay from i to k+
length of the holding queue of node i)
where k < A_i

From the delaytable a similar N-dimensional vector called routingvec is also obtained. The routingvec[l] of node i tells it which immediate destination to send a packet bound to some final destination (node l). That is, for node i

delayvec[l]:= delaytable[routingvec[l],l] +
delay from i to k +
length of the holding queue of node i

In other words, routingvec[l] points to the best minimum delay route. Periodically, each node will update its routingvec and delayvec and send out to its neighbors its delayvec. Each of the neighboring nodes will copy the delayvec into its delaytable and used it in the future routingvec and delayvec computation.

The table could reflect network congestion and node or link failure. However, disconnected node cannot be detected rapidly enough from delay vector, a special protocol for this is needed (see section on 2) As the delaytable could not be completely up to-date (i.e. immediate state of the network is not reflected in the table) during delayvec and routingvec calculations, routing loops which could degrade the network performance severely could occur. This problem was encountered in the ARPANET, it was prevented by instead of modifying the vectors every 500 ms, the new vectors calculated are kept and modified for 3 routing updates while the old vectors are still being used during this period. At the end of the 3rd update the old one would be replaced. This apparently provides enough time for a node to get enough information of the network to prevent routing loops. However, the adaptiveness of the scheme is sacrified and the response time of the scheme to congestion is decreased.

It is felt that better adaptiveness and response would be needed to make best use of the new SENET concept where the resources for data packets depend on voice traffic. It was also noted that the delayvec computation is not sensible for some cases. For example, in a straight line network 1 - 2 - 3 - 4, for node 3 to calculate the delayvec or routingvec to node 1 would be meaningless. Routing loops could occur when node 2 is congested and node 3 erroneously considers node 4 as the shortest route to node 1 under our present implementation.

1.4. Behavior of a Network Under Various Load Levels

The immediate effect of high load level in a network would be the buffer storage requirement for packets to and from the node and I/O buffering. A series of test with various load level was conducted on a simple network as shown in figure 5.1.

The two storage requirements considered were the holding queue (for storage of transmitted packets which are awaiting for their acknowledgements) and the input queue. (for storage of input packets waiting to be processed) There are four possible combinations of buffer condition :-

1 Both queues are not full. Traffic is not high enough to cause congestion. The time between putting a packet into the output queue and the packet being examined by the destination is essentially less than a frame time.

2 The holding queue is not full, but the input queue is full. The bottle neck is the no. of processors to handle the incoming traffic.

3 The holding queue is full, but the input queue is not. Here the bottle neck is the output device (e.g. line) or the destination node. It was observed that the input queue becomes full rapidly if the holding queue remains full.

4 Both queues are full. This is the final state of state 3 if state 3 is not checked. Potential deadlock could occur when the holding queue is too full to accept any packet from the input queue and the input queue is too full to receive any acknowledgement which could ease up the congestion in the holding queue.

Case two had been studied with two different kinds of test. In one case the source of packets generation was assigned infinite amount of memory, whereas a more realistic size of memory was assigned in the other series of test.

1.4.1 Description of Experiments and Results

Infinite Memory For Holding Queue

The data file and results could be found in section 5. In this series of experiment, node 3, where most traffic was generated was given a large memory for the holding queue (640,000 bits). This is similar to having a great no. of nodes represented by node 3. The large memory is to keep node 3 from being congested, so that node 2 would truly experience the traffic flow desired. The centre node was

equipped with only one processor, so that congestion in the input queue of node 2 (the centre node) would occur. The results that are of interest are :

1 The delay time It is the time between putting a packet into an input queue and the time the packet is sent out from the node.

2 The % rejection It is the % rejection of input packets by the line controller of center node.

Two series of tests were done. Each series consists of four set of data which are almost the same, except that the rate of generation was varied. The rates ranged from .6 packets/ms to .9 packets/ms. The two series were run with a different random seed for their random number generator. It was found that the results from the two series are quite close. (see figure 5.2 and figure 5.3)

The mix of the traffic in the simulation was 50 % type 3 and 50 % type 4 packets. It was found from the tests that since type 2 and 3 had higher priorities than type 4 packets, their delay distributions were quite constant (remained between 0 ms -20 ms) throughout all the tests. However, type 4 packets experienced quite a wide range of delay.

Congestion of the centre node occurred between the rate of .7 packets/ms to .8 packets/ms. Congestion degraded the network performance severely. Delay of type 4 packets increased drastically as the network approached congestion. In fact, the occurrence of congestion could be seemed from histograms for the delay of type 4 packets(see section 5). As the packet generation rate increased the delay of type 4 increased accordingly.

% line rejection remained around 32 % for packet rate of 0.8 and 0.9 packets / ms. The different is not too significant here because the simulation was limited by the memory size at .9 packets/ms. Approximately .5 packets were rejected by node 3 (as its holding queue was full) per 10 ms.

Limited Memory for Holding Queue

Another series of similar tests was run to observed the behavior of the network at the threshold of traffic congestion. In this series, some background traffic was added to the network (node 1 and node 2 generated data packets too). However, only the packets generation rate of node 3 to node 1 was varied. The network is more realistic here because extra large memory was not given to node 3. Essentially node 3 was allowed to have congestion if node 2 was congested.

For a description of the network see figure 5.1, In this experiment the center node in the simulation was assigned with one processor only so that congestion at node 2 could be created. The three results of interest are :-

a The delay time - the time between putting a packet into an input queue and the time the packet is sent out from the node. (figure 5.5)

b The % of packets dropped by the destination node. (designated by % dropped on figure 5.6) (i.e. packets dropped by the destination node due to congestion in the input queue/total packets that were sent to the destination node but not necessarily accepted)

c The % of new packets rejected by the node. (designated by % rejected on figure 5.7) (i.e. packets sent from local host but rejected by the local node/total new packets generated by the local host)

It was observed that when congestion (input queue filled) occurred the network performance (traffic delay) degraded rapidly. Even at the threshold of congestion (when data rate generation is .5 packets /ms) the overall traffic delay time is considerable lower than the overall traffic delay time of a slightly congested network. It was also observed that at some point the delay time of the congested route is less than the delay time of an uncongested route. It means that the average delay time over a short interval (few seconds) could not be an indication of the degree of

congestion that the traffic level could be encountering nor an indication of the amount of excess traffic.

Extreme fluctuation of % dropped and % rejected was observed. It was also observed that with different random seed for the random number generator in the simulation, completely different characteristic curve could be obtained. It could be concluded that neither of this could be used as a measure of % overload. As was anticipated, once the input queue is full, the holding queue of the sending node would become full rapidly. In our simulation, the % rejected curve followed closely with the % dropped curve, which suggests that the holding queue length could be a good indication of the condition of the corresponding receiving node if we have separate holding queue for different lines.

2. Error Control

2.1. Error Control of Node-to-Node Packet Processing

It has been observed in the literature [Bur72] that conventional random-error or burst-error-correcting codes cannot assure reliable data communication. Since it has also been shown qualitatively [Bur72] that automatic repeat request (ARQ) scheme are inherently more suitable for most data communication than forward-error-control (FEC) systems, an acknowledgement/retransmission strategy was used to handle data transmission over noisy channels.

It is assumed that the data contents of each packet would be checked by the line controllers against error due to transmission. The packet would not be passed into the node if found unacceptable. The node, not seeing the packet would not send out any negative acknowledgement signal. There are basically three ARQ schemes:

1. Store, forward and wait. This is by far the most widely used scheme. With this scheme, after sending a packet, the sending terminal would wait for positive acknowledgement from the receiving terminal before sending another packet . The scheme has the advantage of being very simple. However, it is inherently inefficient due to the idle time spent in waiting for acknowledgement or timeout for retransmission.

2. Store and forward. The same packet is transmitted repeatly until a positive acknowledgement has been received. This scheme is simpler, but it has the disadvantage of generating extra traffic and it suffers from the delay mentioned in the previous scheme.

3. Store and forward. Different packets are transmitted continuously. If no positive acknowledgement for certain packet has been received after an elapsed period , that packet would be retransmitted.

It is suggested that for highly noisy link (e.g. satellite links) scheme 2 should be used. However, for our network scheme 3 is to be used.

2.2. Retransmission Strategy

- As a packet moves through a node, the node stores the packet until a positive acknowledgement is returned from the succeeding node. This acknowledgement indicates that the packet was received without error and was accepted.

- Once a node has accepted a packet and returned a positive acknowledgement , it holds onto the packet until it in turn receives an acknowledgement from the succeeding node.

- If the packet is not accepted by the receiving node, no acknowledgement would be sent. This would be readily detected by the sending node from the absence of a returned acknowledgement within a reasonable time interval. The packet would be retransmitted There is a probability that the acknowledgement might be received just after the retransmission In this case the packet in a holding queue would be deleted.

- Acknowledgement themselves are not to be acknowledged

- Loss of acknowledgement results in the eventual retransmission of the packet.

- Duplicated copies would be sorted out by the host of the final destination.(i.e. there is a reassembly queue although we do not include this feature in our experiment).

2.3. Error Control on Link Failure

A link would be declared down if a number of immediate response request had been sent through the link at fixed intervals and yet no positive response has been received via the link.

The transmission of a immediate response request could be invoked under the following conditions:

1 Sustain absence on that link of either regular packets or acknowledgement.

2 The no. of times a packet has been retransmitted through the same link has exceeded some threshold.

3 No positive response has been received for an earlier immediate response request.

4 Receipt of request from other node to send immediate response request.

(see test case show1.a in section ? for detailed example)

In order to keep the link from overflowing with immediate response requests (this may happen if all of the above condition is satisfied at about the same time). A fixed amount of time has to be elapsed before the next request would be sent out.

2.3.1 Control After Link Failure is Detected

- Routing table of the node that detects the fault would be updated

- New routing information would be sent to the neighbor node. The neighboring node upon receiving the new routing information would update portion of its routing table.

- As each node has a record of node condition of all the nodes in the network (it is the number of operating links connected to that node) The record of the detecting node would be updated. There are three possible outcomes :

1 The detecting node discovered that it is disconnected. It would stop transmitting and receiving packets.

2 The other node is disconnected. In this case the "bad news" would be propagated to all the other nodes. The receiving nodes would stop transmitting and accepting packets for the node that is down.

3 The node condition is not zero. The node would send out control packets to all the nodes for them to update their records of node condition. The node (sender) would wait for control packets from other node informing it of the condition of the node on the other end of the link which is down. If no positive response is received after a prespecified period, the node would assume that the node at the other end of the link is down and would generated packets to announce the "bad news".

2.4. Error Control on Node Failure

A node would be declared down under the following conditions:

1 The record of the number of lines connected to the node has become zero. (see test case show2 in section 4 for a detailed example)

2 No positive response about the node has been received since the transmission of control packets requesting information about the node and that elapsed time has exceeded a threshold. (see test case SHOW1.C in section 4 for a detailed example)

2.4.1 Control After Node Failure is Detected

- routing table would be updated
- the packets of the node would be dropped
- no more new packets for the node would be generated
- the news would be propagated to all the other nodes via special control packets

A disconnection of a network into 2 separate networks is not assumed to happen here.

2.5. Control Messages for Error Control

The individual control messages used for error control described above are listed and described in the following.

The format for traces of these control packets is the same as trace for acknowledgement packets described in Part I. In short, it is

<time> GEN CTLPKT <packet> <source> <destination> <sender> <receiver>

<msg1> <msg2> <msg3> <msg4> <msg5> <msg6> <length>

There are six types of control messages that are concerned with error control. They are,

2.5.1 Immediate Response Request

It (type 23) is generated when the source node suspects that the link connecting it to the destination node is malfunctioning. The packet has to be acknowledged immediately by the receiving node. Absence of acknowledgement (see 2.5.2 below) would invoke retransmission. If the number of retransmission has exceeded certain prescribed limit, the line would be considered down and suitable procedure would be invoked.

It has a trace format of the following form:

<time> GEN CTLPACKET .....<receiver>

23 0 0 0 0 0 <length>

2.5.2 Acknowledgement for Immediate Response Request

It (type 24) is generated in response to an immediate response request (see 2.5.1) Its destination would be the source node of the immediate response request packet. It is similar to the general acknowledgement for data packets, except that it has information concerning the condition of the source node (i.e. the number of functioning link connected to the source node).

It has a trace format of the following form:

<time> GEN CTLPACKET ..... <receiver>

24 <packet number> <node condition> 0 0 0 <length>

where,

<packet number> is the packet number of the immediate response request packet

<node condition> is the number of operating links connected to the source node.

2.5.3 Node Information

This type of packet (type 25) would be invoked if a link is declared down by the source node. It informs its destination node the node condition of the source node and that the node at the other end of the link which is down has one less functioning link. It also acts as a request for the destination node to test that specific node if the destination node is a neighbor of that specific node.

It has a trace format of the following form:

<time> GEN CTLPACKET <source> <destination> <sender> 0

25 1 <node condition> <specific node> 0 0 <length>

where,

<node condition> is the number of operating links connected to the source node.

<specific node> is the node number on the other end of the link which is down.

2.5.4 Node Status Information

This packet (type 26) is generated to inform the destination node which earlier has sent a NODE INFORMATION (see 2.5.3 above) control packet, that a specific node is functioning.

It has a trace format of the following form:

<time> GEN CTLPACKET <source> <destination> <sender> 0

26 <specific node> 0 0 0 0 <length>

2.5.5 Node Down Information

This packet (type 27) informs the destination node that a specific node is down. It also contains the source node condition (i.e. the number of operating links connected to the source node).

It has a trace format of the following form:

<time> GEN CTLPACKET <source> <destination> <sender> 0

27 0 <node condition> <node number> 0 0 <length>

where, <node number> is the number of the node which is down

2.5.6 Routing Information

The packet (type 28) is used to transmit routing information of the source node to the destination node (which is a neighbor of source node). The delayvec of the source node is transmitted. Since it is possible for the destination node to examine the source node in the simulation, the packet is a dummy packet which initiates the destination node to copy the delayvec of the source node to the delaytable of the destination.

It has a trace format of the following form:

<time> GEN CTLPACKET <source> <destination> <sender> 0

28 0 0 0 0 0 <length>

2.6. Trace Formats for Error Control

2.6.1 Packet Rejection by Node and Local Host

The PKT RJCTD entries have the following form :

PKT RJCTD <pkt> <r.sn> <r.dn> <this node> <r.type> <r.length>

The PKT RJCTD entry describes the rejection of a packet when the destination node of the packet is not functioning.

The HST STP entries have the following form:

HST STP< > <r.sn> <r.dn> <r.type>

The HST STP entry describes the rejection of a packet generated from a local host from being put into the input queue. The rejection of the packet could be due to:

1 the node connecting to the local host is not functioning.

2 the holding queue of the node is full

The M.FULL entries have the following form:

M.FULL <pkt> <r.sn> <r.dn> <this node>

The M.FULL entry describes the rejection of a packet which were already in the input queue of the node due to filled holding queue The packet could be from the local host or line input.

The XPKT RJCTD entries have the following form:

XPKT RJCTD <pkt> <r.sn> <r.dn> <this node> <nxt node>

The XPKT RJCTD entry describes the rejection of a

packet from the holding queue because the number of retransmission of the packet has exceeded a prescribed limit.

2.6.2 Packets Rejection by Line Controller

The INDRP entries have the following form:

INDRP <pkt> <r.sn> <r.dn> <this node> <nxt node>

The INDRP entry describes the dropping of an incoming packet because of the filled input queue.

The DRP entries have the following form:

DRP PKT<pkt> <r.sn> <r.dn> <this node> <nxt node>

The DRP entry describes the dropping of an outgoing packet to simulate line failure.

2.6.3 Miscellaneous Traces

The I'M DISCON entries have the following form:

I'M DISCON <this node> <last connected node>

The I'M DISCON describes the time when a node discovers that it itself is disconnected.

The RETX PKT entries have the following form:

RETX PKT<pkt> <r.sn> <r.dn> <this node> <nxt node> <r.type>

The RETX PKT entry describes the retransmission of a packet.

2.7. Behavior of Networks with Component Failures

The limited storage capacity of a store-and-forward node makes it vulnerable to congestion if packets are allowed to enter the system faster then they are delivered to their destination. Failure to deliver packets at a rate as fast as they are received could be due to:

a Congestion at the destination node.

b Limited capacity of the output devices (lines).

c Simultaneous selection of the same route by different nodes. (see example in section 6)

Congestion usually occurs with component failure (node, line, etc.) Under our present distributed routing scheme, components failure could not be adjusted immediately and hence the scheme could not be effective and optimum for a long time. Routing loops which could degrade the system performance drastically could occur.

2.7.1 Description of Experiments

The following types of component failure have been studied and could be found in 4

line failure

Network studied:

a. A fully connected network of three nodes (a triangle). Test case SHOW1.A.

b. A straight line network of 3 nodes. Test case SHOW1.B.

c. A straight line network of 4 nodes with link failure in the middle. (disconnecting the network into two halves.) Test case SHOW1.C.

d. A two connected four node network (a square) with link failures disconnecting the network into two halves. Test case SHOW1.D.

node failure

Network studied:

A fully connected 3 node network (a triangle) with node failure.Test case show2.

2.7.2 Results

Results of simulation were condensed into formats as described in 3. However, only the results of interest to the above tests were attached. They are the data files, major traces of error detection control packets generated ,graphs and tables of traffic between nodes are attached in Chapter 4.

Depending on the flow of traffic, congestion may occur with any failure. However, in the tests performed, congestion occurred whenever failure happened even though the line utilization is only 10 to 25 %.

The protocol was sufficiently fast enough to detect failures within 625 ms after failure occurred (it depends on the parameters for error control that was selected. For example, Nopktinterval, Timeout). However, it is not intelligent enough to detect disconnection yet. Depending on the flow of traffic, congestion may occur with failure. However, in the tests performed congestion occurred whenever failure happened even though the line utilization was only 10 to 25 %. The adaptive routing scheme worked sufficiently well in some cases and was able to switch the routes before failures were discovered.

Although more storage would increase the mean time between congestion, more storage alone cannot, in general prevent its occurrence. However, it could be reasoned that large storage could possibly stamp out congestion due to small transient and thus preventing potential deadlocks. Assuming the time for discovering a failure is 250 ms and the channel utilization of a T1 carrier is 50 % (purely data packets - 7000 bits), it would need 250 ms * 7000 bits per frame/(10 ms per frame * 8 bits per byte) - 22kbytes (which is more than half of the storage for normal operation) to absorb the extra packets. Hence to prevent congestion due to component failure under heavy load condition, simply increasing the storage size is impractical. Congestion is inevitable if line or node failure does occur under heavy load condition. Prompt failure discovery is needed to reduce the level of congestion.

3. Collection of Experiment Results

The simulation results are condensed into two groups :-

1. Internode Traffic

The results tabulated are concerned with the traffic between two specific nodes.

2. Intranode Traffic

The results tabulated are concerned with the overall traffic of a specific node.

3.1. Internode Traffic

Internode traffic results are further subdivided into two categories:-

3.1.1 Density

The following tables are concerned with the internode traffic density.

TABLE 1A

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	0	588	589	0	0	0	0	0	587
1000	0	632	637	0	0	0	0	0	632
2000	0	577	578	0	0	0	0	0	577
3000	0	576	572	0	0	0	0	0	576
4000	0	599	603	0	0	0	0	0	599
5000	0	627	627	0	0	0	0	0	627
6000	0	625	621	0	0	0	0	0	625
7000	0	589	592	0	0	0	0	0	589
8000	0	616	616	0	0	0	0	0	616
9000	0	587	582	0	0	0	0	0	587

TABLE 1B

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	600	0	0	0	0	0	0	587	0
1000	630	0	0	0	0	0	0	633	0
2000	567	0	0	0	0	0	0	574	0
3000	577	0	0	0	0	0	0	568	0
4000	599	0	0	0	0	0	0	599	0
5000	611	0	0	0	0	0	0	623	0
6000	622	0	0	0	0	0	0	617	0
7000	594	0	0	0	0	0	0	589	0
8000	605	0	0	0	0	0	0	612	0

9000 586 0 0 0 0 0 0 578 0

The tables above have four major columns : TIME, TRAF (traffic), REJ (rejection) and END (arrival). Individually, they are,

a. TIME (time in millisecond since the start of simulation) The column (table 1A) contains the starting time of the interval in which the results were collected. For example, (table 1A) the results in the second row were collected between the interval of 1000 ms to 2000 ms.

b. HOST TRAF (HOST TRAFFIC)

The column (table 1B) contains the number of packets (type 3 or 4) generated by local host to the host of the other node. For example, in the 2nd column of the first row in table 1B, 600 packets were generated by host of node 3 to host of node 1 during the 1st 1000 ms.

c. LINE TRAF (LINE TRAFFIC)

The column contains the number of packets (all types) that were sent through the link connecting the two nodes. (in one direction only). For example, consider table 1A, in column 3 of row 1, 588 packets was sent from node 1 to node 2 through the link between node 1 to 2.

d. NODE TRAF (NODE TRAFFIC)

The column (table 1A) contains the number of control packets (type 2) generated by node 1 to node 2.

e. RET TRAF (RETRANSMISSION TRAFFIC)

The column (table 1A) contains the number of packets (all types) that were retransmitted from node 1 to node 2 via the link between node 1 and 2.

f. H+N REJ (REJECTION BY NODE and BY HOST OF THE SOURCE)

Each row of the column (table 1A) contains the number of new packets (all types) that would have been generated by the host of node 1 or node 1 if they were not rejected by the node (due to filled holding queue).

g. LINE REJ (REJECTION BY LINE CONTROLLER OF DESTINATION NODE)

The column (table 1A) contains the number of packets (all types) that would have been accepted by the line controller of the destination node (node 2) if its input queue was not full.

h. RET REJ (REJECTION OF RETRANSMITTED PACKETS)

The column (table 1A) contains the number of packets which had been retransmitted for several times and the number of retransmission had exceeded certain preset limit and hence discarded. (note: the packets concerned here are packets whose source is node 1 or host of node 1 and whose final destination is node 2)

i. HOST END (ARRIVAL OF PACKETS TO DESTINATION HOST)

The column (table 1B) contains the number of packets (type 3 and 4) which had arrived the destination host. Here the packets were generated by host of node 3 and the destination of the packets were node 1.

j. NODE END (ARRIVAL OF PACKETS TO DESTINATION NODE)

The column (table 1A) contains the number of type 2 packets which were generated by node 1 and had reached their final destination - node 2.

3.1.2 Delay

The delay tables considered here is the time between the generation of a packet and the time the packet being examined by its final destination. The delays were tabulated into two formats:

a. MEAN, VARIANCE and PACKET COUNT during different intervals (table 2)

TABLE 2
TYPE 3 PACKETS BETWEEN 3 TO 1

TIME	MEAN	VAR	COUNT
0	16.140	13.162	323
1000	15.685	13.633	354
2000	16.226	10.881	357
3000	16.311	13.201	373
4000	15.682	13.021	332
5000	16.186	12.120	353
6000	16.552	11.214	359
7000	16.254	11.943	351
8000	16.420	12.842	314
9000	16.501	12.429	326

The means and variances of transmission delay of different type of packet between their source and destination were tabulated. For example, from the table above for packets of type 3 generated by host of node 3 to host of node 1 has a mean delay of 16.311 ms and a variance of 13.201 ms during the interval of 3000 ms to 4000 ms. 373 packets arrived the destination during the interval.

b. HISTOGRAMS of TRANSMISSION DELAY (both actual count and % count)

TABLE 3A
histogram of type 3 from 3 to 1 (in packet count)
time created to time examined by the destination

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	19	245	59	0	0	0	0	0	0	0	0	323
1000	32	271	51	0	0	0	0	0	0	0	0	354
2000	18	291	48	0	0	0	0	0	0	0	0	357
3000	24	283	66	0	0	0	0	0	0	0	0	373
4000	25	258	49	0	0	0	0	0	0	0	0	332
5000	17	278	58	0	0	0	0	0	0	0	0	353
6000	13	285	61	0	0	0	0	0	0	0	0	359
7000	20	286	45	0	0	0	0	0	0	0	0	351
8000	13	235	66	0	0	0	0	0	0	0	0	314
9000	18	248	60	0	0	0	0	0	0	0	0	326
OVERALL	199	2680	563	0	0	0	0	0	0	0	0	3442

TABLE 3B
histogram of type 3 from 3 to 1 (in %)
time created to time examined by the destination

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	5.9	75.9	18.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1000	9.0	76.6	14.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2000	5.0	81.5	13.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3000	6.4	75.9	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

4000	7.5	77.7	14.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5000	4.8	78.8	16.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6000	3.6	79.4	17.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7000	5.7	81.5	12.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8000	4.1	74.8	21.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9000	5.5	76.1	18.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OVERALL	5.8	77.9	16.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Delay histograms of type 2, 3 and 4 were tabulated versus time (10 ms histogram interval). Different histograms are obtained for different intervals of the simulation as well as for the overall simulation interval. For example (table 3A), row 3 contains a histogram of delay during the interval 2000 ms - 3000 ms. 291 packets had a delay of 10 ms - 20 ms during this interval. The total number of packets of type 3 that arrived during the interval is 357. Correspondingly row 3 of table 3B shows that 81.5 % of packets arrived during the 10 ms to 20 ms time slot. The column " 100 up " contains the number of packets that experienced a delay of more than 100 ms.

3.2. Intranode Traffic

Similar to the above condensed node to node results the results are divided into two categories :-

3.2.1 Density

TABLE 4
node 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	LINE REJ	H+N REJ	%REJ H+N	%REJ LINE
0	600	590	3	0	0	0.0	0.0
1000	630	637	4	0	0	0.0	0.0
2000	567	577	4	0	0	0.0	0.0
3000	577	573	4	0	0	0.0	0.0
4000	599	602	4	0	0	0.0	0.0
5000	611	626	4	0	0	0.0	0.0
6000	622	623	4	0	0	0.0	0.0
7000	594	591	3	0	0	0.0	0.0
8000	605	616	4	0	0	0.0	0.0
9000	586	583	4	0	0	0.0	0.0

This type of table (the table above) is concerned with the total traffic in and out of an individual node and traffic rejected. The columns are explained as follows :

a. TIME

As before this column indicates the start of the interval in which the results are compiled. For example, result in row 2 of

the table was collected during the interval of 1000 ms to 2000 ms.

b. HOST TRAF (Host traffic)

Each row of the column contains the total number of packets (type 3 & 4) generated by host of node 3 and accepted by node 3.

c. LINE TRAF (Line traffic)

Each row of the column contains the total number of packets (all types) which were sent to node 3 and were accepted by node 3.

d. NODE TRAF (node traffic)

Each row of the column contains the total number of control packets (type 2) that were generated by node 3.

e. LINE REJ (line rejection)

Each row of the column contains the total number of packets sent from nodes connecting to 1 but were rejected by the line controller of node 1 (rejected because the input buffer was filled).

f. H+N REJ (Host and node rejection)

Each row of the column contains the total number of packets that would be generated by the host of node 3 or node 3 if the holding queue of node 3 was not full.

g. % REJ H+N (% of Host and Node rejection)

Each row of the column is the result of 100 * (H+N REJ)/(H+N REJ + HOST TRAF + NODE TRAF) where the variables are from the same row.

h. % REJ LINE (% of Line rejection)

Each row of the column is the result of 100 * (LINE REJ)/(LINE REJ + LINE TRAF) where the variables are from the same row.

3.2.2 Delay

This is concerned with the processing time of packets in a node (i.e. the time the packets is put into the input queue of the node to the time the packet is sent out of the node). It is a measure of queueing delay in the node. Again it is condensed into two formats very similar to those in node to node results.

a. The MEAN, VARIANCE and the total number of packets

TABLE 5
TYPE 3 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	5.508	5.756	310
1000	5.702	6.591	316
2000	5.614	6.576	274
3000	5.715	7.000	300
4000	5.482	5.790	291
5000	5.489	6.291	318
6000	5.437	6.814	318
7000	5.753	7.125	289
8000	5.854	7.227	288
9000	5.860	6.592	291

For each node two tables were tabulated for type 3 & 4 packets only because it was found that type 2 packets always have a delay less than a frametime.

b. Histogram (of queueing delay)

TABLE 6A
node 3 (Histogram of type 3 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	305	5	0	0	0	0	0	0	0	0	0	310
1000	297	19	0	0	0	0	0	0	0	0	0	316
2000	264	10	0	0	0	0	0	0	0	0	0	274
3000	289	11	0	0	0	0	0	0	0	0	0	300
4000	281	10	0	0	0	0	0	0	0	0	0	291
5000	311	7	0	0	0	0	0	0	0	0	0	318
6000	303	15	0	0	0	0	0	0	0	0	0	318
7000	277	12	0	0	0	0	0	0	0	0	0	289
8000	270	18	0	0	0	0	0	0	0	0	0	288
9000	280	11	0	0	0	0	0	0	0	0	0	291
TOTAL	2877	118	0	0	0	0	0	0	0	0	0	2995

TABLE 6B
CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	98.4	1.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1000	94.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2000	96.4	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3000	96.3	3.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4000	96.6	3.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5000	97.8	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6000	95.3	4.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7000	95.8	4.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8000	93.8	6.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9000	96.2	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TOTAL	96.1	3.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tables of histogram for type 3 and 4 packets were tabulated for each node. (table above). As for node to node results, histograms of packets count as well is % count are available. The histograms have a format similar to the histograms described for table 3A and 3B.

4. Error Control Experiments

The following is a series of experiment on error detection protocol. They could be found described in 2.7. The experiments done are:

a. A fully connected network of three nodes (a triangle). Test case SHOW1.A.

b. A straight line network of 3 nodes. Test case SHOW1.B.

c. A straight line network of 4 nodes with link failure in the middle. (disconnecting the network into two halves.) Test case SHOW1.C.

d. A two connected four node network (a square) with link failures disconnecting the network into two halves. Test case SHOW1.D.

e. A fully connected 3 node network (a triangle) with node failure.Test case show2.

4.1. Test Case SHOW1.A

The following is a data file used to demonstrate the error control procedure of the network when a link failure occurs. The network is a three nodes network which is fully connected as shown in figure 4.1. Link failure between 1 and 3 was scheduled at 600.0ms. (file SHOW1.A)

```
NEWNET 3
PROCNUM   1- 1  2- 1  3- 1
FRAMETIME 10
CONNECTIONS   1- 2
CONNECTIONS   1- 3
CONNECTIONS   2- 3
HOST 1  H 2:0.50
HOST 1  H 3:0.50
HOST 1    A: 0.10    D:0.50
HOST 2  H 1:0.50
HOST 2  H 3:0.50
HOST 2    A: 0.10    D:0.50
HOST 3  H 1:0.50
HOST 3  H 2:0.50
HOST 3    A: 0.10    D:0.50
TRACE DSK:TSHOW1.A
Fail Link  1- 3 600.0
WMOVEDELAY    0.00500
```

```
MAXHOLDQ    1 128000
MAXHOLDQ    2 128000
MAXHOLDQ    3 128000
MAXINFRAMEQ   1  56000
MAXINFRAMEQ   2  56000
MAXINFRAMEQ   3  56000
MODE  DATA E
x 2000
```

Results

The calculated time taken by a node to detect a link failure since a link is brought down is given by Nopktinterval + (Hellolimit + 2) * timeoutinterval. It gives 625ms in this test. As predicted the time at which the failure was detected was around 600ms + 625ms = 1225ms. The adaptive routing showed its effectiveness in our simulation. In fact, the routing for packets from node 1 to node 3 was switched to pass through node 2 before failure was detected. This could be seemed from the significant increase in line traffic from 1 to 2 during the interval of 1000ms to 1200ms (see accompanied figure 4.2 & 4.3). The adaptive routing was not fast enough to switch the route from node 3 to node 1 though. There was a large spike of traffic through node 2 during the interval when link failure was detected. This was because of the retransmission of packets from 1 to 3 via node 2 and vice versa. This potentially would lead to congestion if the traffic was higher. Node 3 changed its routing table as soon as link failure was detected and was able to prevent congestion which might happen if it had to wait for its next routing update time.

Traces of Major Control Packets for Failure Detection

To explain the operation of the error detection procedure, the trace of special control packets that were generated when a link failure is listed and explained below. (SHOW1.A)

note: Unimportant portions of control message had been truncated (message contents <msg1> <msg2>....<msg6>), since they are not necessary for the understanding of the protocol.

#	TIME	ACTION	PKT #	On	Dn	Sn	Rn	TYPE	
1.	710.896	GEN CTLPKT	462	3	1	3	1	23	
2.	720.052	GEN CTLPKT	465	1	3	1	3	23	
3.	840.096	RETX PKT	462	3	1	3	1		
4.	846.001	RETX PKT	465	1	3	1	3		
5.	966.782	RETX PKT	462	3	1	3	1		
6.	971.347	RETX PKT	465	1	3	1	3		
7.	1089.053	RETX PKT	462	3	1	3	1		
8.	1188.556	RETX PKT	465	1	3	1	3		
9.	1226.511	GEN CTLPKT	749	3	1	3	0	25	
10.	1226.511	GEN CTLPKT	750	3	2	3	0	25	
11.	1226.511	GEN CTLPKT	751	3	2	3	0	28	
12.	1226.511	XPKT RJCTD	462	3	1	3	1		
13.	1226.454	GEN CTLPKT	755	1	2	1	0	25	
14.	1226.454	GEN CTLPKT	756	1	3	1	0	25	
15.	1226.454	GEN CTLPKT	757	1	2	1	0	28	
16.	1226.454	XPKT RJCTD	465	1	3	1	3		
17.	1230.077	REC CTLPKT	755	1	2	1	2	25	
18.	1230.077	GEN CTLPKT	758	2	1	2	1	1	755
19.	1230.077	GEN CTLPKT	759	2	3	2	3	23	
20.	1230.323	REC CTLPKT	750	3	2	3	2	25	
21.	1230.323	GEN CTLPKT	762	2	3	2	3	1	750
22.	1240.129	REC CTLPKT	756	1	3	2	3	25	
23.	1240.129	GEN CTLPKT	775	3	2	3	2	1	756
24.	1240.286	REC CTLPKT	749	3	1	2	1	25	
25.	1240.286	GEN CTLPKT	776	1	2	1	2	1	749

Explanations:

line

1 Immediate response request control packet (# 462) generated by node (On) 3 to node (Dn) 1 as link 3 to 1 was suspected to be not operating.

2 Immediate response request control packet (# 465) generated by node (On) 1 to node (Dn) 3 as link 1 to 3 was suspected to be not operating.

3.-8 Retransmission of Immediate response request control packets after about 125 ms of last transmission.

9-10 Link of 3 to 1 was detected down by node (On) 3 after three retransmissions

of packet # 462.The news was propagated to node 1 and node 2 by generating packets # 749 and # 750 respectively. (Node information packets)

11 Updated new routing information was transmitted to the connected neighboring node of node 3 by packet # 751 (type 28, here to node 2 only, since link 3-1 was disconnected)

12 The Immediate response request packet # 462 dropped (since it was no longer needed)

13-16 Similar to 9-12

17 Packet # 755 (type 25) sent from node (On) 1 to node (Dn) 2 received and processed by node 2.

18 Control packet # 758 generated to node 1 to acknowledge arrival of packet # 755.

19 Node 2 upon receiving the control packet # 755 from node 1, decided to test if node 3 was actually down or not by sending out a immediate response request packet # 759 to node 3.

20-21 Since node 2 had received a control packet of type 25 from node 1 not too long ago, upon receiving control packet # 750 of type 25 from node 3, it would not send out a immediate response request packet to node 1 as it knew that node 1 was up.

22-23 Node 3 received the packet # 756 type 25 generated by node 1. Since it knew that the link was down, no action was taken. Acknowledgement (#775) for packet #756 generated.

24-25 Node 1 received the packet # 749 type 25 generated by node 3. Since it knew that the link was down, no action was taken. Acknowledgement (#776) for packet #749 generated.

The following are tables showing the node to node traffic for network described in file SHOW1.A.

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	22	10	0	0	0	0	13	9
200	10	25	18	0	0	0	0	9	16
400	10	26	12	0	0	0	0	11	15
600	6	16	10	0	0	0	0	6	10
800	7	14	8	0	0	0	0	6	8
1000	9	39	11	1	0	0	0	10	11
1200	11	70	43	0	0	0	0	11	43
1400	7	41	23	0	0	0	0	7	23
1600	12	46	22	0	0	0	0	11	21
1800	7	44	22	0	0	0	0	7	23

FROM NODE 1 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	23	7	0	0	0	0	16	7
200	13	25	12	0	0	0	0	13	12
400	11	16	5	0	0	0	0	11	5
600	11	0	2	8	0	0	0	0	0
800	18	0	0	26	0	0	0	0	0
1000	9	0	1	33	0	0	7	11	2
1200	9	0	1	6	0	0	2	18	1
1400	11	0	0	0	0	0	0	13	0
1600	14	0	0	0	0	0	0	14	0
1800	16	0	0	0	0	0	0	14	0

FROM NODE 2 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	21	13	0	0	0	0	10	11
200	17	29	9	0	0	0	0	18	11
400	12	23	11	0	0	0	0	12	11
600	9	16	8	0	0	0	0	8	8
800	8	14	6	0	0	0	0	8	6
1000	10	32	29	0	0	0	0	10	22
1200	12	74	30	0	0	0	0	12	34
1400	11	44	18	0	0	0	0	11	21
1600	12	47	27	0	0	0	0	12	27
1800	9	43	21	0	0	0	0	10	21

FROM NODE 2 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	29	16	0	0	0	0	13	16
200	3	11	10	0	0	0	0	3	8
400	7	19	12	0	0	0	0	7	12
600	9	26	16	0	0	0	0	9	17
800	10	17	6	0	0	0	0	10	7

1000	9	32	13	0	0	0	0	8	11
1200	5	67	42	0	0	0	0	4	44
1400	10	47	23	0	0	0	0	12	22
1600	14	47	18	0	0	0	0	14	19
1800	15	48	25	0	0	0	0	15	19

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	7	22	16	0	0	0	0	7	15
200	12	25	13	0	0	0	0	12	13
400	6	17	11	0	0	0	0	5	12
600	18	0	1	6	0	0	0	0	0
800	8	0	1	23	0	0	0	0	0
1000	10	0	0	22	0	0	6	0	0
1200	8	0	1	24	0	0	5	26	2
1400	13	3	0	0	0	0	0	12	0
1600	7	0	0	0	0	0	0	8	0
1800	17	0	0	0	0	0	0	12	0

FROM NODE 3 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	29	13	0	0	0	0	16	13
200	9	13	3	0	0	0	0	10	3
400	12	19	7	0	0	0	0	12	7
600	15	24	10	0	0	0	0	15	9
800	5	16	11	0	0	0	0	5	11
1000	14	35	21	0	0	0	0	13	22
1200	9	64	27	1	0	0	0	9	27
1400	10	48	25	0	0	0	0	11	25
1600	8	44	30	0	0	0	0	8	28
1800	8	55	29	0	0	0	0	8	30

4.2. Test Case SHOW1.B

The following is a data file used to demonstrate the error control protocol and to study the behavior of a simple network with failure. The network is a straight line network of three nodes. Link failure which disconnects an end node is scheduled to happen at 600.0ms. The network is shown in figure 4.4. (file : SHOW1.B)

```
newnet 3
procnum 1-1 2-1 3-1
frametime 10
connection 1-2
connection 2-3
host 1 h2: .50 h3: .50 a: .10 d: .50
host 2 h1: .50 h3: .50 a: .10 d: .50
host 3 h1: .50 h2: .50 a: .10 d: .50
mode date error
trace dsk: tshow1.b
fail node 3  600.0
maxinframeq 1 28000
maxinframeq 2 56000
maxinframeq 3 28000
maxholdq    1 64000
maxholdq    2 64000
maxholdq    3 64000
nopktinterval 125
x 2000
```

Results

As expected the test of link failure started about 125ms after the link between 2 and 3 was brought down (600ms) i.e. about 725ms. After three retransmissions of the control packet (type 23 - immediate response request) i.e. after 125 * (2 + 2) ms, the node was decided to be down (725 + 500 = 1225ms) and the information was passed on to all the other nodes in the network. (see figure 4.5 and 4.6)

Node 1 was informed of the failure of node 3 shortly after 1200ms of the simulation and hence it stopped generating packets for node 3. Similarly node 2 stopped packets generation for node 3 during the interval of 1200-1400ms. Node 3 discovering that it was disconnected also stopped all host traffic.

Traces of Major Control Packets for Failure Detection

To explain the operation of the error detection procedure, the trace of special control packets that were generated when failure occurred were listed. (file SHOW1.B)

(note:for explanation of the traces see 3)

TIME	ACTION	>PKT	ON	DN	SN	RN	RESSRCES 1 (TYPE)	2	3	4	5	6	length
721.295	GEN CTLPKT	519	3	2	3	2	23	0	0	0	0	0	100
721.920	GEN CTLPKT	521	2	3	2	3	23	0	0	0	0	0	100
848.053	RETX PKT	521	2	3	2	3	2						
850.865	RETX PKT	519	3	2	3	2	2						
977.333	RETX PKT	519	3	2	3	2	2						
980.865	RETX PKT	521	2	3	2	3	2						
1102.936	RETX PKT	519	3	2	3	2	2						
1109.649	RETX PKT	521	2	3	2	3	2						
1229.693	I'M DISCON	3	2										
1235.181	GEN CTLPKT	632	2	1	2	0	27	0	1	3	0	0	100
1240.096	REC CTLPKT	632	2	1	2	1	27	0	1	3	0	0	

Brief summary:

Link failure was suspected at round 721 ms, control packets of type 25 (immediate response request) were sent out. Node 3 discovered that it was disconnected at 1229.69 ms where as node 3 was discovered down at around 1235.2 ms by node 2. Control packets of type 27 (node down information) was then sent out by node 2. The following are tables showing the node to node traffic for network described in file SHOW1.B

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	45	17	0	0	0	0	13	16
200	10	49	30	0	0	0	0	9	27
400	10	43	17	0	0	0	0	11	21
600	6	29	11	0	0	0	0	6	11
800	13	29	5	0	0	0	0	3	5
1000	5	44	4	34	1	41	0	0	4
1200	10	29	15	13	3	8	0	27	14
1400	12	16	3	0	0	0	0	12	4
1600	9	15	8	0	0	0	0	9	6
1800	13	25	11	0	0	0	0	13	12

FROM NODE 1 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	0	0	0	0	0	0	15	0
200	13	0	0	0	0	0	0	13	0
400	11	0	0	0	0	0	0	12	0
600	11	0	0	0	0	0	0	0	0
800	12	0	0	0	0	0	0	0	0
1000	9	0	0	0	0	0	0	0	0
1200	0	0	0	0	0	0	23	0	0

1400	0	0	0	0	0	0	0	0	0
1600	0	0	0	0	0	0	0	0	0
1800	0	0	0	0	0	0	0	0	0

FROM NODE 2 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	43	29	0	0	0	0	18	26
200	17	54	22	0	0	0	0	18	24
400	12	40	22	0	0	0	0	12	23
600	9	28	20	0	0	0	6	8	19
800	4	14	8	0	5	0	0	5	9
1000	0	4	5	1	3	0	0	0	4
1200	11	53	40	2	0	0	0	11	41
1400	3	14	12	0	0	0	0	3	11
1600	6	17	11	0	0	0	0	6	11
1800	12	24	13	0	0	0	0	11	13

FROM NODE 2 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	51	23	0	0	0	0	13	23
200	3	36	22	0	0	0	0	3	20
400	7	36	16	0	0	0	0	7	17
600	9	0	2	12	0	0	0	0	0
800	5	0	0	39	4	0	0	0	0
1000	0	0	0	53	14	0	0	0	0
1200	9	0	0	5	4	0	14	0	0
1400	0	0	0	0	0	0	0	0	0
1600	0	0	0	0	0	0	0	0	0
1800	0	0	0	0	0	0	0	0	0

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	7	0	0	0	0	0	0	7	0
200	12	0	0	0	0	0	0	12	0
400	6	0	0	0	0	0	0	5	0
600	10	0	0	0	0	0	0	1	0
800	2	0	0	0	5	0	0	0	0
1000	0	0	0	0	16	0	0	0	0
1200	0	0	0	0	9	0	10	0	0
1400	0	0	0	0	12	0	3	0	0
1600	0	0	0	0	9	0	0	0	0
1800	0	0	0	0	8	0	0	0	0

FROM NODE 3 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	51	28	0	0	0	6	15	28
200	9	38	16	0	0	0	0	10	16
400	12	37	19	0	0	0	0	12	19
600	15	0	1	15	0	0	0	0	0
800	1	0	1	47	10	0	0	0	0

1000	0	0	0	43	10	0	0	0	0
1200	0	0	0	20	12	0	12	0	0
1400	0	0	0	2	12	0	6	0	0
1600	0	0	0	1	11	0	0	0	0
1800	0	0	0	2	11	0	0	0	0

4.3. Test Case SHOW1.C

The following is a data file used to demonstrate the error control protocol. The network used is a four nodes straight line network as in figure 4.7. A link failure between node 2 and 3 was scheduled at 600.0ms. The link failure disconnected the network into two halves. Since the present protocol would not be able to detect the disconnection, traffic for nodes on the other side of the disconnected network would continue and looped around the network.

```
newnet 4
procnum 1-1 2-1 3-1 4-1
frametime 10
connection 1-2
connection 2-3
connection 3-4
host 1 h2: .34 h3: .33 h4: .33 a:.10 d:.50
host 2 h3: .34 h4: .33 h1: .33 a:.10 d:.50
host 3 h4: .34 h1: .33 h2: .33 a:.10 d:.50
host 4 h1: .34 h2: .33 h3: .33 a:.10 d:.50
mode data error
trace dsk: tshow1.c
fail link 2-3 600ms
maxinframeq 1 28000
maxinframeq 2 56000
maxinframeq 3 56000
maxinframeq 4 28000
maxholdq    1 64000
maxholdq    2 64000
maxholdq    3 64000
maxholdq    4 64000
nopktinterval 125
x 3000
```

Results

The results are quite similar to SHOW1.C Link failure between 3 and 2 was detected at around 1225ms. (node information packets - type 25 were sent out). Node 3 was declared down after waiting for 400ms of prescribed time limit for hearing no positive response for node information packets. However, the network was not able to detected the disconnection of the network. For example, node 1 continued to generate packets for node 4 (see figure 4.8 & 4.9) and the packets looped between node 2 and node 1. (as can be seemed from the attached tables for heavy line traffic between 1 and 2)

Traces of Major Control Packets for Failure Detection

To explain the operation of the error detection procedure, the trace of special control packets that were generated when failure occurred were listed below.

							MESSAGES						
TIME	ACTION	PKT	ON	DN	SN	RN	1 (TYPE)	2	3	4	5	6	length
719.067	GEN CTLPKT	748	2	3	2	3	23	0	0	0	0	0	100
720.096	GEN CTLPKT	749	3	2	3	2	23	0	0	0	0	0	100
845.161	RETX PKT	748	2	3	2	3	2						
845.246	RETX PKT	749	3	2	3	2	2						
971.295	RETX PKT	748	2	3	2	3	2						
980.065	RETX PKT	749	3	2	3	2	2						
1100.096	RETX PKT	748	2	3	2	3	2						
1110.065	RETX PKT	749	3	2	3	2	2						
1225.181	GEN CTLPKT	919	2	1	2	0	25	0	1	3	0	0	100
1225.181	GEN CTLPKT	920	2	3	2	0	25	0	1	3	0	0	100
1225.181	GEN CTLPKT	921	2	4	2	0	25	0	1	3	0	0	100
1235.181	GEN CTLPKT	932	3	1	3	0	25	0	1	2	0	0	100
1235.181	GEN CTLPKT	933	3	2	3	0	25	0	1	2	0	0	100
1235.181	GEN CTLPKT	934	3	4	3	0	25	0	1	2	0	0	100
1625.829	GEN CTLPKT	1817	2	1	2	0	27	0	1	3	0	0	100
1625.829	GEN CTLPKT	1818	2	4	2	0	27	0	1	3	0	0	100
1635.829	GEN CTLPKT	1836	3	1	3	0	27	0	1	2	0	0	100
1635.829	GEN CTLPKT	1837	3	4	3	0	27	0	1	2	0	0	100

Brief Summary:

Failure was suspected at around 720ms. Control packets immediate response request (type 25) were sent out. Link failure was declared 500ms later. Node information packets were sent out to test if node 2 and node 3 was down by node 3

and node 2 respectively. After waiting for 400ms, hearing no response for the node information packets, node 2 declared that node 3 is down and node down information packets (type 27) were sent out. Node 3 decided that node 2 was down and dispatched the node down information packets. The following are tables showing the node to node traffic for network described in file SHOW1.C.

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	10	67	35	0	0	0	0	10	33
300	11	58	27	0	0	0	0	11	29
600	10	42	18	0	0	0	0	7	10
900	1	48	3	42	2	48	0	0	3
1200	4	279	146	40	2	45	0	7	137
1500	5	350	175	31	6	29	0	5	175
1800	10	443	226	21	4	7	1	8	220
2100	6	382	185	27	4	29	0	7	195
2400	1	115	56	68	12	61	0	0	59
2700	8	255	122	41	8	39	0	9	125

FROM NODE 1 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	15	0	0	0	0	0	0	15	0
300	8	0	0	0	0	0	0	8	0
600	14	0	0	0	0	0	0	0	0
900	8	0	0	0	5	0	0	0	0
1200	2	0	0	0	5	0	3	0	0
1500	2	0	0	0	4	0	17	0	0
1800	0	0	0	0	2	0	0	0	0
2100	0	0	0	0	10	0	0	0	0
2400	0	0	0	0	10	0	0	0	0
2700	0	0	0	0	7	0	0	0	0

FROM NODE 1 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	0	0	0	0	0	0	8	0
300	10	0	0	0	0	0	0	10	0
600	10	0	0	0	0	0	0	0	0
900	10	0	0	0	1	0	0	0	0
1200	2	0	0	0	8	0	2	0	0
1500	5	0	0	0	5	0	0	0	0
1800	0	0	0	0	4	0	1	0	0
2100	2	0	0	0	7	0	0	0	0
2400	1	0	0	0	10	0	0	0	0
2700	2	0	0	0	11	0	10	0	0

FROM NODE 2 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	67	34	8	0	0	0	17	32
300	10	57	29	0	0	0	0	9	30
600	5	33	24	0	2	0	0	6	25
900	0	5	4	1	10	0	0	0	4
1200	3	303	149	9	8	21	0	1	149
1500	5	235	179	52	4	36	0	4	173
1800	5	458	225	31	5	1	0	4	231
2100	3	379	188	17	4	24	0	7	188
2400	0	110	57	71	6	76	0	0	57
2700	2	243	134	44	12	43	0	2	125

FROM NODE 2 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	16	86	38	0	0	0	0	14	37
300	14	83	41	0	0	0	0	16	39
600	4	1	2	26	3	0	0	0	0
900	0	0	0	80	13	0	0	0	0
1200	2	0	2	26	12	0	2	0	0
1500	1	0	0	0	10	0	5	0	0
1800	0	0	0	0	10	0	0	0	0
2100	0	0	0	0	6	0	0	0	0
2400	0	0	0	0	10	0	0	0	0
2700	0	0	0	0	8	0	0	0	0

FROM NODE 2 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	12	0	0	0	0	0	0	12	0
300	10	0	0	0	0	0	0	9	0
600	9	0	0	0	2	0	0	1	0
900	0	0	0	0	11	0	0	0	0
1200	4	0	1	0	9	0	7	0	0
1500	7	0	1	0	5	0	0	0	0
1800	2	0	0	0	11	0	1	0	0
2100	9	0	0	0	5	0	0	0	0
2400	1	0	0	0	13	0	0	0	0
2700	4	0	0	0	5	0	6	0	0

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	0	0	0	0	0	0	9	0
300	10	0	0	0	0	0	0	9	0
600	5	0	0	0	0	0	0	1	0
900	0	0	0	0	10	0	0	0	0
1200	4	0	1	0	9	0	3	0	0
1500	2	0	1	0	6	0	0	0	0
1800	11	0	0	0	7	0	0	0	0
2100	3	0	0	0	10	0	0	0	0
2400	0	0	0	0	5	0	3	0	0
2700	1	0	0	0	6	0	0	0	0

FROM NODE 3 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	85	49	0	0	0	0	11	47
300	10	85	44	0	0	0	0	10	44
600	5	1	2	26	0	0	0	0	0
900	0	0	0	75	10	0	0	0	0
1200	9	0	1	26	3	0	3	0	0
1500	0	0	0	0	6	0	7	0	0
1800	0	0	0	0	1	0	0	0	0
2100	0	0	0	0	5	0	0	0	0
2400	0	0	0	0	8	0	8	0	0
2700	0	0	0	0	6	0	8	0	0

FROM NODE 3 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	60	32	0	0	0	0	9	31
300	11	57	26	0	0	0	0	11	27
600	8	38	29	0	2	0	0	8	29
900	0	3	3	0	8	0	0	0	3
1200	0	373	196	19	7	17	0	0	186
1500	7	385	193	20	3	24	0	5	197
1800	6	470	225	22	6	0	0	7	231
2100	4	218	112	41	8	54	0	3	112
2400	2	126	61	63	14	66	0	1	61
2700	3	240	126	40	10	42	2	2	126

FROM NODE 4 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	0	0	0	0	0	0	9	0
300	9	0	0	0	0	0	0	9	0
600	16	0	0	0	0	0	0	1	0
900	8	0	0	0	0	0	0	0	0
1200	2	0	0	0	5	0	6	0	0
1500	3	0	0	0	5	0	3	0	0
1800	5	0	0	0	6	0	3	0	0
2100	1	0	0	0	10	0	0	0	0
2400	0	0	0	0	15	0	4	0	0
2700	2	0	0	0	7	0	8	0	0

FROM NODE 4 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	0	0	0	0	0	0	9	0
300	12	0	0	0	0	0	0	11	0
600	4	0	0	0	0	0	0	0	0
900	5	0	0	0	0	0	0	0	0
1200	3	0	0	0	6	0	2	0	0
1500	0	0	0	0	5	0	5	0	0
1800	0	0	0	0	4	0	0	0	0
2100	0	0	0	0	8	0	0	0	0
2400	0	0	0	0	7	0	0	0	0

2700	0	0	0	0	4	0	0	0	0

FROM NODE 4 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	14	58	29	0	0	0	0	14	26
300	6	59	30	0	0	0	0	5	33
600	10	42	11	0	0	0	0	9	11
900	7	52	2	25	0	32	0	0	1
1200	7	367	178	34	3	23	0	14	179
1500	6	374	285	30	7	24	1	5	195
1800	4	464	230	38	4	6	1	4	230
2100	1	222	107	50	4	59	0	5	113
2400	2	127	66	54	9	62	0	0	66
2700	2	232	120	50	7	41	0	4	116

4.4. Test Case SHOW1.D

The following is a data file used to demonstrate the error control protocol. The network used is a four nodes network as shown in figure 4.10 Failures were scheduled so that the network would be disconnected into two halves. Although our present protocol is not implemented to recognize disconnection (in which more than one node which are connected together being disconnected from the rest of the network), this is a special case in which disconnection was recognized. The link between 2 and 3 and the link between 1 and 4 were scheduled to be brought down at 600.0ms. (file SHOW1.D)

```
newnet 4
procnum 1-1  2-1 3-1 4-1
connection 1-2
connection 2-3
connection 3-4
connection 4-1
frametime 10
host 1 h2: .34 h3: .33 h4: .33 a:.10 d:.50
host 2 h3: .34 h4: .33 h1: .33 a:.10 d:.50
host 3 h4: .34 h1: .33 h2: .33 a:.10 d:.50
host 4 h1: .34 h2: .33 h3: .33 a:.10 d:.50
mode data error
trace dsk: tshow1.d
fail link 2-3 600ms
fail link 4-1 600ms
```

```
maxinframeq 1 56000
maxinframeq 2 56000
maxinframeq 3 56000
maxinframeq 4 56000
maxholdq    1 64000
maxholdq    2 64000
maxholdq    3 64000
maxholdq    4 64000
nopktinterval 125
x 3000
```

Results

This is a special case of disconnection in which link failure occurs at the same time. The network was able to detect the disconnection because control packets (type 25) had been sent out to test all the nodes in the network. (see graphs for termination of traffic for node on the opposite side of the disconnected network) If no positive response was received for the control packets, nodes would be declared down. As in previous cases link failures were detected at around 1225 ms. Node information packets (type 25) were sent out. After 400 ms (the default prescribed time limit for receiving response for type 25 control packets), having receive no positive response for the control packets, node 1 decided that node 4 was down and node 2 decided that node 3 was down and vice versa and they sent out control packet (node down information) to their neighbors.

Traces of Major Control Packets for Failure Detection

To explain the operation of the error detection procedure, the trace of special control packets that were generated when failure occurred were listed. (file SHOW1.D)

TIME	ACTION	#PKT	ON	DN	SN	RN	MESSAGES 1 (TYPE)	2	3	4	5	6	length
710.096	GEN CTLPKT	664	1	4	1	4	23	0	0	0	0	0	100
720.065	GEN CTLPKT	671	4	1	4	1	23	0	0	0	0	0	100
721.920	GEN CTLPKT	675	2	3	2	3	23	0	0	0	0	0	100
720.096	GEN CTLPKT	672	3	2	3	2	23	0	0	0	0	0	100
836.730	RETX PKT	664	1	4	1	4	2						
845.246	RETX PKT	672	3	2	3	2	2						
850.065	RETX PKT	675	2	3	2	3	2						
851.295	RETX PKT	671	4	1	4	1	2						
971.360	RETX PKT	672	3	2	3	2	2						
980.065	RETX PKT	675	2	3	2	3	2						
980.065	RETX PKT	678	2	3	2	3	3						
981.295	RETX PKT	671	4	1	4	1	2						
1088.063	RETX PKT	664	1	4	1	4	2						
1109.008	RETX PKT	672	3	2	3	2	2						
1110.065	RETX PKT	671	4	1	4	1	2						
1110.096	RETX PKT	675	2	3	2	3	2						
1213.464	GEN CTLPKT	847	1	2	1	0	25	0	1	4	0	0	100
1213.464	GEN CTLPKT	848	1	3	1	0	25	0	1	4	0	0	100
1213.464	GEN CTLPKT	849	1	4	1	0	25	0	1	4	0	0	100
1220.096	REC CTLPKT	847	1	2	1	2	25	0	1	4	0	0	
1234.080	GEN CTLPKT	857	3	1	3	0	25	0	1	2	0	0	100
1234.080	GEN CTLPKT	858	3	2	3	0	25	0	1	2	0	0	100
1234.080	GEN CTLPKT	859	3	4	3	0	25	0	1	2	0	0	100
1240.096	GEN CTLPKT	863	4	1	4	0	25	0	1	1	0	0	100
1240.096	GEN CTLPKT	864	4	2	4	0	25	0	1	1	0	0	100
1240.096	GEN CTLPKT	865	4	3	4	0	25	0	1	1	0	0	100
1240.226	REC CTLPKT	859	3	4	3	4	25	0	1	2	0	0	
1244.121	GEN CTLPKT	871	2	1	2	0	25	0	1	3	0	0	100
1244.121	GEN CTLPKT	872	2	3	2	0	25	0	1	3	0	0	100
1244.121	GEN CTLPKT	873	2	4	2	0	25	0	1	3	0	0	100
1250.096	REC CTLPKT	871	2	1	2	1	25	0	1	3	0	0	
1250.290	REC CTLPKT	865	4	3	4	3	25	0	1	1	0	0	
1613.886	GEN CTLPKT	1952	1	2	1	0	27	0	1	4	0	0	100
1613.886	GEN CTLPKT	1953	1	3	1	0	27	0	1	4	0	0	100
1621.971	REC CTLPKT	1952	1	2	1	2	27	0	1	4	0	0	
1635.183	GEN CTLPKT	2016	3	1	3	0	27	0	1	2	0	0	100
1635.183	GEN CTLPKT	2017	3	4	3	0	27	0	1	2	0	0	100
1640.098	GEN CTLPKT	2023	4	2	4	0	27	0	1	1	0	0	100
1640.098	GEN CTLPKT	2024	4	3	4	0	27	0	1	1	0	0	100
1640.163	REC CTLPKT	2017	3	4	3	4	27	0	1	2	0	0	
1641.781	REC CTLPKT	2024	4	3	4	3	27	0	1	1	0	0	
1645.190	GEN CTLPKT	2058	2	1	2	0	27	0	1	3	0	0	100
1651.317	REC CTLPKT	2058	2	1	2	1	27	0	1	3	0	0	

Brief summary:

Link failures were suspected at around 720ms, control packets of type 23 (immediate response request) were sent out. Link failure was detected 500 ms later

and node information (type 25 packets) were sent out. After waiting for 400 ms (prescribed time limit for receiving positive response for type 25 packets), i.e. at about 1625ms, nodes were decided to be down and packets containing node down information (type 27) were sent out since no positive response was received. The following are tables showing the node to node traffic for network described in file SHOW1.D

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	10	72	38	0	0	0	0	10	38
300	11	57	28	0	0	0	0	11	27
600	8	36	13	0	0	0	0	8	14
900	2	58	3	8	6	5	0	0	3
1200	2	333	181	23	3	41	0	2	187
1500	6	263	131	25	2	24	0	8	145
1800	12	19	8	0	0	0	0	11	8
2100	11	20	9	0	0	0	0	11	9
2400	18	32	14	0	0	0	0	18	14
2700	10	16	6	0	0	0	0	10	6

FROM NODE 1 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	15	0	0	0	0	0	0	15	8
300	8	0	0	0	0	0	0	8	0
600	14	0	0	0	0	0	0	0	0
900	6	0	0	0	2	0	0	0	0
1200	4	0	1	0	10	0	4	0	0
1500	0	0	1	0	4	0	7	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 1 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	39	18	0	0	0	0	9	18
300	10	40	21	0	0	0	0	10	21
600	11	0	2	17	0	0	0	0	0
900	7	0	1	49	6	0	0	0	0
1200	4	0	1	20	7	0	2	0	0
1500	0	0	0	0	3	0	14	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 2 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET. TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	70	34	0	0	0	0	17	32
300	10	60	30	0	0	0	0	9	32
600	5	35	24	0	0	0	0	6	24
900	2	5	5	0	7	0	0	1	4
1200	3	389	172	17	3	0	0	2	173
1500	3	241	130	19	4	31	0	5	129
1800	9	20	11	0	0	0	0	8	12
2100	7	20	13	0	0	0	0	7	13
2400	11	31	20	0	0	0	0	12	19
2700	6	17	10	0	0	0	0	6	11

FROM NODE 2 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	16	49	20	0	0	0	0	14	20
300	14	42	20	0	0	0	0	16	18
600	7	0	2	16	0	0	0	0	0
900	1	0	1	74	8	0	0	0	0
1200	4	0	1	32	8	0	2	0	0
1500	6	0	0	0	3	0	6	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 2 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	12	0	0	0	0	0	0	12	0
300	10	0	0	0	0	0	0	9	0
600	10	0	0	0	0	0	0	0	0
900	1	0	0	0	5	0	0	0	0
1200	1	0	1	0	9	0	7	0	0
1500	1	0	0	0	2	0	5	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	0	0	0	0	0	0	9	0
300	10	0	0	0	0	0	0	9	0
600	6	0	0	0	0	0	0	1	0
900	11	0	0	0	4	0	0	0	0
1200	0	0	1	0	2	0	3	0	0
1500	0	0	1	0	2	0	5	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 3 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	47	29	0	0	0	0	11	27
300	10	45	24	0	0	0	0	10	25
600	6	1	2	14	0	0	0	0	0
900	5	0	0	52	5	0	0	0	0
1200	3	0	1	31	6	0	3	0	0
1500	1	0	0	0	5	0	2	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 3 TO NODE 4

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	22	14	0	0	0	0	9	13
300	11	17	5	0	0	0	0	11	6
600	9	18	12	0	0	0	0	8	10
900	3	10	4	1	3	0	0	3	0
1200	3	426	192	16	8	0	0	3	192
1500	9	256	141	13	1	31	0	10	141
1800	11	20	8	0	0	0	0	12	8
2100	9	25	17	0	0	0	0	8	17
2400	13	25	11	0	0	0	0	14	11
2700	9	20	11	0	0	0	0	9	11

FROM NODE 4 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	36	21	0	0	0	0	9	18
300	9	42	19	0	0	0	0	10	21
600	14	0	2	14	0	0	0	0	0
900	6	0	0	63	5	0	0	0	0
1200	4	0	1	35	4	0	5	0	0
1500	1	0	0	0	2	0	5	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0
2700	0	0	0	0	0	0	0	0	0

FROM NODE 4 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	9	0	0	0	0	0	0	9	0
300	12	0	0	0	0	0	0	11	0
600	6	0	0	0	0	0	0	0	0
900	3	0	0	0	8	0	0	0	0
1200	0	0	1	0	5	0	3	0	0
1500	0	0	1	0	3	0	1	0	0
1800	0	0	0	0	0	0	0	0	0
2100	0	0	0	0	0	0	0	0	0
2400	0	0	0	0	0	0	0	0	0

2700	0	0	0	0	0	0	0	0	0

FROM NODE 4 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	14	22	9	0	0	0	0	14	8
300	6	17	11	0	0	0	0	5	12
600	10	20	10	9	0	0	0	10	10
900	2	8	5	0	8	0	0	3	5
1200	4	402	282	13	5	0	0	1	187
1500	3	283	126	12	5	13	0	6	140
1800	8	21	12	0	0	0	0	8	13
2100	15	25	10	0	0	0	0	15	10
2400	10	25	16	0	0	0	0	9	16
2700	10	20	9	0	0	0	0	11	9

4.5. Test Case SHOW2

The following is a data file used to demonstrate the error control procedure of the network when a node failure occurs. The network is a three nodes network which is fully connected as shown in figure 4.5. Node 3 was failure scheduled at 600.0ms. (file SHOW2)

```
NEWNET 3
PROCNUM   1- 1  2- 1  3- 1
FRAMETIME 10
CONNECTIONS  1- 2
CONNECTIONS  1- 3
CONNECTIONS  2- 3
HOST 1  H 2:0.50
HOST 1  H 3:0.50
HOST 1    A: 0.10   D:0.50
HOST 2  H 1:0.50
HOST 2  H 3:0.50
HOST 2    A: 0.10   D:0.50
HOST 3  H 1:0.50
HOST 3  H 2:0.50
HOST 3    A: 0.10   D:0.50
TRACE DSK:tshow2
Fail Node  3  600.0
NOPKTINTERVAL 125
MAXHOLDQ    1 128000
MAXHOLDQ    2 128000
MAXHOLDQ    3 128000
MAXINFRAMEQ   1  56000
MAXINFRAMEQ   2  56000
```

MAXINFRAMEQ 3 56000
MODE DATA ERROR

Results

The network used is the same as the previous test (SHOW1.A) and since initially node failure is similar to link failure, the traffic was quite similar too. As expected from the calculation from previous example, link failure between 2 to 3 and 1 to 3 were discovered at about 1225ms and 1245ms respectively. For node 3, it discovered its disconnection at 1235ms. Node failure of node 3 was discovered after passing Node Information control packets between node 2 and 1. As in the previous test, the adaptive routing algorithm had changed the route of packets from node 1 to node 3 to pass through node 2 before link or node failure was detected. This could be seen from the increase in line traffic between 1 and 2 starting from the interval 1000ms to 1200ms. (see accompanied graphs).

Traces of Major Control Packets For Failure Detection

To explain the operation of the error detection procedure, the trace of special control packets that were generated when failure occurred were listed below. (file SHOW2)

note: Unimportant portions of control message had been truncated (message contents <msg1> <msg2>....<msg6>), since they are not necessary for the understanding of the protocol.

#	TIME	ACTION	PKT	On	Dn	Sn	Rn	TYPE
1.	718.212	GEN CTLPKT	447	2	3	2	3	23
2.	720.052	GEN CTLPKT	448	3	1	3	1	23
3.	720.052	GEN CTLPKT	449	3	2	3	2	23
4.	730.052	GEN CTLPKT	451	1	3	1	3	23
5.	848.053	RETX PKT	447	2	3	2	3	
6.	850.052	RETX PKT	448	3	1	3	1	
7.	850.052	RETX PKT	449	3	2	3	2	
8.	860.052	RETX PKT	451	1	3	1	3	
9.	980.180	RETX PKT	447	2	3	2	3	
10.	980.460	RETX PKT	448	3	1	3	1	

```
11.      980.468 RETX PKT    443  3  2  3  2
12.     1107.715 RETX PKT    447  2  3  2  3
13.     1110.052 RETX PKT    448  3  1  3  1
14.     1110.052 RETX PKT    449  3  2  3  2
15.     1120.052 RETX PKT    451  1  3  1  3
16.     1233.216 GEN CTLPKT  672  2  1  2  0  25
17.     1233.216 GEN CTLPKT  673  2  3  2  0  25
18.     1233.216 GEN CTLPKT  674  2  1  2  0  28
19.     1233.216 XPKT RJCTD  447  2  3  2  3
20.     1235.181 GEN CTLPKT  678  3  1  3  0  25
21.     1235.181 GEN CTLPKT  679  3  2  3  0  25
22.     1235.181 GEN CTLPKT  680  3  2  3  0  28
23.     1235.181 XPKT RJCTD  448  3  1  3  1
24.     1235.181 I'M DISCON  3  2
25.     1235.181 XPKT RJCTD  449  3  2  3  2
26.     1240.286 REC CTLPKT  672  2  1  2  1  25
27.     1240.286 GEN CTLPKT  681  1  2  1  2  1    672
28.     1245.181 GEN CTLPKT  686  1  2  1  0  27
29.     1245.181 GEN CTLPKT  687  1  2  1  0  28
30.     1245.181 XPKT RJCTD  451  1  3  1  3
31.     1245.181 XPKT RJCTD  499  1  3  1  3
32.     1250.452 REC CTLPKT  686  1  2  1  2  27
33.     1258.452 GEN CTLPKT  698  2  1  2  1  1    686
34.     1260.142 PKT RJCTD   673  2  3  1
```

Explanations:

LINE

1 Immediate response request control packet (# 447) generated by node 2 to node 3 as link 2 to 3 was suspected to be not operating.

2 Immediate response request control packet (# 448) generated by node 3 to node 1 as link 3 to 1 was suspected to be not operating.

3 Immediate response request control packet (# 449) generated by node 3 to node 2 as link 3 to 2 was suspected to be not operating.

4 Immediate response request control packet (# 451) generated by node 1 to node 3 as link 1 to 3 was suspected to be not operating.

5-15 Retransmission of immediate response request control packets after about 125ms of last transmission.

16-19 Link 2 to 3 was detected down by node 2 after three retransmissions of the

immediate response request packet #447. Control packets of type 25 were sent to all the nodes informing them that a link was down. The routing table was updated and the new routing information was sent to its neighboring node (# 674 to node 1 here). The immediate response request packet #477 was dropped since the link was determined down already.

20-23 Link 3 to 1 was detected down by node 3 after three retransmissions of the immediate response request packet #488. Control packets of type 25 were sent to all the nodes informing them that a link was down. The routing table was updated and the new routing information was sent to its neighboring node (node 2 here). The immediate response request packet #448 was dropped since the link was determined down already.

24-25 Link 3 to 2 was determined down by node 3. However, node 3 found that it was then totally disconnected, so it issued a trace "TM DISCON" to the trace. The immediate response request packet (# 449) was dropped since it had lost its use.

26-27 Node 1 received the control packet from 2 that informed it that link between 2 and 3 was down. It generated an acknowledgement (# 681) for that packet (# 672).

28-31 Node 1 determined that link 1 to 3 was down when it examined the no. of times the immediate response request (# 451) had been retransmitted. Since it had a record of the connectivity of node 3, it determined that now node 3 was totally disconnected or down. It then sent out control packet of type 27 (# 686) to inform node 2 that node 3 was down. It updated its routing table and sent out its new routing information. It also started dumping the packets destined for node 3 (for example #499 dumped). Immediate response request packet (# 451) dropped too.

32-34 Node 2 processed the control packet (type 27) which proclaimed that node 3

was down. It acknowledged it and started dumping packets destined for node 3 (e.g. #673). The following are tables showing the node to node traffic for network described in file show2.

FROM NODE 1 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	22	10	0	0	0	0	13	9
200	10	25	18	0	0	0	0	9	16
400	10	26	12	0	0	0	0	11	15
600	6	16	10	0	0	0	0	6	10
800	7	14	8	0	0	0	0	6	8
1000	9	39	11	1	0	0	0	10	11
1200	17	45	25	0	0	0	0	17	23
1400	8	18	9	0	0	0	0	7	11
1600	11	29	17	0	0	0	0	12	17
1800	8	18	13	0	0	0	0	7	11

FROM NODE 1 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	23	7	0	0	0	0	16	7
200	13	25	12	0	0	0	0	13	12
400	11	16	5	0	0	0	0	11	5
600	11	0	2	8	0	0	0	0	0
800	10	0	0	26	0	0	0	0	0
1000	9	0	1	33	0	0	7	0	0
1200	2	0	0	3	0	0	26	0	0
1400	0	0	0	0	0	0	0	0	0
1600	0	0	0	0	0	0	0	0	0
1800	0	0	0	0	0	0	0	0	0

FROM NODE 2 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	11	21	13	0	0	0	0	10	11
200	17	29	9	0	0	0	0	18	11
400	12	23	11	0	0	0	0	12	11
600	9	16	8	0	0	0	0	8	8
800	8	14	6	0	0	0	0	8	6
1000	10	32	29	0	0	0	0	10	22
1200	12	52	26	3	0	0	0	12	31
1400	8	17	7	0	0	0	0	9	8
1600	15	29	14	0	0	0	0	15	14
1800	13	21	7	0	0	0	0	13	8

FROM NODE 2 TO NODE 3

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	13	29	16	0	0	0	0	13	16
200	3	11	10	0	0	0	0	3	8

400	7	19	12	0	0	0	0	7	12
600	9	0	2	5	0	0	0	0	0
800	10	0	0	19	0	0	0	0	0
1000	9	0	0	30	0	0	6	0	0
1200	1	0	1	10	0	0	19	0	0
1400	0	0	0	0	0	0	0	0	0
1600	0	0	0	0	0	0	0	0	0
1800	0	0	0	0	0	0	0	0	0

FROM NODE 3 TO NODE 1

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	7	22	16	0	0	0	0	7	15
200	12	25	13	0	0	0	0	12	13
400	6	17	11	0	0	0	0	5	12
600	10	0	1	6	0	0	0	0	0
800	8	0	1	23	0	0	0	0	0
1000	10	0	0	22	0	0	6	0	0
1200	4	0	1	27	9	0	6	0	0
1400	0	0	0	0	6	0	12	0	0
1600	0	0	0	3	8	0	9	0	0
1800	0	0	0	3	11	0	0	0	0

FROM NODE 3 TO NODE 2

TIME (ms)	HOST TRAF	LINE TRAF	NODE TRAF	RET TRAF	H+N REJ	LINE REJ	RET REJ	HOST END	NODE END
0	17	29	13	0	0	0	0	16	13
200	9	13	3	0	6	0	0	10	3
400	12	19	7	0	0	0	0	12	7
600	15	0	1	9	0	0	0	0	0
800	5	0	1	26	0	0	0	0	0
1000	14	0	0	32	0	0	9	0	0
1200	1	0	2	35	9	0	10	0	0
1400	0	0	0	45	8	0	5	0	0
1600	0	0	1	8	9	0	13	0	0
1800	0	0	0	5	19	0	0	0	0

5. Experiments on Congestion

Two series of test were performed to study the behavior of a simple network under heavy load condition. The near congestion behavior and the level the network (figure 5.1 & 5.4) has a major driver i.e. a source node with high rate of packet generation (in this experiment node 3 is the source node).

One of the series of experiment was run with node 3 having an unlimited amount of storage space and the other series of experiment was run with node 3 having a limited storage space.

1 Congestion with Unlimited Storage

The network experimented with is as shown in figure 5.1 Node 3 of the network was allowed to have a storage space of 640000 bits to simulate unlimited storage space. Node 2 was assigned with only one processor so that congestion could occur at its input queue. Two different series of experiment were run. As could be seen from figure 5.2 and figure 5.3 the results agree quite well. Congestion started to occurred between the data packet rate of .7 packets/ ms to .8 packets/ms (the rate is the packet generation rate of node 3). Processing delay at the center node (congested node) increased drastically when congested. In fact, when looking at the histograms attached, during congestion almost all type 4 packets have a delay of more than 100 ms. The mean delay time fluctuates but it is an indication of the level of congestion if taken over a long period of say 10 seconds ,as higher traffic level shows a larger delay time.

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packet generation rate was about 6 packets/10 ms. Congestion did not occur and the delay is minimal.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	128	127	25	1	0	0	0	0	0	0	0	281
1000	163	122	32	2	0	0	0	0	0	0	0	319
2000	170	119	7	0	0	0	0	0	0	0	0	296
3000	140	109	22	0	0	0	0	0	0	0	0	271
4000	136	157	15	0	0	0	0	0	0	0	0	308
5000	132	129	32	8	0	0	0	0	0	0	0	301
6000	90	165	42	3	0	0	0	0	0	0	0	300
7000	135	165	5	0	0	0	0	0	0	0	0	305
8000	124	178	16	0	0	0	0	0	0	0	0	318
9000	130	138	24	0	0	0	0	0	0	0	0	292
TOTAL	1348	1409	220	14	0	0	0	0	0	0	0	2991

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	45.6	45.2	8.9	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1000	51.1	38.2	10.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2000	57.4	40.2	2.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3000	51.7	40.2	8.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4000	44.2	51.0	4.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5000	43.9	42.9	10.6	2.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6000	30.0	55.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7000	44.3	54.1	1.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8000	39.0	56.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9000	44.5	47.3	8.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TOTAL	45.1	47.1	7.4	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	12.144	24.398	281
1000	11.793	29.703	319
2000	10.279	10.929	296
3000	11.141	20.509	271
4000	11.280	15.955	308
5000	12.874	38.182	301
6000	12.642	30.950	300
7000	10.577	12.759	305
8000	11.488	18.373	318
9000	11.683	22.459	292

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packet generation rate was about 7 packets/10 ms. Congestion did not occur and the mean delay is not high. However, sign of near congestion was shown. The histogram shows a wider spread of delay distribution than the histogram for packet generation rate of .6 packet/ms.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	94	149	89	10	0	0	0	0	0	0	0	342
1000	86	176	74	3	0	0	0	0	0	0	0	339
2000	87	114	104	48	4	0	0	0	0	0	0	357
3000	77	173	80	11	2	0	0	0	0	0	0	343
4000	114	125	38	40	22	7	1	0	0	0	0	347
5000	95	140	74	21	8	0	0	0	0	0	0	338
6000	17	85	81	56	34	43	38	33	4	0	0	391
7000	79	103	68	29	12	11	0	0	0	0	0	302
8000	93	133	62	60	17	7	0	0	0	0	0	372
9000	95	211	53	4	0	0	0	0	0	0	0	363
TOTAL	837	1409	723	282	99	68	39	33	4	0	0	3494

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	27.5	43.6	26.0	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1000	25.4	51.9	21.8	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2000	24.4	31.9	29.1	13.4	1.1	0.0	0.0	0.0	0.0	0.0	0.0
3000	22.4	50.4	23.3	3.2	0.6	0.0	0.0	0.0	0.0	0.0	0.0
4000	32.9	36.0	11.0	11.5	6.3	2.0	0.3	0.0	0.0	0.0	0.0
5000	28.1	41.4	21.9	6.2	2.4	0.0	0.0	0.0	0.0	0.0	0.0
6000	4.3	21.7	20.7	14.3	8.7	11.0	9.7	8.4	1.0	0.0	0.0
7000	26.2	34.1	22.5	9.6	4.0	3.6	0.0	0.0	0.0	0.0	0.0
8000	25.0	35.8	16.7	16.1	4.6	1.9	0.0	0.0	0.0	0.0	0.0
9000	26.2	58.1	14.6	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TOTAL	24.0	40.3	20.7	8.1	2.8	1.9	1.1	0.9	0.1	0.0	0.0

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	15.371	60.004	342
1000	14.676	45.447	339
2000	19.106	103.353	357
3000	15.295	48.379	343
4000	18.355	165.143	347
5000	16.574	92.241	338
6000	37.830	437.109	391
7000	19.045	150.383	302
8000	19.620	159.819	372
9000	13.474	36.759	363

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packet generation rate was about 7 packets/10 ms. No congestion occurred but sign of near congestion was shown in the histograms below as the distribution of type 4 packet delay spread.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	84	93	71	33	28	20	4	0	0	0	0	333
1000	52	172	98	31	5	0	0	0	0	0	0	358
2000	107	160	48	17	7	0	0	0	0	0	0	339
3000	89	165	64	14	1	0	0	0	0	0	0	333
4000	101	136	34	16	26	4	2	9	5	0	0	333
5000	26	63	71	73	56	33	52	6	0	0	0	380
6000	124	175	49	0	0	0	0	0	0	0	0	348
7000	66	139	69	27	23	30	12	0	0	0	0	366
8000	96	144	64	39	10	0	0	0	0	0	0	353
9000	116	136	72	16	2	0	0	0	0	0	0	342
TOTAL	861	1383	640	266	158	87	70	15	5	0	0	3485

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	25.2	27.9	21.3	9.9	8.4	6.0	1.2	0.0	0.0	0.0	0.0
1000	14.5	48.0	27.4	8.7	1.4	0.0	0.0	0.0	0.0	0.0	0.0
2000	31.6	47.2	14.2	5.0	2.1	0.0	0.0	0.0	0.0	0.0	0.0
3000	26.7	49.5	19.2	4.2	0.3	0.0	0.0	0.0	0.0	0.0	0.0
4000	30.3	40.8	10.2	4.8	7.8	1.2	0.6	2.7	1.5	0.0	0.0
5000	6.8	16.6	18.7	19.2	14.7	8.7	13.7	1.6	0.0	0.0	0.0
6000	35.6	50.3	14.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7000	18.0	38.0	18.9	7.4	6.3	8.2	3.3	0.0	0.0	0.0	0.0
8000	27.2	40.8	18.1	11.0	2.8	0.0	0.0	0.0	0.0	0.0	0.0
9000	33.9	39.8	21.1	4.7	0.6	0.0	0.0	0.0	0.0	0.0	0.0
TOTAL	24.7	39.7	18.4	7.6	4.5	2.5	2.0	0.4	0.1	0.0	0.0

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	22.488	214.488	333
1000	18.504	83.257	358
2000	14.622	68.434	339
3000	15.608	59.185	333
4000	20.136	300.373	333
5000	36.266	329.587	380
6000	12.788	28.204	348
7000	23.301	255.066	366
8000	17.395	99.117	353
9000	14.938	67.770	342

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packet generation rate was about 8 packets/10 ms. Congestion occurred and 33 % of type 4 packets experienced delay larger than 100 ms.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	9	53	57	68	72	38	37	10	14	0	0	366
1000	24	0	0	0	0	0	8	45	51	65	158	351
2000	0	0	0	0	0	0	2	51	66	138	108	365
3000	0	0	0	0	0	0	0	6	77	115	76	274
4000	0	0	0	0	0	0	0	7	37	78	71	193
5000	0	0	0	0	0	0	0	0	17	56	142	215
6000	0	0	0	0	0	0	0	7	27	82	88	204
7000	0	0	0	0	0	0	0	2	17	41	140	200
8000	0	0	0	0	0	0	0	4	41	94	55	194
9000	0	0	0	0	0	0	6	30	52	81	30	199
TOTAL	33	53	57	68	72	38	53	170	399	750	858	2561

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	2.5	14.5	15.6	18.6	19.7	10.4	10.1	4.9	3.8	0.0	0.0
1000	6.8	0.0	0.0	0.0	0.0	0.0	2.3	12.8	14.5	18.5	45.0
2000	0.0	0.0	0.0	0.0	0.0	0.0	0.5	14.0	18.1	37.8	29.6
3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2	28.1	42.0	27.7
4000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.6	19.2	40.4	36.8
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.9	26.0	66.0
6000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.4	13.2	40.2	43.1
7000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	8.5	20.5	70.0
8000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	21.1	48.5	28.4
9000	0.0	0.0	0.0	0.0	0.0	0.0	3.0	15.1	26.1	40.7	15.1
TOTAL	1.3	2.1	2.2	2.7	2.8	1.5	2.1	6.6	15.6	29.3	33.9

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	41.805	403.750	366
1000	92.890	805.317	351
2000	94.523	130.163	365
3000	95.607	65.499	274
4000	98.261	114.987	193
5000	103.475	86.077	215
6000	98.935	107.644	204
7000	104.667	114.513	200
8000	96.139	66.312	194
9000	90.716	111.597	199

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packet generation rate was about 8 packets/10 ms. Congestion occurred and 92 % of type packets experienced delay of more than 100 ms in this experiment.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	20	7	5	3	9	1	9	3	36	62	168	323
1000	0	0	0	0	0	0	0	0	0	9	271	280
2000	0	0	0	0	0	0	0	0	0	0	217	217
3000	0	0	0	0	0	0	0	0	0	0	188	188
4000	0	0	0	0	0	0	0	0	0	0	180	180
5000	0	0	0	0	0	0	0	0	0	0	206	206
6000	0	0	0	0	0	0	0	0	0	0	201	201
7000	0	0	0	0	0	0	0	0	0	6	196	202
8000	0	0	0	0	0	0	0	0	0	1	180	181
9000	0	0	0	0	0	0	0	0	0	0	179	179
TOTAL	20	7	5	3	9	1	9	3	36	78	1986	2157

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	6.2	2.2	1.5	0.9	2.8	0.3	2.8	0.9	11.1	19.2	52.0
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.2	96.8
2000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
4000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
6000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
7000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	97.0
8000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	99.4
9000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
TOTAL	0.9	0.3	0.2	0.1	0.4	0.0	0.4	0.1	1.7	3.6	92.1

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	92.128	1080.802	323
1000	124.745	119.296	280
2000	127.998	82.372	217
3000	128.778	58.573	188
4000	140.869	152.968	180
5000	128.663	38.500	206
6000	120.831	68.486	201
7000	128.239	217.787	202
8000	121.484	99.672	181
9000	137.447	120.140	179

The following tables are the processing delay of type 4 packets in node 2 (the center node). The packets generation rate was about 9 packets/10 ms. Congestion occurred and 100 % of type 4 packets experience delay greater than 100 ms during the later period of simulation.

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	23	18	3	4	1	8	11	24	0	4	224	320
1000	0	0	0	0	0	0	0	0	0	0	186	186
2000	0	0	0	0	0	0	0	0	0	0	205	205
3000	0	0	0	0	0	0	0	0	0	0	145	145
4000	0	0	0	0	0	0	0	0	0	0	144	144
5000	0	0	0	0	0	0	0	0	0	0	140	140
6000	0	0	0	0	0	0	0	0	0	0	82	82
7000	0	0	0	0	0	0	0	0	0	0	195	195
8000	0	0	0	0	0	0	0	0	0	0	8	8
9000	0	0	0	0	0	0	0	0	0	0	155	155
TOTAL	23	18	3	4	1	8	11	24	0	4	1484	1580

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	7.2	5.6	0.9	1.3	0.3	2.5	3.4	7.5	0.0	1.3	70.0
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
2000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
4000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
6000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
7000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
8000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
9000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
TOTAL	1.5	1.1	0.2	0.3	0.1	0.5	0.7	1.5	0.0	0.3	93.9

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	98.065	1769.517	320
1000	162.891	223.252	186
2000	144.823	131.531	205
3000	130.455	36.657	145
4000	144.114	297.199	144
5000	141.777	252.546	140
6000	142.700	136.123	82
7000	145.606	102.036	195
8000	182.322	54.509	8
9000	166.761	186.239	155

The following tables are the processing delay of type 4 packets in node (the center node). The packet generation rate was about 9 packets/10 ms (data file :- dsd30.q9)

node 2 (Histogram of type 4 packet counts)
time put into input queue to time sent out by line

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up	total
0	16	5	3	3	5	4	0	2	6	11	212	267
1000	0	0	0	0	0	0	0	0	0	0	189	189
2000	0	0	0	0	0	0	0	0	0	0	139	139
3000	0	0	0	0	0	0	0	0	0	0	135	135
4000	0	0	0	0	0	0	0	0	0	0	96	96
5000	0	0	0	0	0	0	0	0	0	0	216	216
6000	0	0	0	0	0	0	0	0	0	0	144	144
7000	0	0	0	0	0	0	0	0	0	0	137	137
8000	0	0	0	0	0	0	0	0	0	0	175	175
9000	0	0	0	0	0	0	0	0	0	0	64	64
TOTAL	16	5	3	3	5	4	0	2	6	11	1507	1562

CORRESPONDING % HISTOGRAM

time (ms)	0 - 10	10- 20	20- 30	30- 40	40- 50	50- 60	60- 70	70- 80	80- 90	90- 100	100 up
0	6.0	1.9	1.1	1.1	1.9	1.5	0.0	0.7	2.2	4.1	79.4
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
2000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
3000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
4000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
6000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
7000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
8000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
9000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
TOTAL	1.0	0.3	0.2	0.2	0.3	0.3	0.0	0.1	0.4	0.7	96.5

TYPE 4 PACKET (time put into node to time sent out of node)

TIME	MEAN	VAR	COUNT
0	113.822	1724.700	267
1000	148.152	115.336	189
2000	135.393	91.176	139
3000	149.294	157.021	135
4000	131.349	72.709	96
5000	127.975	65.906	216
6000	124.978	58.584	144
7000	149.342	95.553	137
8000	132.879	231.457	175
9000	140.854	145.315	64

5.1. Congestion with Limited Storage

The network used is as shown in figure 5.3. Node 3 was assigned a more realistic level of storage. Some background traffic was generated by node 1 and 2 in this case (total of about .2 packets per ms). Hence, congestion occurred when node 3 had a packet generation rate of about .5 to .6 packets per ms, as node 2 is able to handle about 7 packets per 10ms.

The behavior of the network on the threshold of congestion was studied. It was found that (Figure 5.5 & 5.6) packets delay fluctuated at the threshold of congestion. For some short period the delay was about the delay of no congestion, Average delay for a short period of a few minutes would not be useful in determining the level of congestion, but with longer period the delay may be useful in formulating a flow control decision. The % rejection could also be a good measure of congestion.

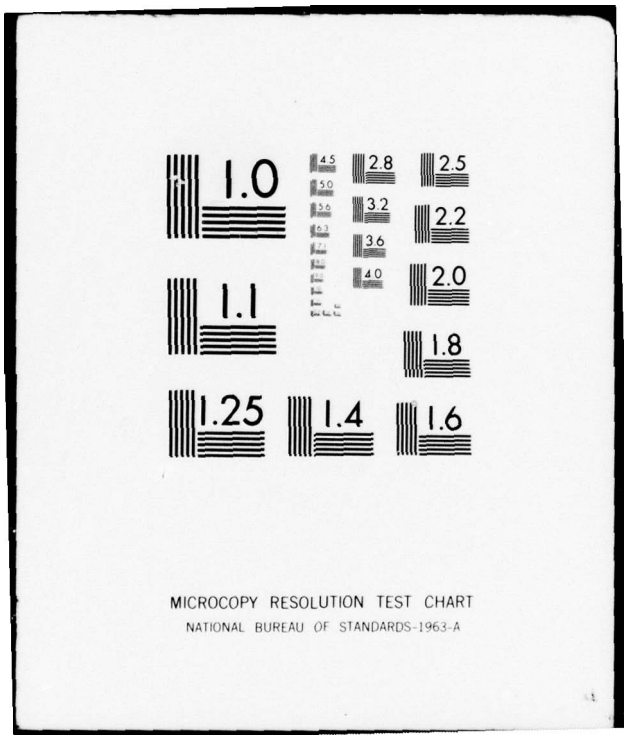
AD-A040 584 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 17/2
THE SIMULATION OF AN INTEGRATED VOICE/DATA COMMUNICATIONS NETWO--ETC(U)
MAY 77 M R BARBACCI DCA100-76-C-0058

UNCLASSIFIED DCA-100-76-C-0058 NL

3 OF 3
AD A040584

END

DATE FILMED
7-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

6. Routing Inefficiency

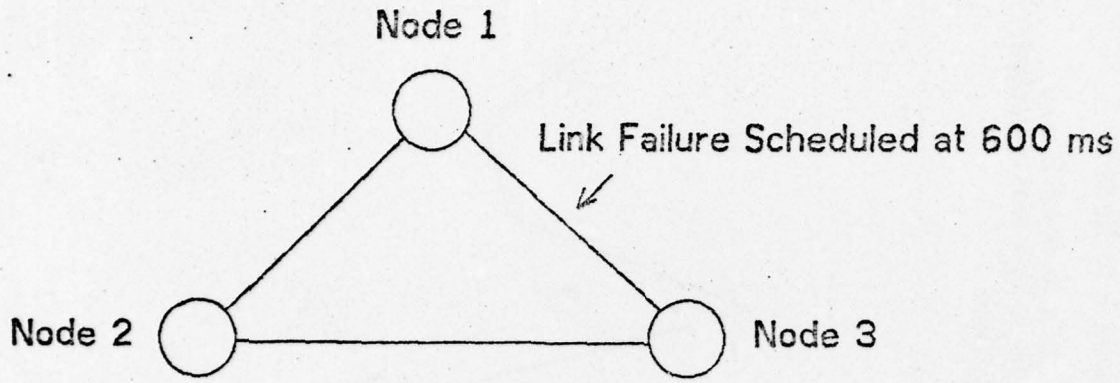
A Sample Simulation to Demonstrate the Inefficiency of the Present Routing Scheme

A simple network was used to demonstrate one of the inefficiencies of the present scheme. The network configuration is as shown in figure 6.1. interesting aspect of the traffic flow could be found on figure 6.2 & figure 6.3.

In this example node 3 and node 5 generate most of the traffic to node 1. Node 2 and node 4 acts as intermediate nodes. Here due to heavy traffic, if node 3 and 5 send their packets via the same route, eventually congestion would occur. Adaptive routing was fast enough to switch their routes to prevent congestion but it was not intelligent enough to switch one of the route only. Hence, the near congestion condition oscillates between node 2 and node 4. The traffic delay was low when the two nodes were not using the same route but increased when they used the same route.

For example, initially the two nodes (5 & 3) chose node # 2 as the prime intermediate node and hence packet transmission delay from source to destination (5 or 3 to node 1) was around 40 ms. The routing algorithm was able to switch the routes fast enough during the 500 to 2500 ms interval to keep the delay low. However, during the interval of 3000 ms to 3500 ms, node # 4 was heavily used as the prime route by both nodes (while 2 had a relatively low traffic) a spike increase in delay (up to 64 ms) resulted. Similarly during the period of 4000 ms to 4500 ms, route 2 was too heavily used and it produced a spike in mean delay. A better strategy would keep the two nodes from using the same route and thus preventing unnecessary delay.

Appendix A: Figures



This experiment is to show the error detection protocol and the network behavior with failure. Congestion occurred and traffic delay increased.

Figure 4.1 Error Control Experiments - SHOW1.A
Network Configuration

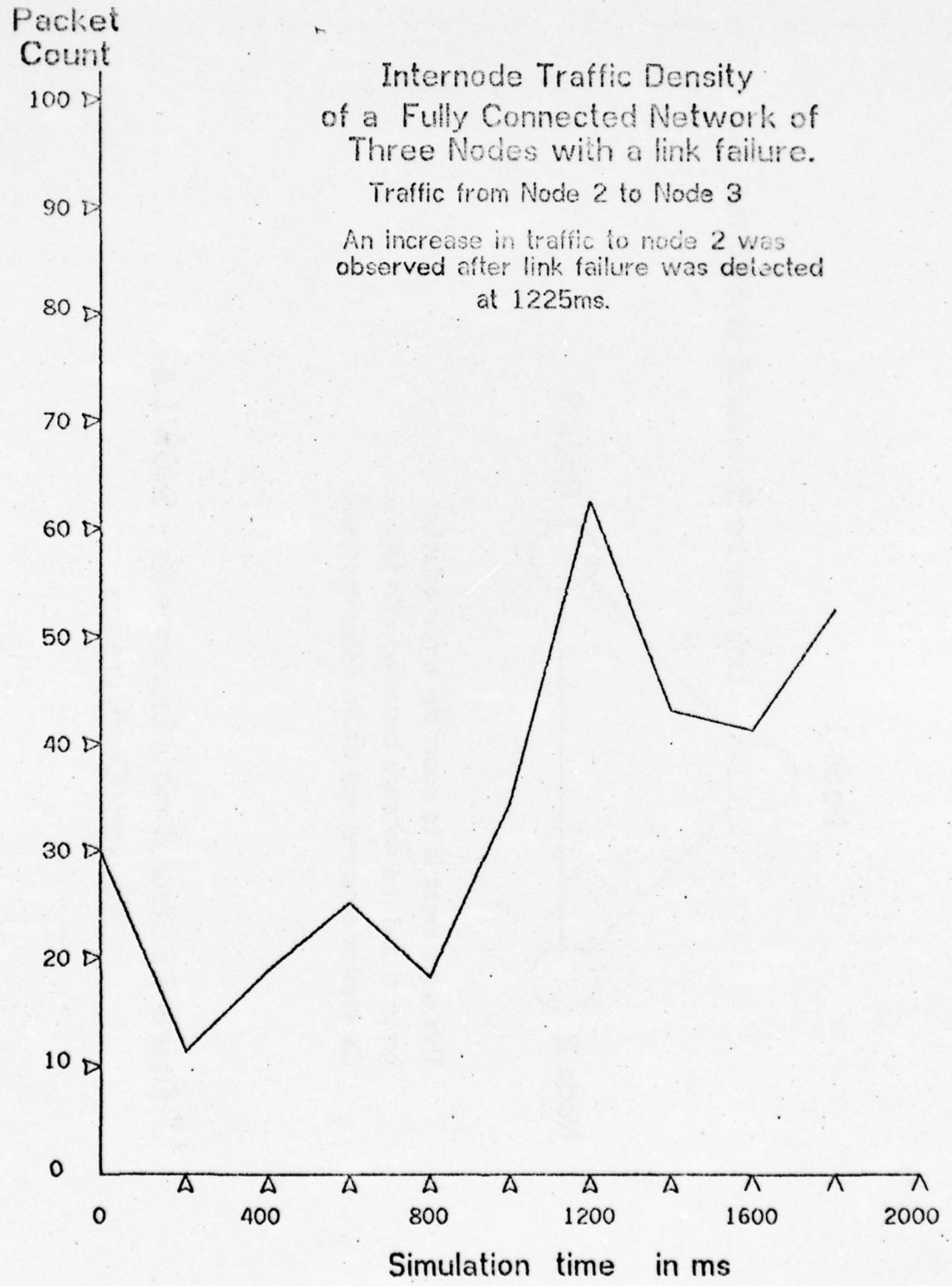


Figure 4.2 Test Case SHOW1.A - Internode Traffic Density

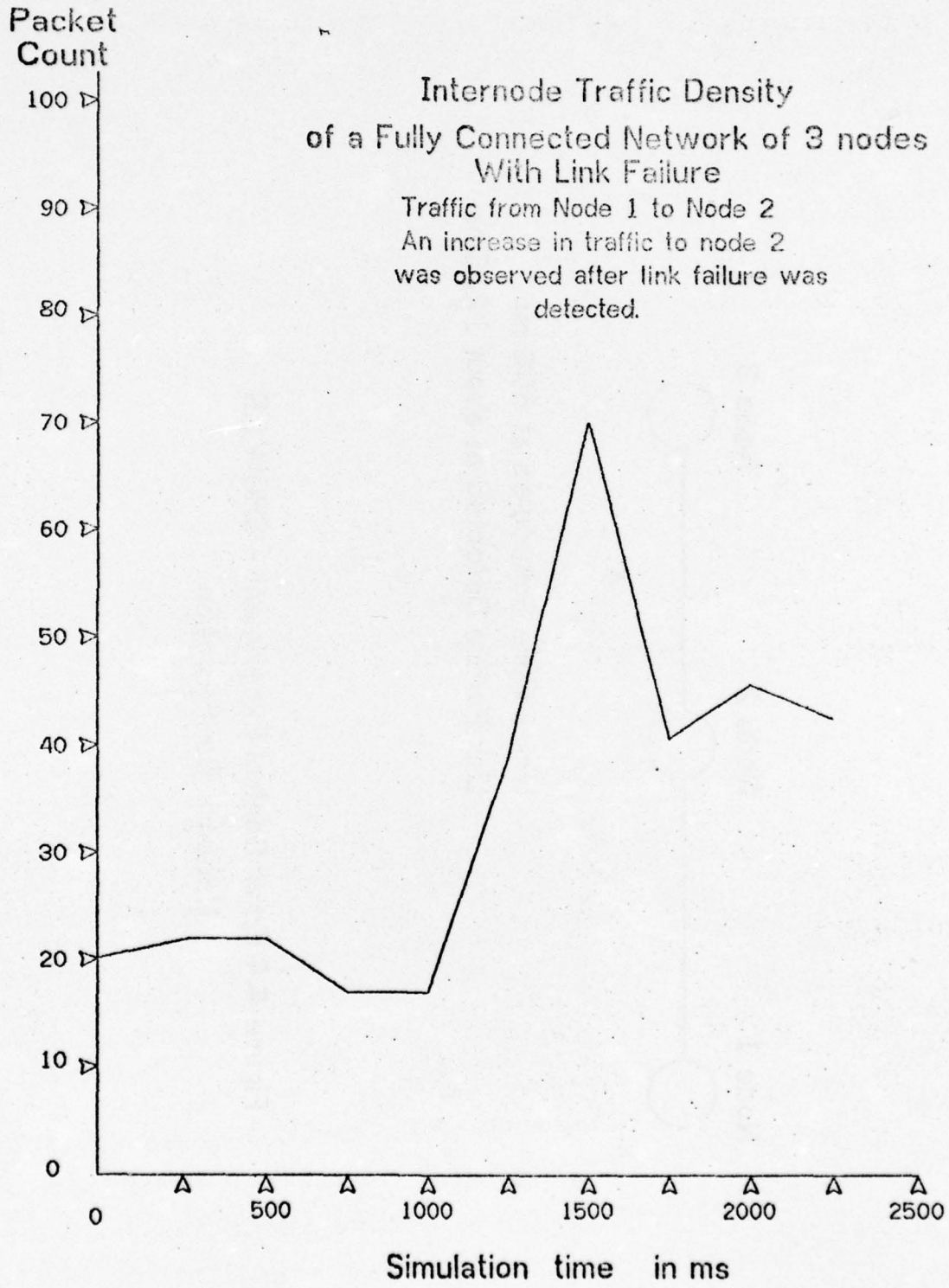
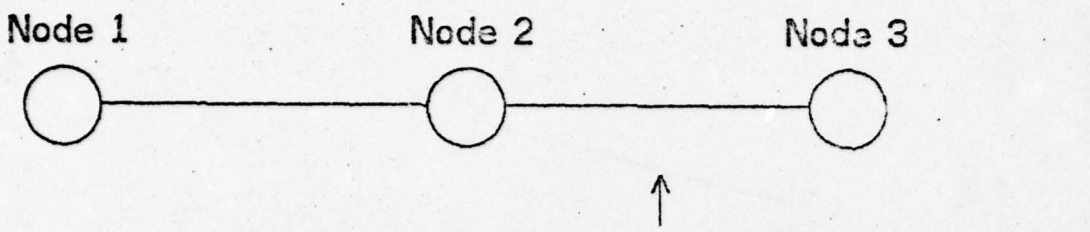


Figure 4.3 Test Case SHOW1.A - Internode Traffic Density



Link Failure Scheduled at 600 ms
Link Failure Detected at about 1225ms

Figure 4.4 Error Control Experiment - SHOW1.B
Network Configuration

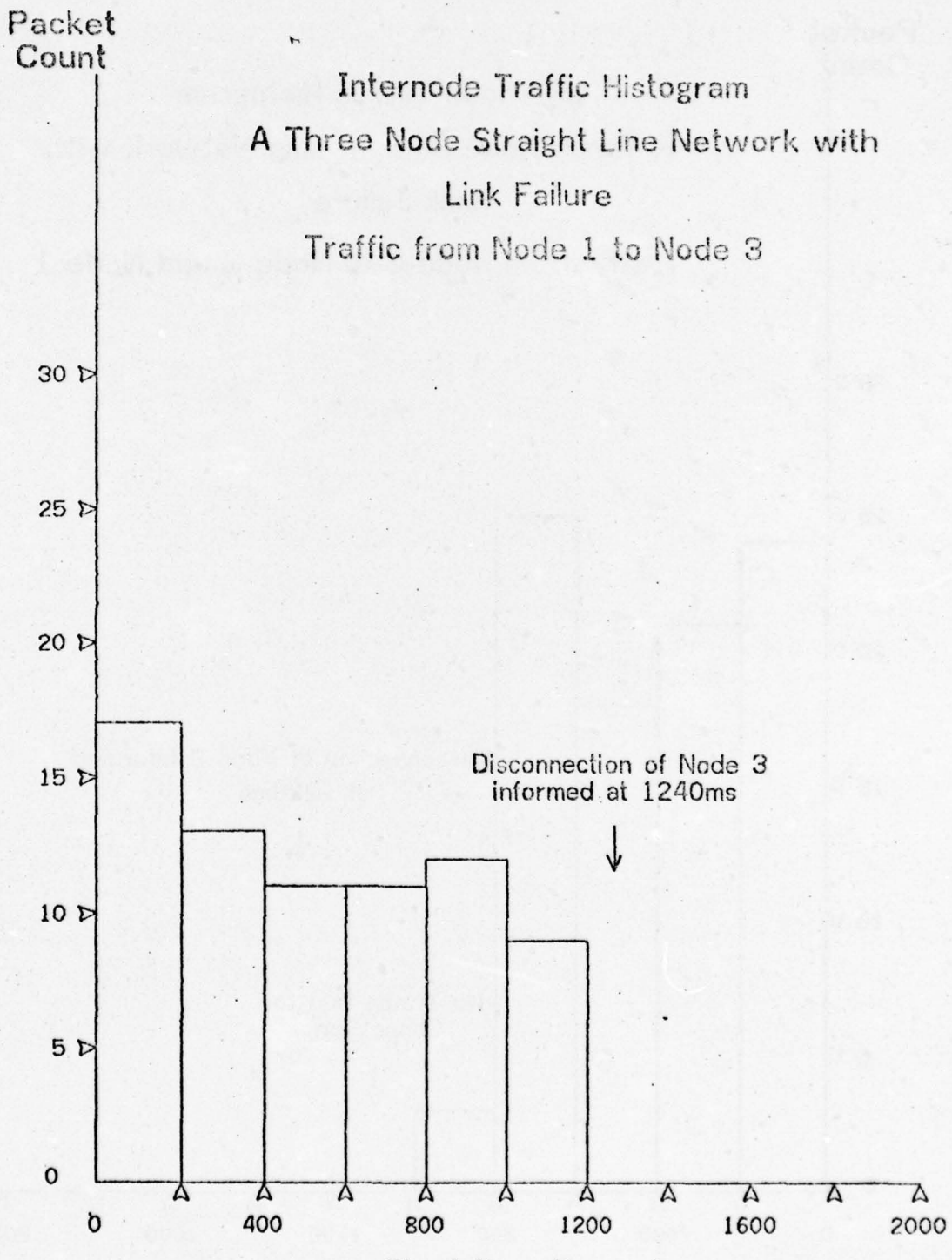


Figure 4.5 Error Control Experiment - SHOW 1.B

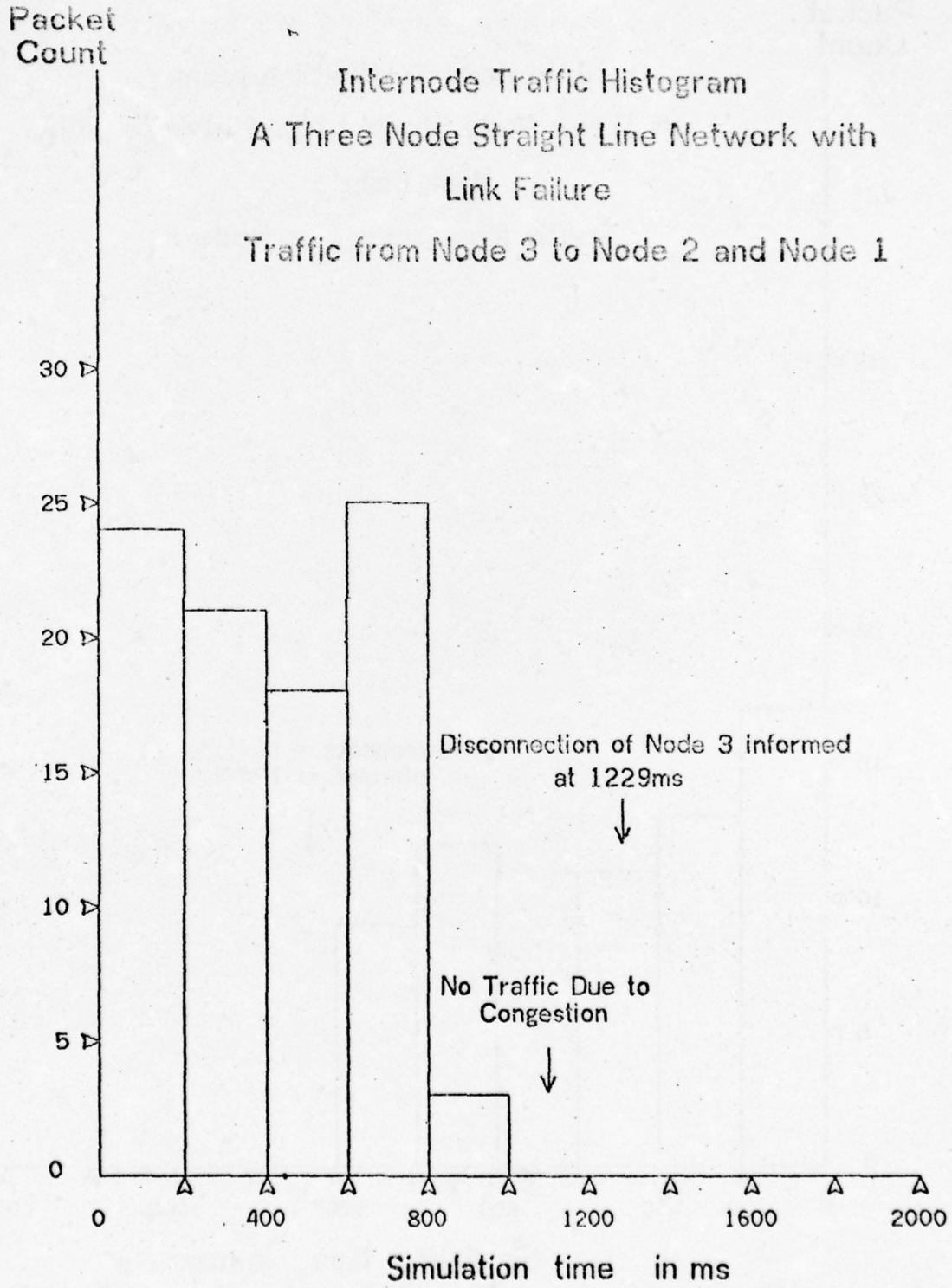


Fig. 4.6 Error Control Experiment - SHOW1.B

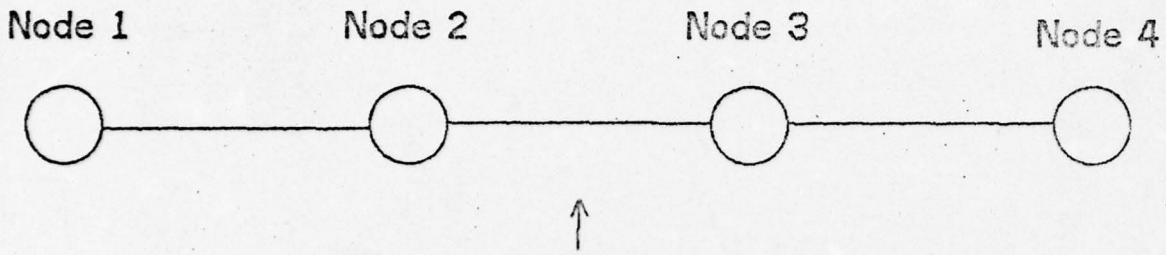


Figure 4.7 Error Control Experiment - SHOW1.C

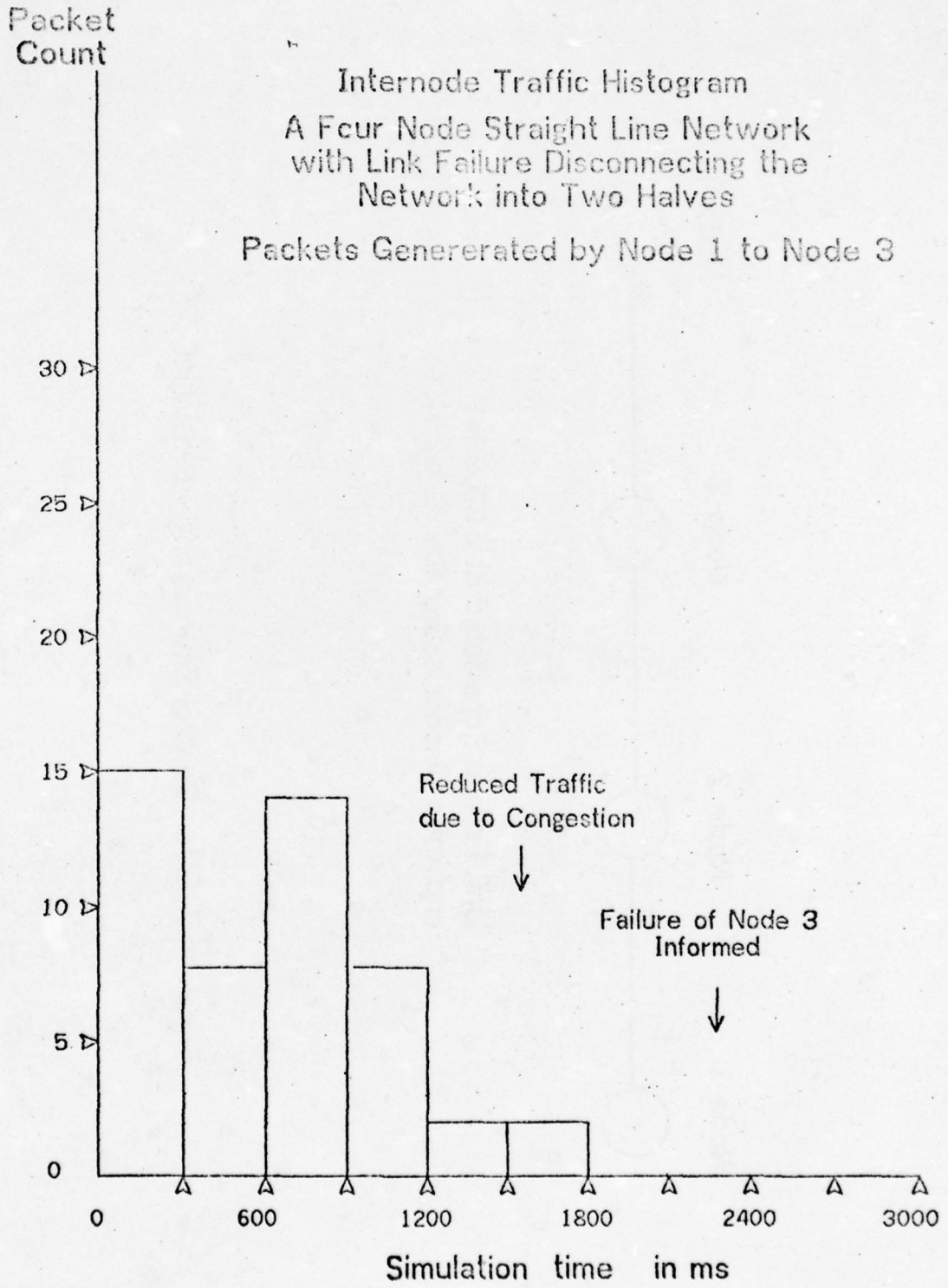


Figure 4.8 Error Control Experiment - SHOW1.C

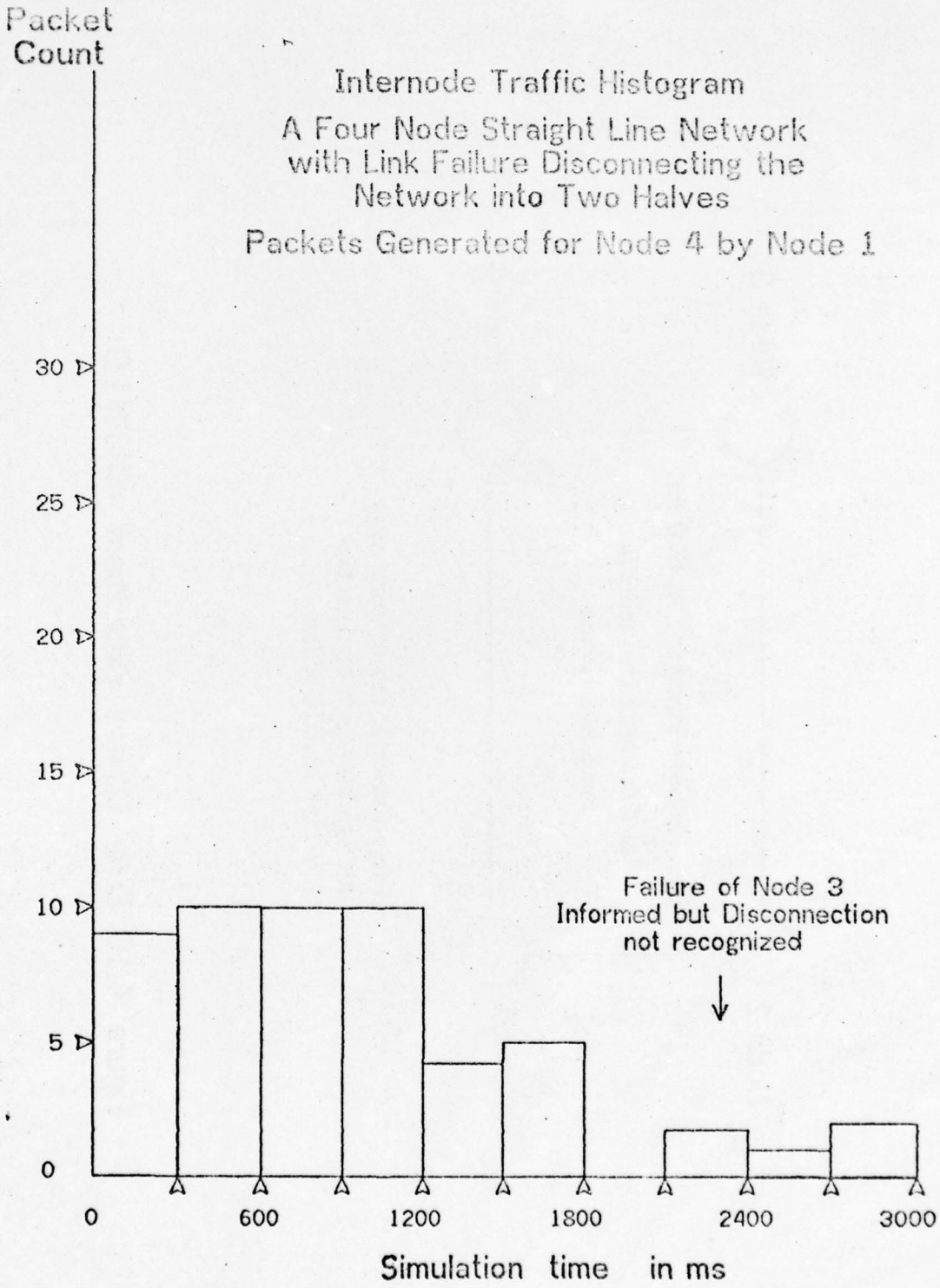
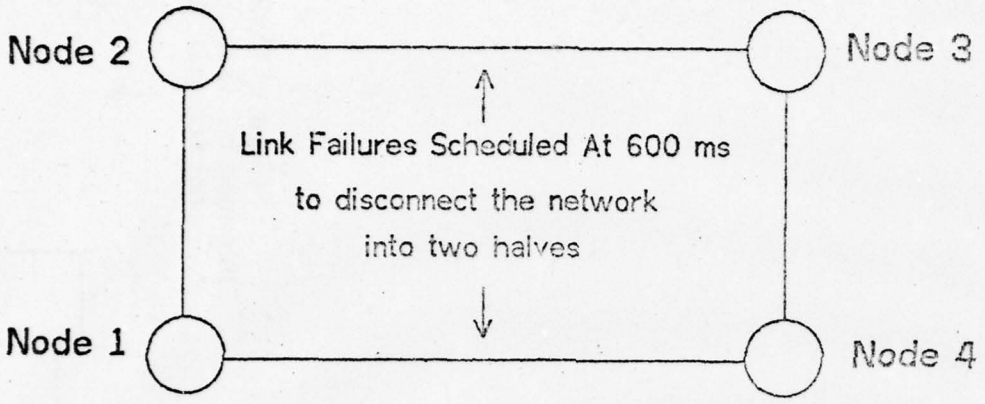


Figure 4.9 Error Control Experiment - SHOW1.C



In this example, the protocol was intelligent enough to detected the disconnection.

Figure 4.10 Error Control Experiment -SHOW1.D

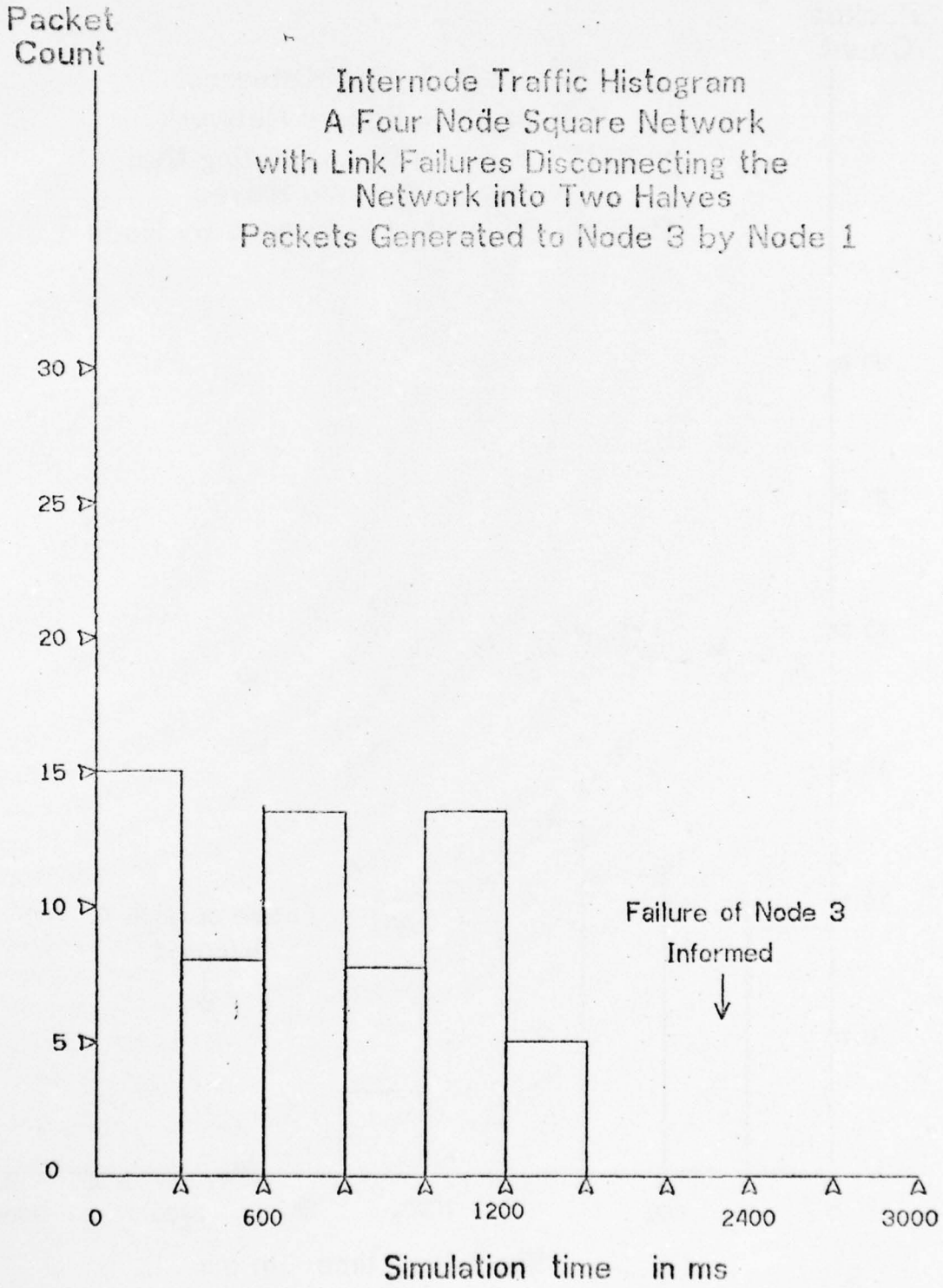


Figure 4.11 Error Control Experiment - SHOW1.D

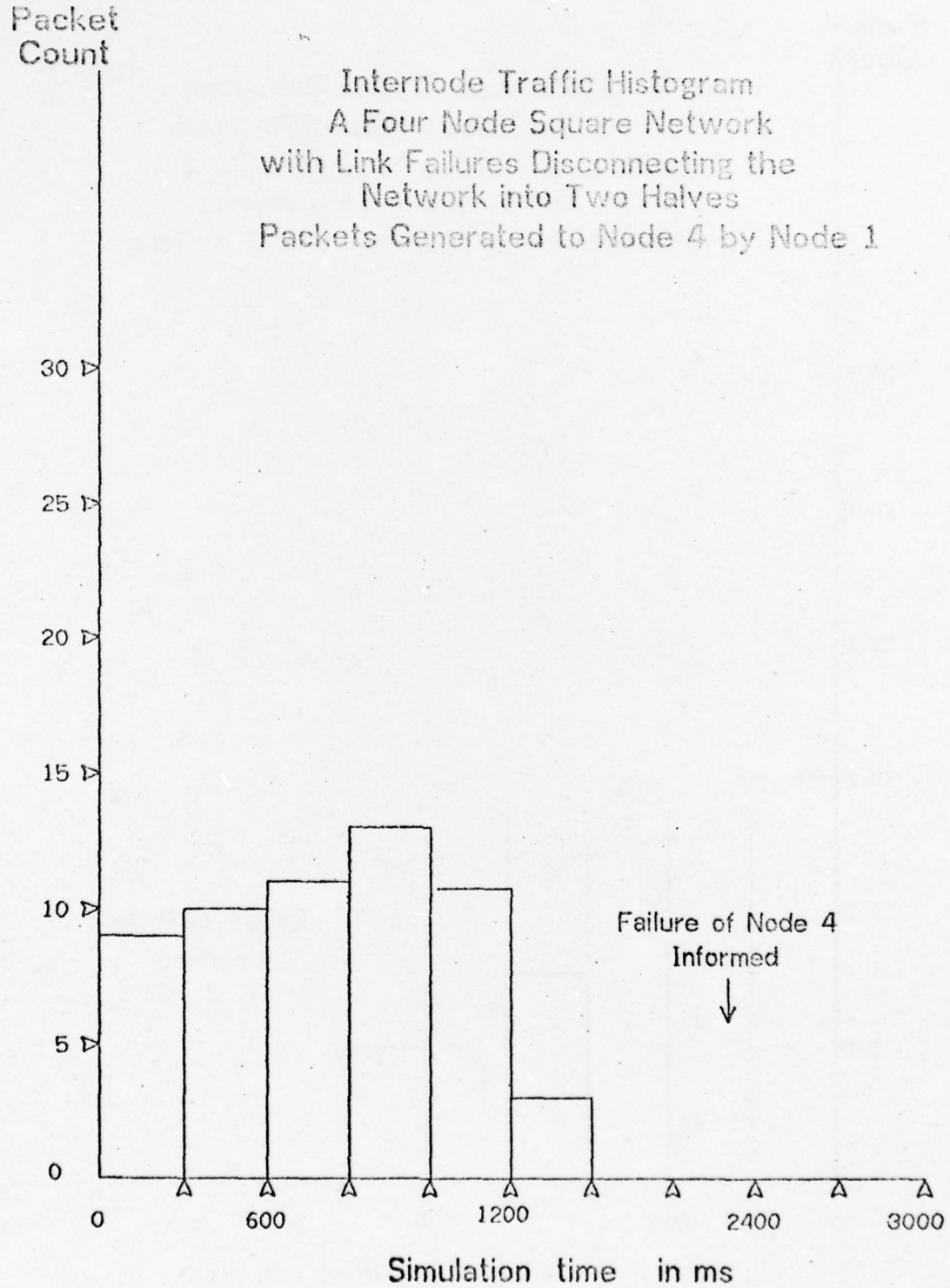


Figure 4.12 Error Control Experiment - SHOW.D

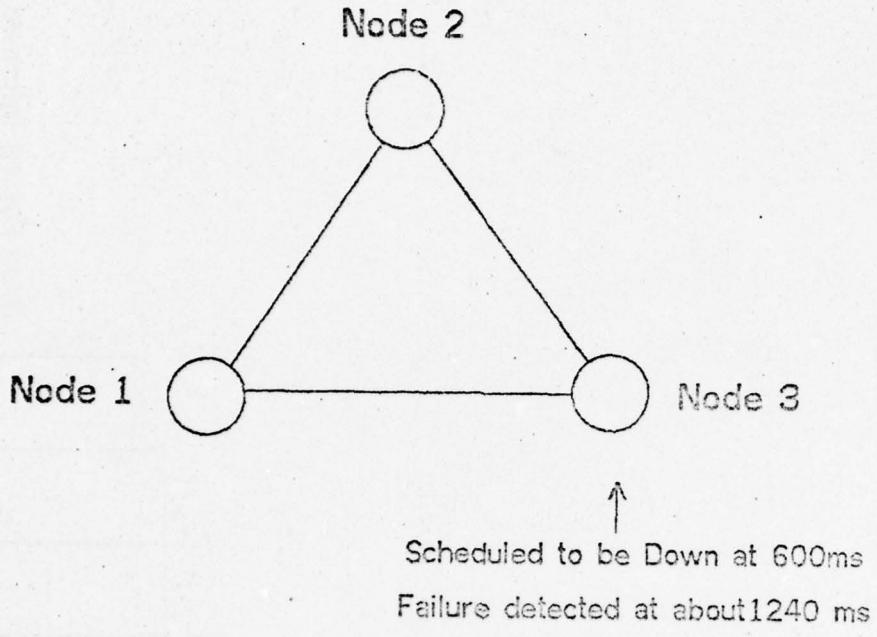


Figure 4.13 Error Control Experiment - Show2

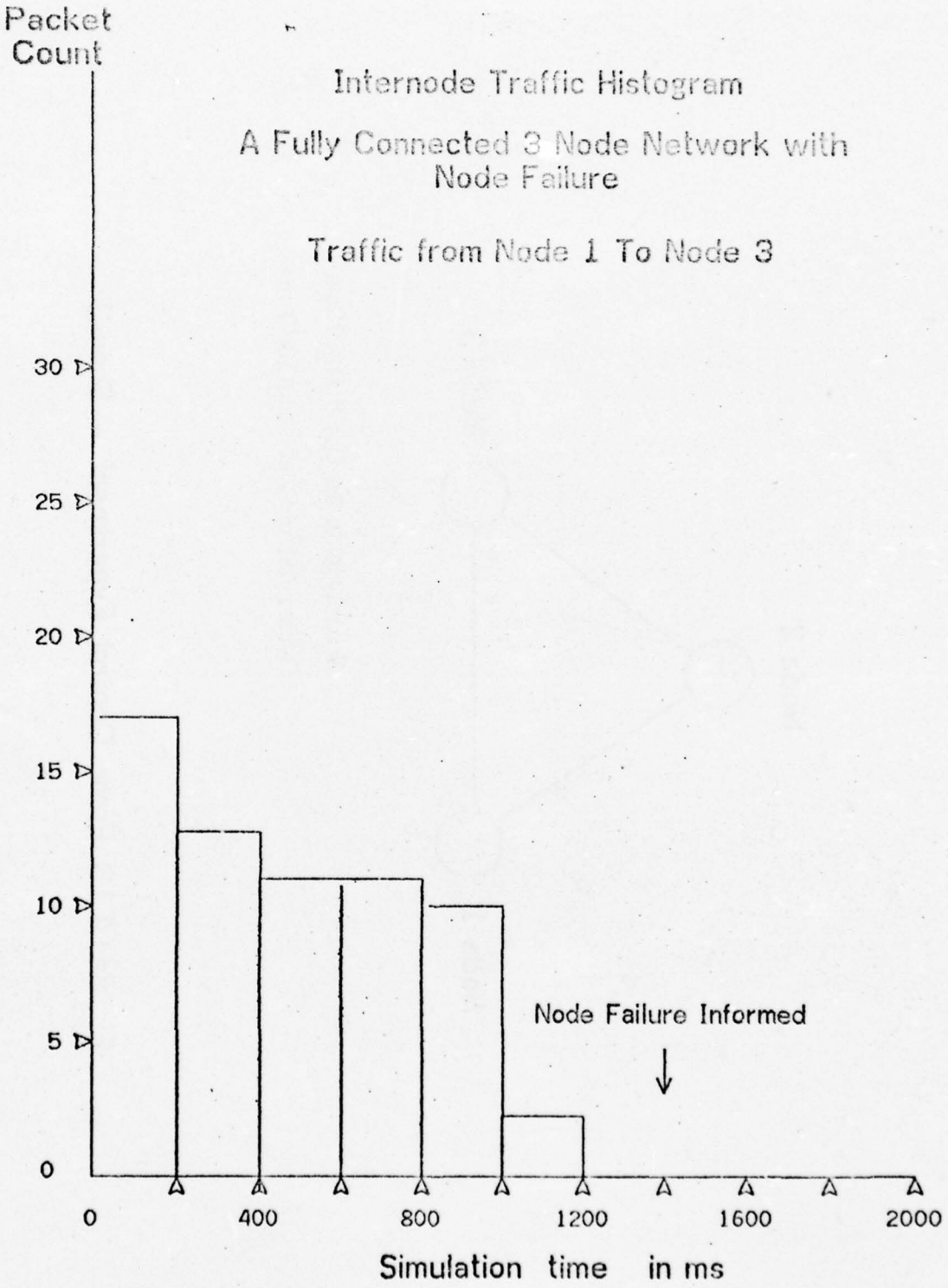


Figure 4.14 Error Control Experiment - Show2

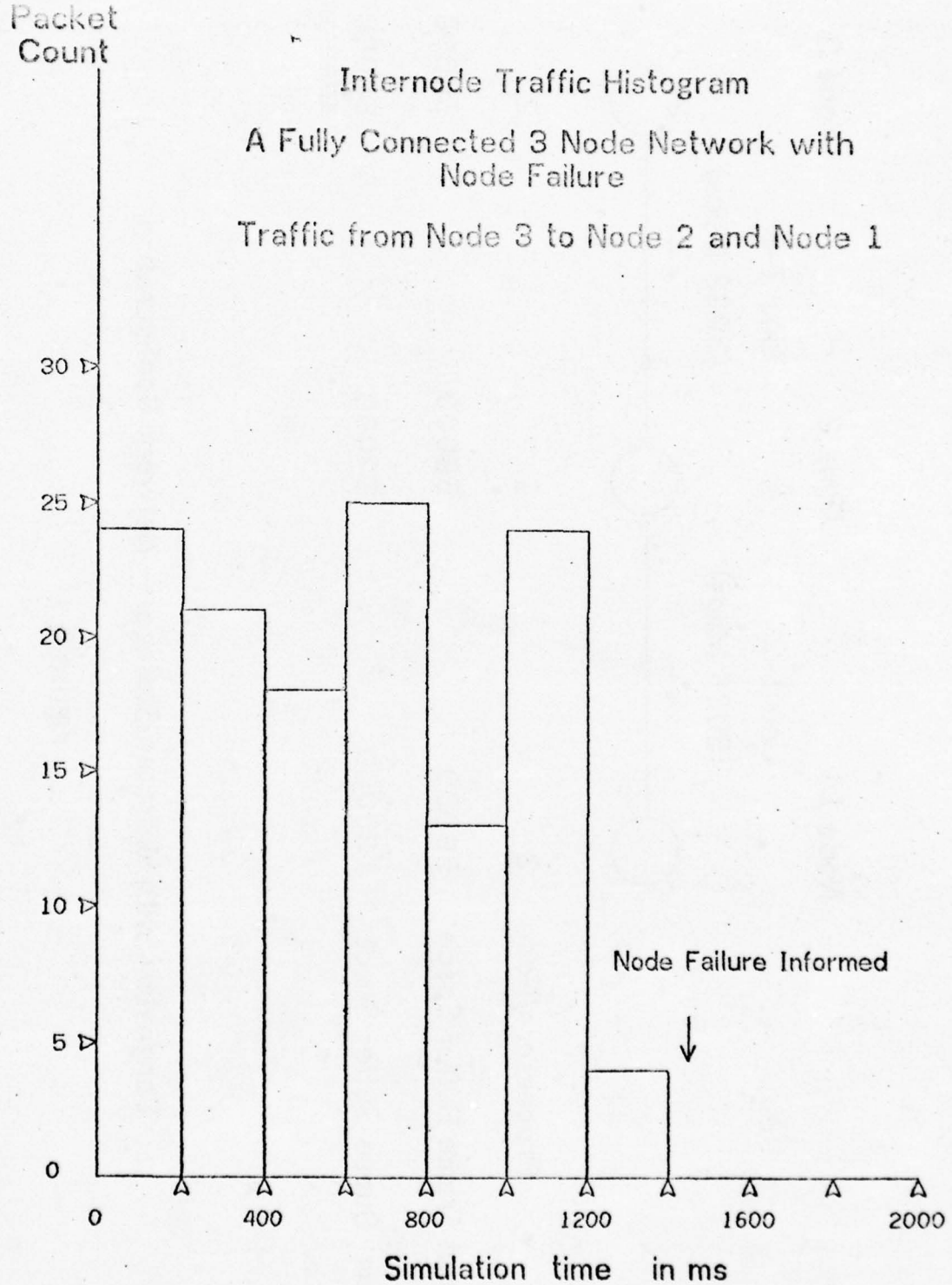
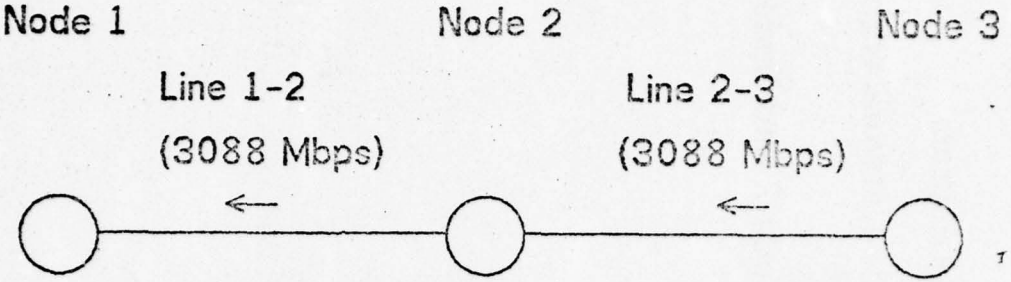


Figure 4.15 Error Control Experiment -SHOW2



Processors/Node	2	1	2
Input Frame Buffer Space	56000	56000	56000
Holding Queue Buffer Space	128000	64000	640000

Congestion with Unlimited Storage -- Network Configuration

Figure 5.1

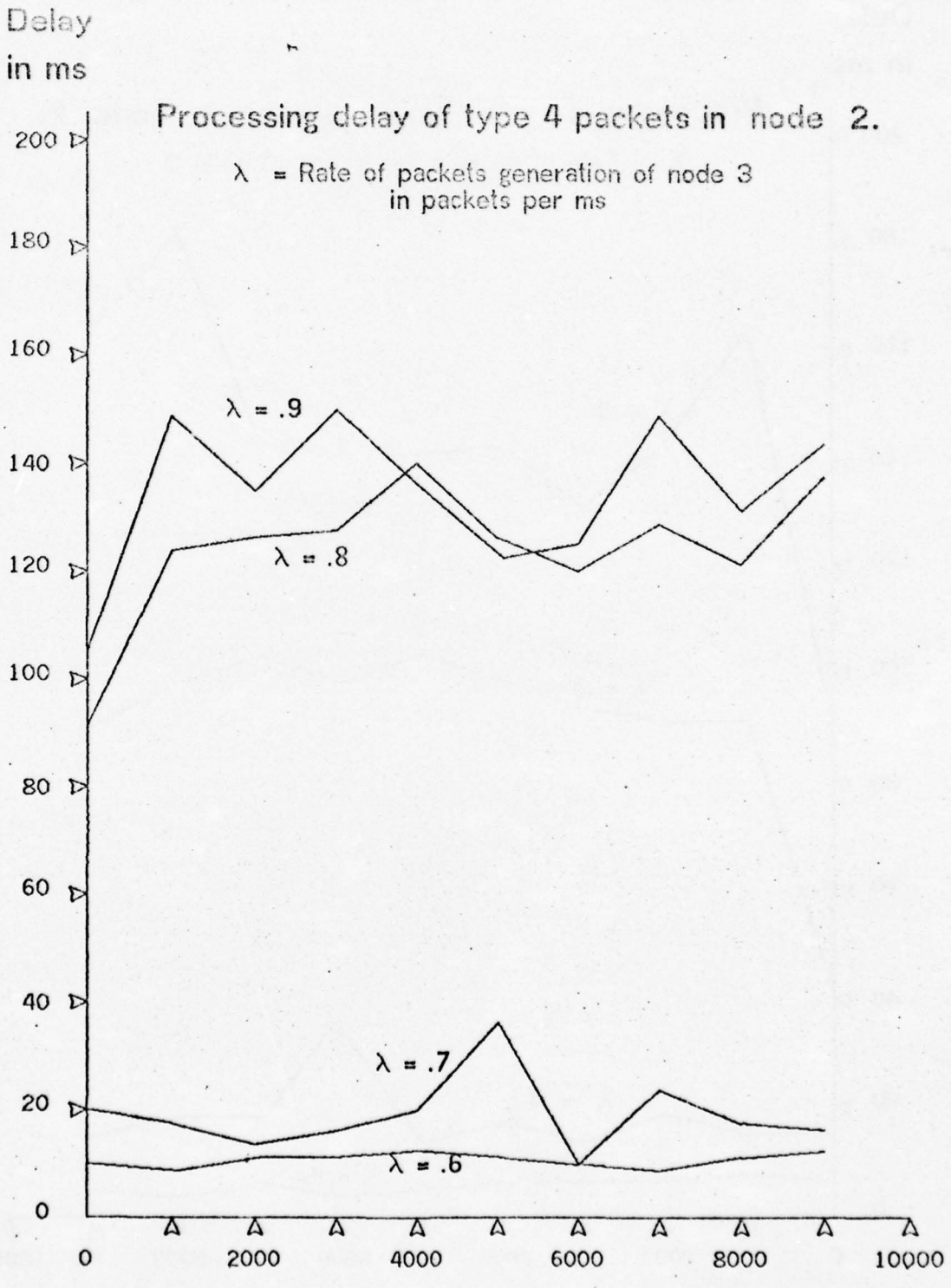


Figure 5.2 Congestion with Unlimited Storage

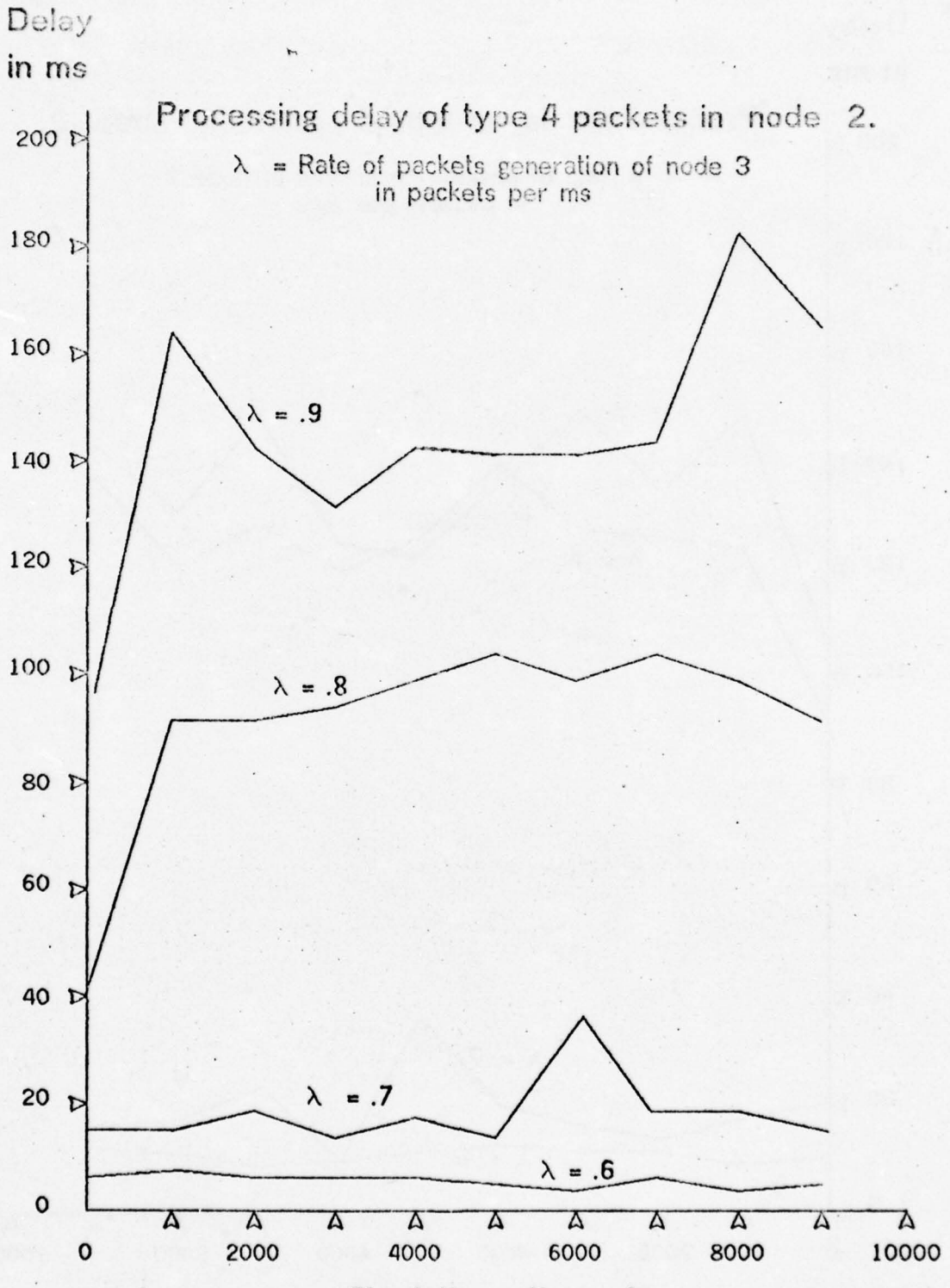
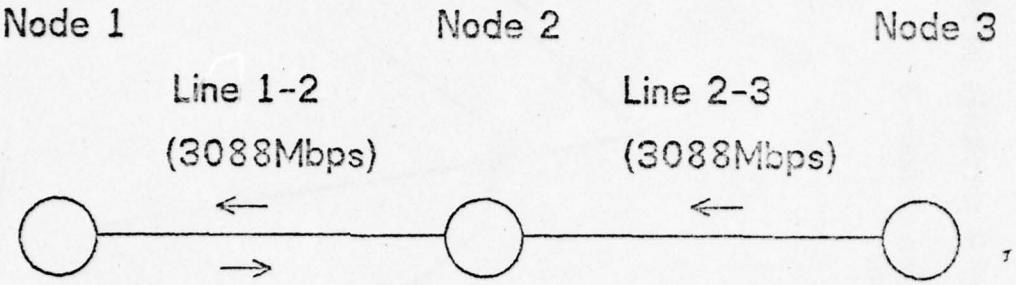


Figure 5.3 Congestion with Unlimited Storage



Processors/Node	2	1	2
Input Frame Buffer Space	56000	56000	56000
Holding Queue Buffer Space	128000	64000	128000

Congestion with Limited Storage - Network Configuration

Figure 5.4

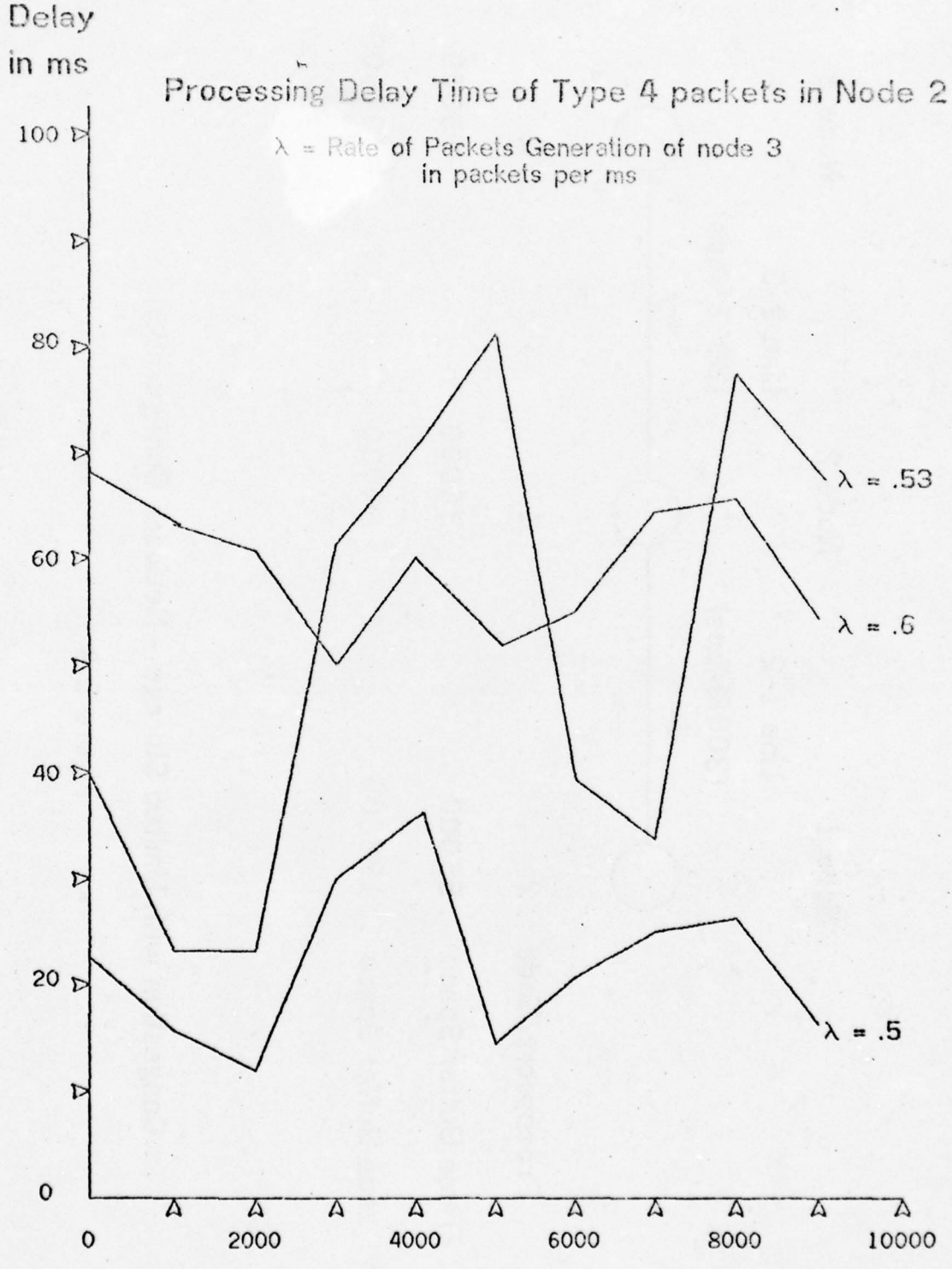


Figure 5.5 Congestion with Limited Storage

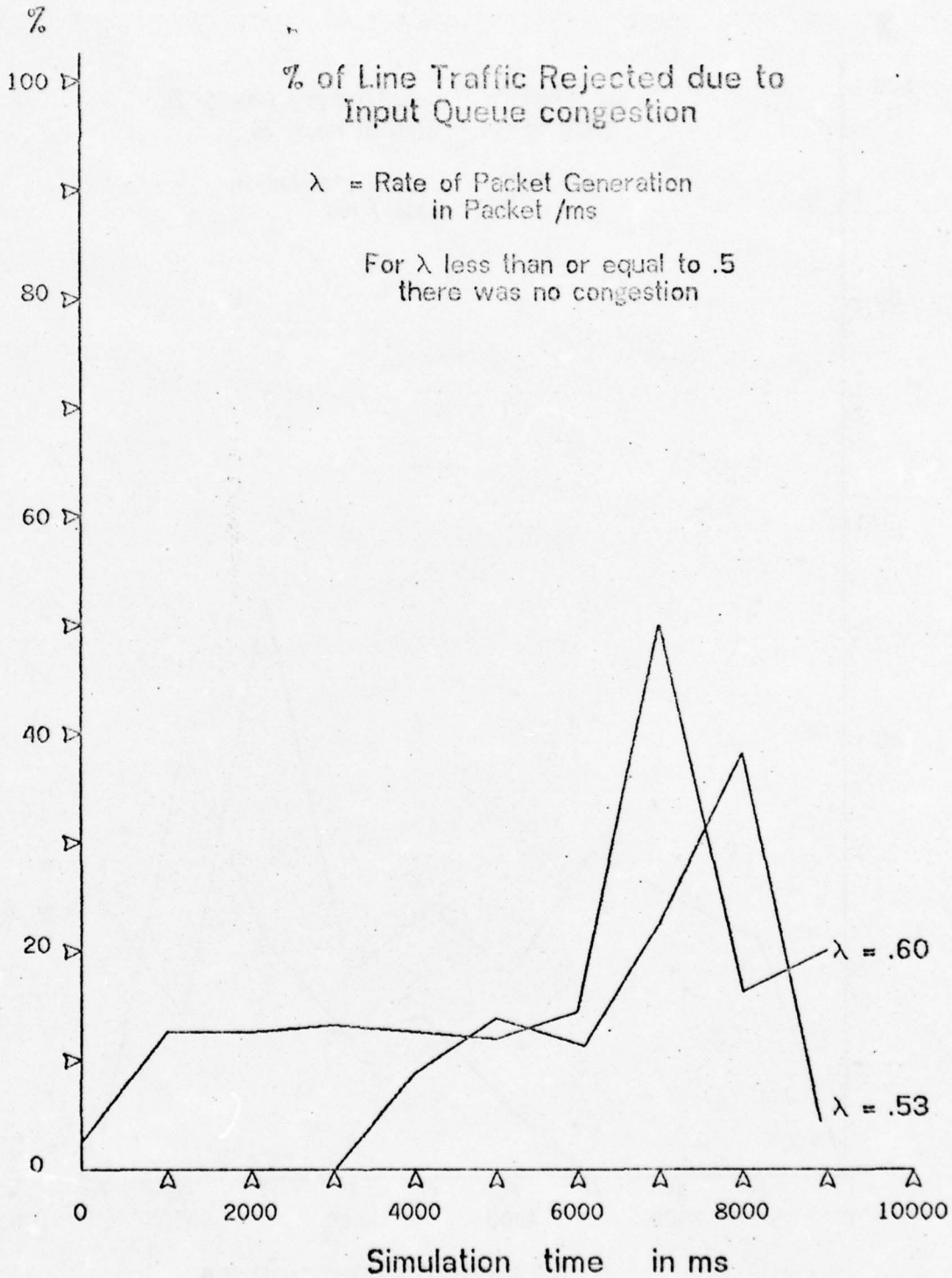


Figure 5.6 Congestion with Limited Storage

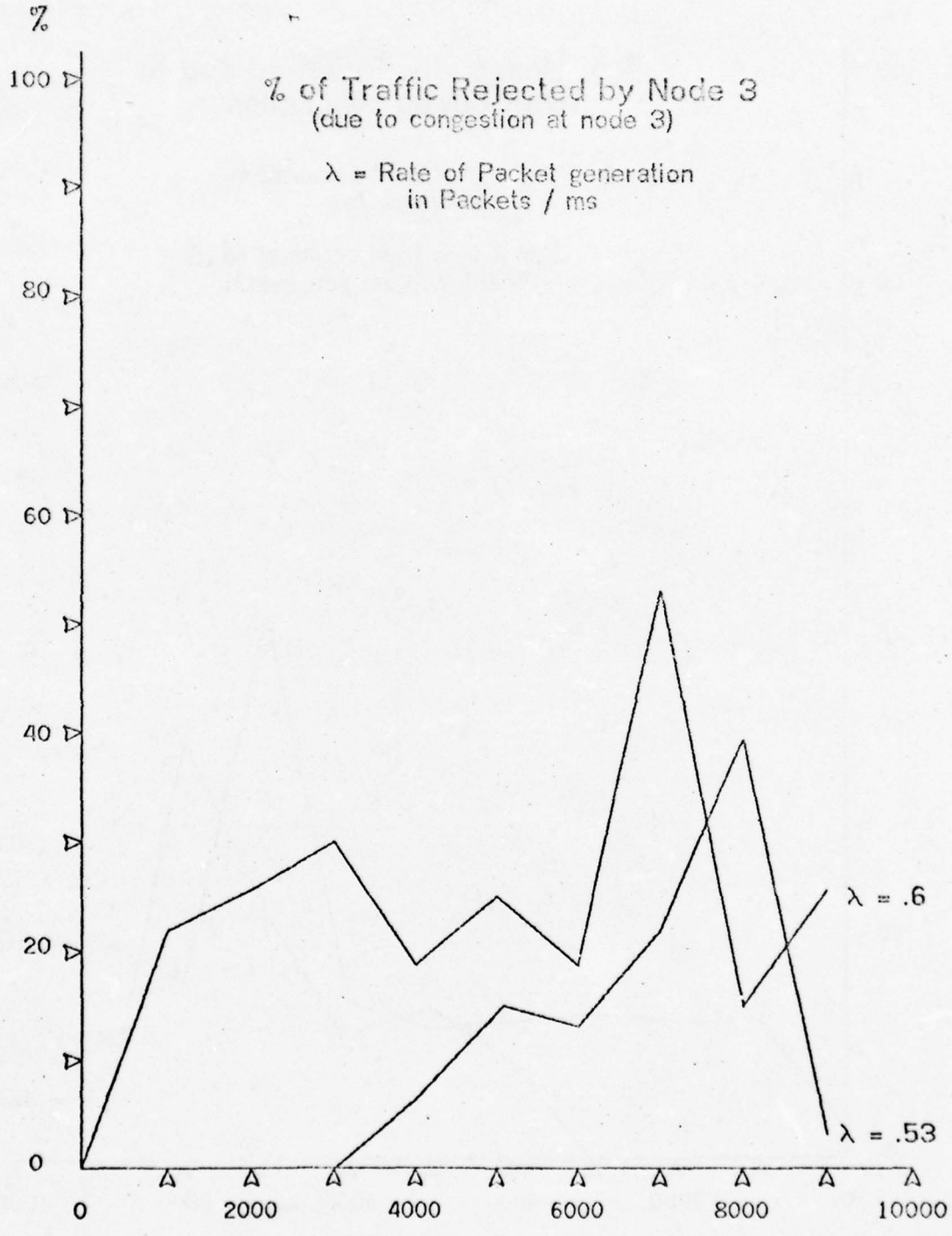
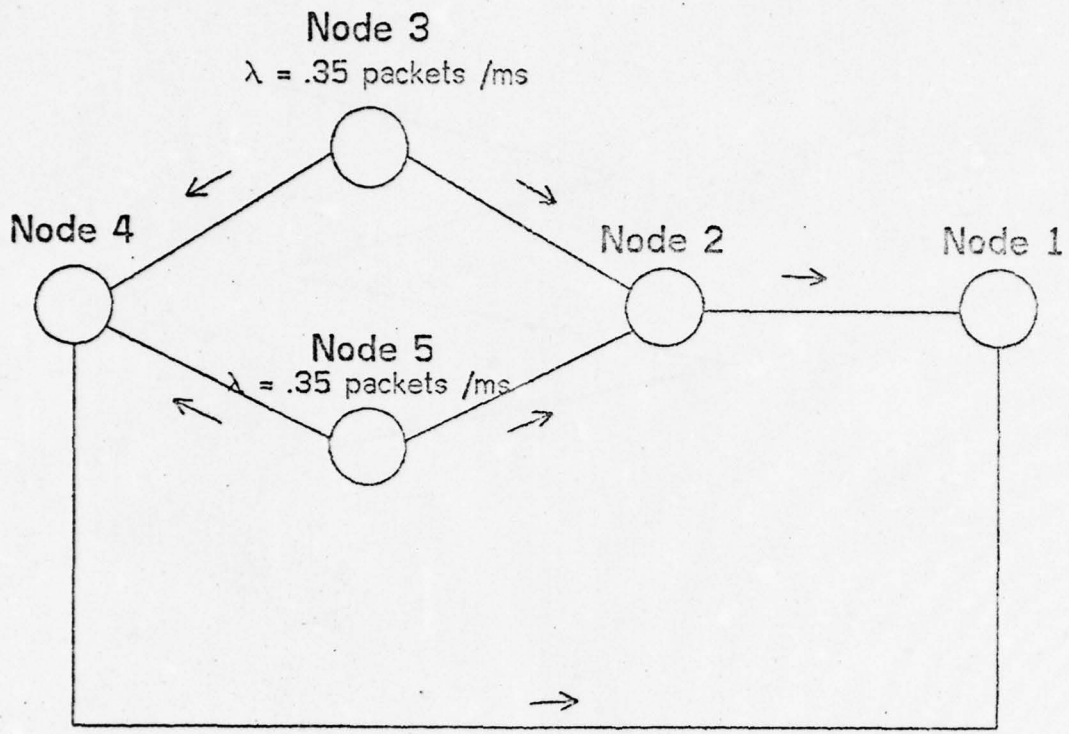


Figure 5.7 Congestion with Limited Storage



λ = Rate of Traffic Generation

Figure 6.1 Routing Inefficiency - Network Configuration

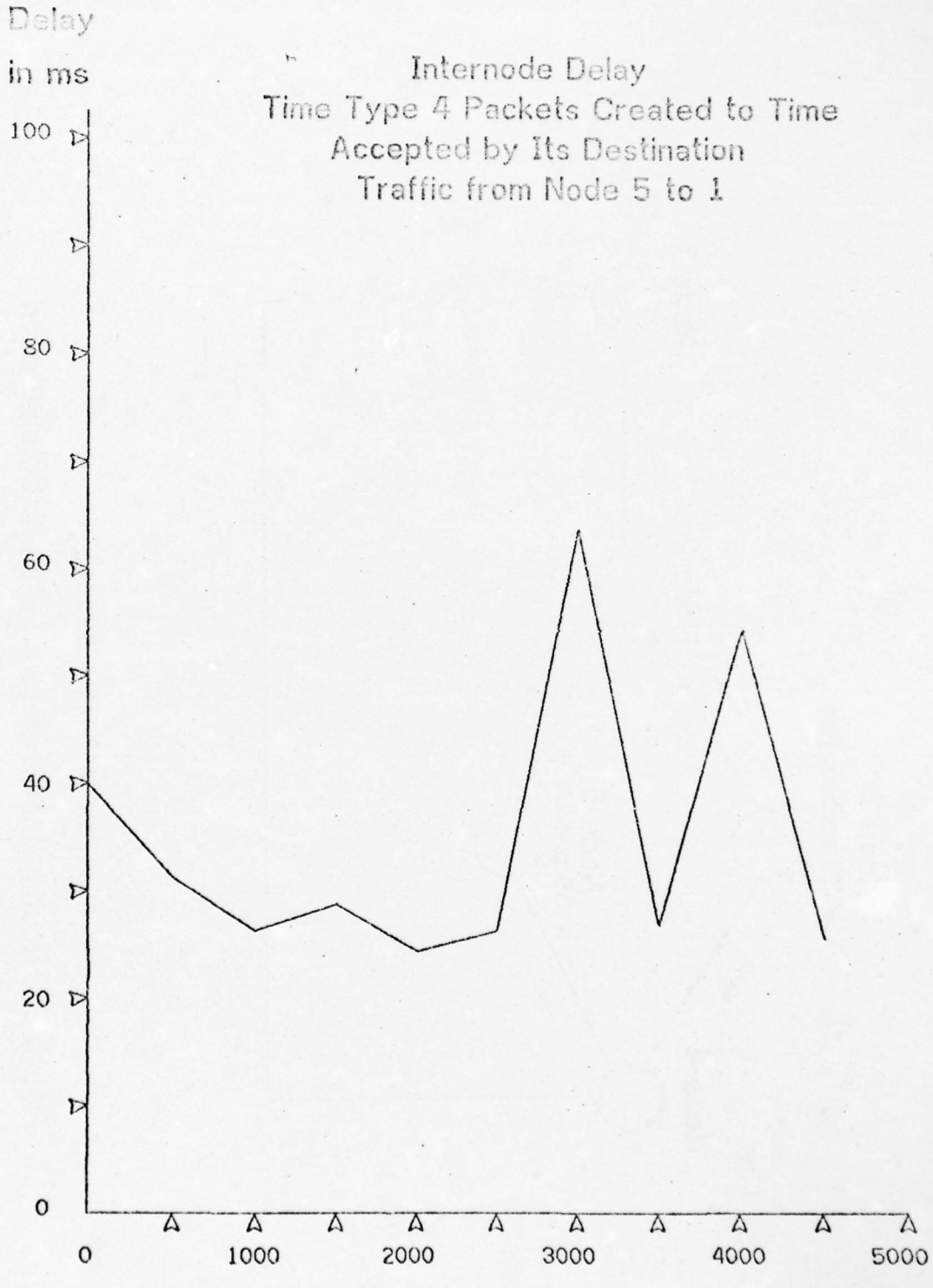


Figure 6.2 Routing Inefficiency - Traffic Delay

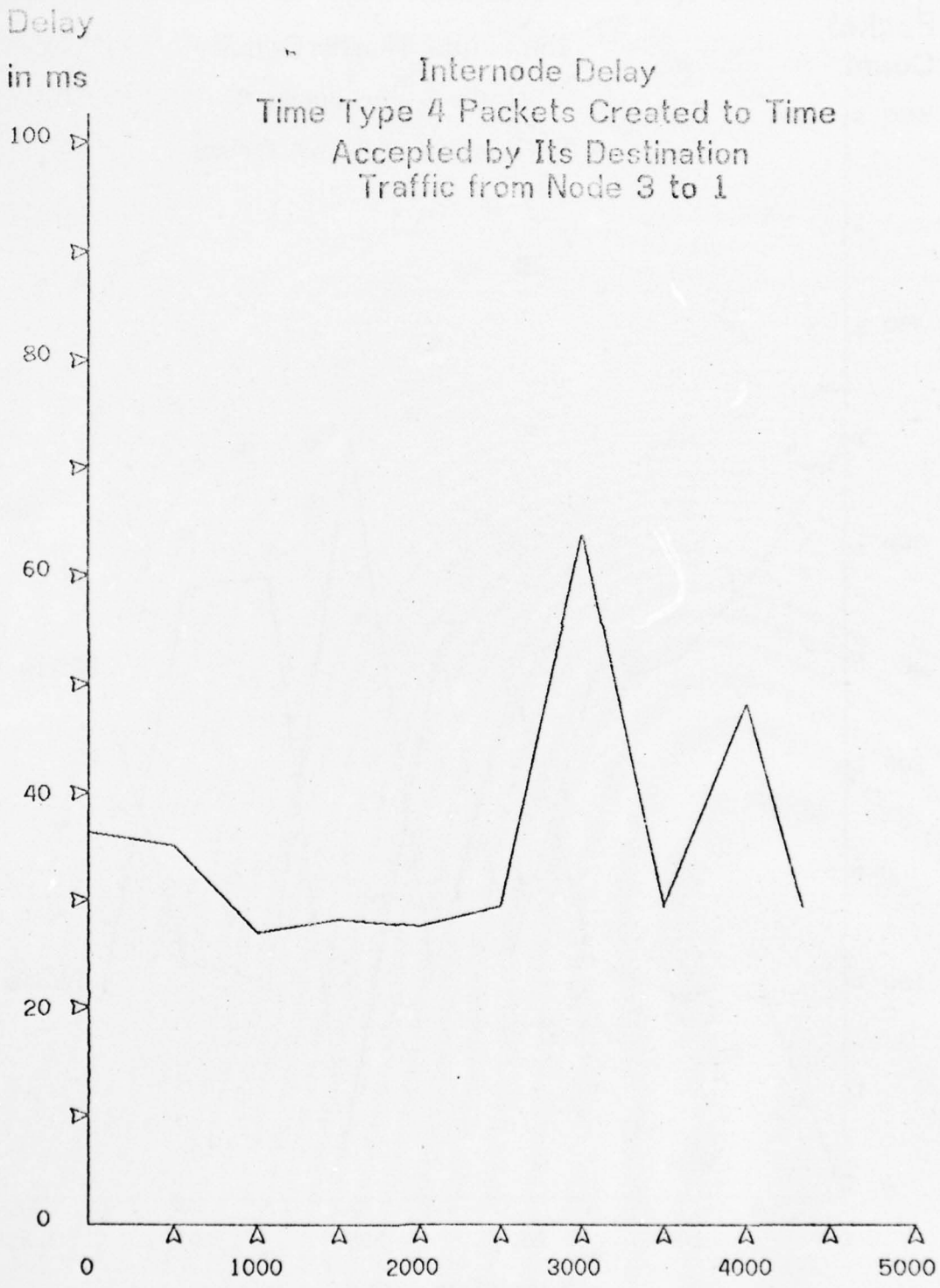


Figure 6.3 Routing Inefficiency - Traffic Delay

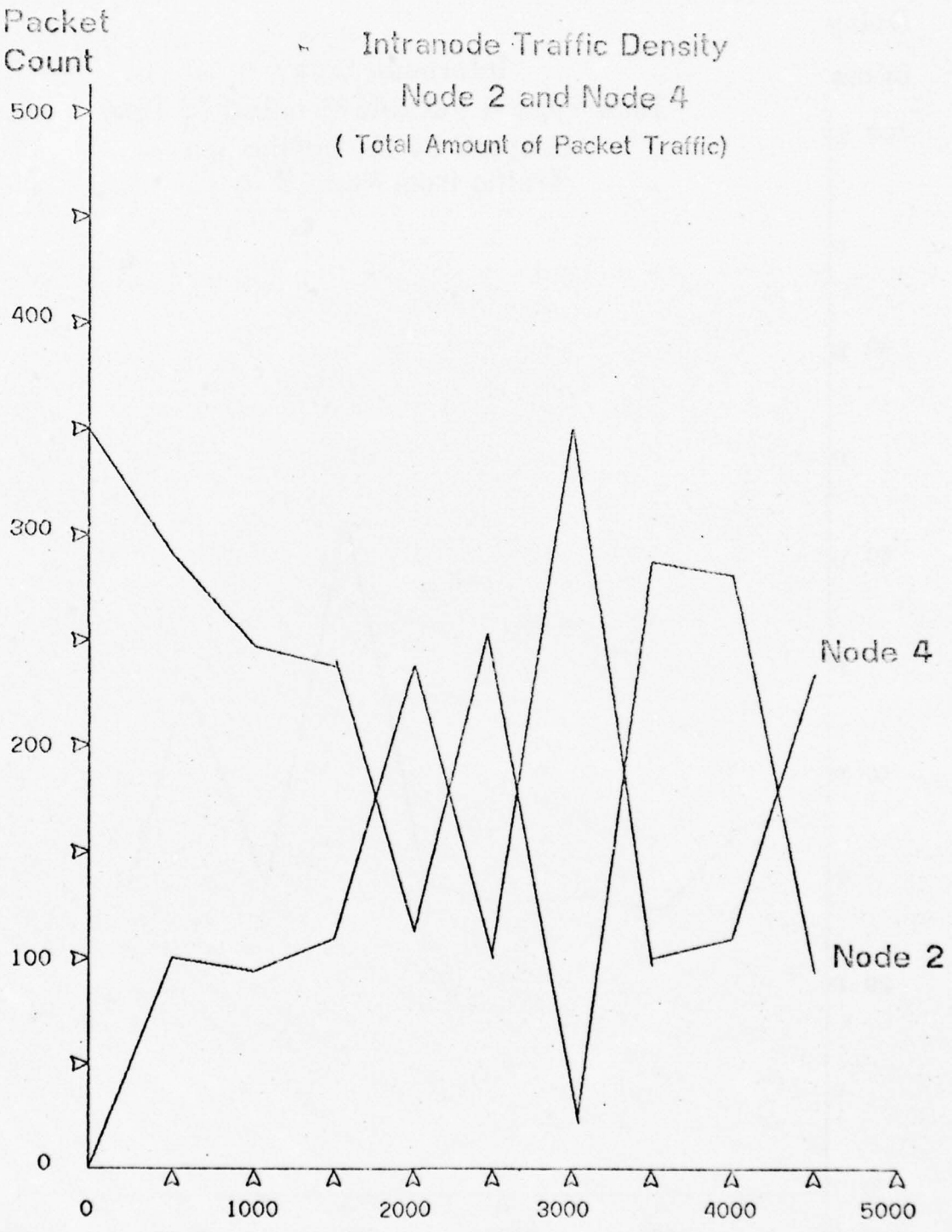


Fig 6.4 Routing Inefficiency - Traffic Density

Part IV

Packetized Voice and Data Networks

William Paulsen
Mario R. Barbacci

1. Packetized Voice/Data Network

The initial idea for the investigation of a packetized voice and data network comes from research currently active at MIT Lincoln Laboratory. The paper by Forgie and Nemeth [For76] describes a PVC, Packetized Virtual Circuit, integrated communication network. The research being reported here covers a segment of the original PVC concept, initial investigation and simulation experiments of a similar network. The areas not covered here are the Statistical Flow Control routing scheme, and a theoretical model analysis.

This report will discuss: Packetized Voice/Data (PVD) network concept and motivations for initial study; expected capabilities of such a network; the techniques and input used in the simulation experiments; analysis and conclusions from this work.

1.1. Overview

The approach to a digitized voice/data network scheme described here differs from most other proposals. In the SENET scheme described by Coviello [Cov75], voice traffic is treated in a circuit switched fashion, and data traffic in a packet switched fashion. The method here is to treat both types of traffic in a similar way: packet switched. The concept described by Forgie utilizes a packetized virtual circuit scheme, in which a source to destination path is first established, and the link is used only when data is required to be transferred. The PVD scheme utilizes a dynamic routing in which packets are handled and routed on an individual basis, dependent on the state of the network. The possibility of having a network using a combination of both these routing methods is discussed later.

The ability of a PVC/PVD network to utilize characteristics of conversational voice

communication to an advantage, which a SENET network could not, was an important motivation for this initial study at CMU. The silent gaps in ordinary speech are unnecessarily transmitted in a circuit switched network, particularly SENET. But during these gaps in a PVD network, other data may be transmitted; there appears to be an increase in the useful bandwidth of a node to node link. For a circuit switched network to approach this would be expensive; the tables containing data on the switched paths would need to be redefined, or data may be inserted in an available channel. This would require that a data packet fit and that the network know it is not voice data.

1.2. Class I Voice/Real Time Data Transmission

In the PVD network concept all voice and real time data are in packetized form. For each active call, the source node accepts data from the local host and a network packet is created. The packet contains header information: destination, call number, etc. In the case of voice traffic, when a silent gap is detected by the vocording device, no packet is created, allowing other data to proceed. For incoming Class I traffic, the packets are processed in order of arrival; a lack of packets for a particular call would indicate silence. The possibility of packets arriving out of sequence has not been studied. The problem might become evident in larger networks and as network traffic increase to such an extent that the routing algorithm decides to select another path with a shorter delay.

At all times Class I packets are given priority over Class II and III packets by the node processor. Class I packets are not acknowledged as they are received.

A PVD network has the ability to accommodate real time devices which generate

data at various periodic time intervals. The requirement that new data appear every 10 milliseconds, as in SENET, is not made.

1.2.1 Silent Gaps in Speech

The statistics on silent periods in conversational speech are presented in [Bra65]. A speech detector and recorder device is used to measure the duration of talk-spurts and silent gaps. Silent periods of less than 200 milliseconds are not considered gaps and are considered a part of talk-spurts. The detector is designed to trigger at a listener understandable level.

The results indicate that a person talks 44.3% of the time, silent 55.7%. The mean length of talkspurts and pauses is .75 seconds; the median about 1.5 seconds. The paper indicates this is in agreement with similar, unpublished measurements. The paper is quick to point out that the conversations used may not be typical of all telephone conversations.

The conclusion from this report which is applicable to the PVD study is the ability to allow a voice call to request, say, a 10 kbps channel, but to assume that, on the average, it will be used to only 44.3% of capacity. For a large number of calls, 50 - 100, this advantage can be used safely.

1.2.2 Consequences of Utilizing Speech Gaps

While taking advantage of these speech gaps, the possibility of almost all conversationists talking at the same time and overloading the system becomes apparent. When operating at nearly full capacity and this condition arises, without any escape procedures, the end to end delays in the voice calls may become intolerable; the internal node memory would overflow, losing data haphazardly.

The PVC proposed solution is to drop half the packets within the node. It is expected that this will happen infrequently enough so that the effects will be minimal, the probability being 1%.

An alternative to this method is to utilize a dynamic routing scheme in the network. When a particular node determines an increase in traffic it is handling, routing control data can be transmitted before a deadly threshold is reached. This plan is contingent on the network's ability to handle congestion and routing; this is not discussed here.

A second alternative is to modify individual voice packets. By shortening some packets it might be possible to continue throughput. The capability of decreasing the size of an arbitrary voice packet in the network is dependent on the technique used to generate the packet. If the representation is such that dropping bits off a voice packet merely increases distortion in the end, and does not make the packet useless, then this scheme is worth investigating.

1.3. Class II and III Data and Network Control Traffic

Data traffic is handled in the same fashion as in the SENET concept. Data packets are acknowledged node-to-node as they pass though the network, and end-to-end when the packet's destination is reached. For each data packet transmitted, an acknowledgement packet of about 100 bits is created. This type of traffic will be discussed.

The processing requirements for a PVD network seem to be greater than for a SENET network. In SENET the voice slots for each call do not invoke much processing; the slots are copied into the appropriate output queue area according to the local Class I mapping table. For the PVD node, each packet's header must be scanned, and

then the appropriate action taken. This greater requirement is offset by the greater link utilization obtained.

1.4. Additions to the Command Language Interpreter

The PVD network simulator is the SENET simulator described in Part 1 with appropriate modifications.

For the PVD experiments, these commands are valid.

1.4.1 VSPECIFY

The VSPECIFY command takes the following form:

VSPECIFY <type>/L:<arg1> R:<arg2> S:<arg3>

The VSPECIFY command specifies a particular type of Class I (voice) transmission mode. Up to 4 modes may be defined. All voice calls in an experiment are of one of these modes. There are no default values.

<type> Specifies the name of this mode. It is an integer in the range 1:4.

<arg1> Defines the length, in bits, of the packets generated in this mode.

<arg2> Defines the total transmission rate, in bits per second, for a call of this type.

<arg3> Defines the percentage of these voice packets generated which are silent.

1.4.2 VGENERATE

The VGENERATE command takes the following form:

VGENERATE <node>/T<type> /D<node1>:<arg1> /D<node2>:<arg2> ...

The VGENERATE command defines the voice traffic generated by the host attached to this node to other nodes in the network. There are no default values, and all nodes referred to must have been previously defined.

<node> Defines the source node for the calls defined by this instance of the command.

<type> Defines the transmission mode type, as defined by the VSPECIFY command, for the calls specified in this command line.

<node1>:<arg1> Specifies a destination node and defines the number of calls to be allocated.

1.4.3 MODE

To activate the PVD simulation sections of the Network Simulator, the MODE command must be specified as:

MODE X

1.4.4 HELP

The HELP command contains assistance on the VSPECIFY and VGENERATE commands.

1.4.5 SHOW

The SHOW command will display the setting of the VSPECIFY and VGENERATE network parameters.

1.4.6 Irrelevent Commands

The following commands and their parameters have no bearing on a PVD simulation experiment:

FRAMETIME, ALLOC, VFRACTION, VRATE, VDIST

1.5. Additions to the Tracing Facilities

1.5.1 NEW VIHOST

The NEW VIHOST initialization tracing entries have the following form:

<time> NEW VIHOST <node>

The NEW VIHOST entry describes the initialization of a node PVD voice generator host.

1.5.2 CREAT VPKT

The CREAT VPKT packet tracing entries have the following form:

<time> CREAT VPKT <packet> <source> <dest> <type> <length>

The <type> parameter identifies the transmission mode type, as specified in the command language. The other parameters are the same used in the CREATE PKT entry.

2. Packetized Voice/Data Experiments

The experiments involve a 2 node network, with a 2-way link connection. The parameters were defined to be similar to those in the PVC [For76] report to attempt to verify results.

2.1. Assumptions

1 The link transmission rate is 1544 kbits/second.

2 Voice packets (Class I) contain 128 bits, of which 96 bits are actual data and 32 bits network overhead. Calls are assigned before the simulation has started, none are initialized during simulation. For each call established, packets are injected into the network with probability 0.45; otherwise they are assumed to be silent, and are not created. The vocording technique is 16 kbits/second CVSD; this will generate a packet every 6 milliseconds, for each call. With the 32 bits/packet overhead the total is 21.3 kbits/second.

3 Data packets are of various lengths; for each packet 32 bits are for network overhead.

4 For each data packet sent from node to node, a control packet is generated and sent in the opposite direction to acknowledge receipt. These control packets are 100 bits long, and have priority under voice packets, but over data packets.

5 Each node is simulated by 1 to 4 processors, as specified in the results. These operate in parallel on the traffic stream. Packets are handled by the processors at the rate of 5 microseconds per 16 bit word.

6 The delay time as reported in the results is the total time the packet remains in the network - from the time the packet is created by the source host until the destination node host receives the packet.

7 The maximum data traffic is what is available after voice and control is allocated. The link utilization is the ratio of the total simulated traffic to the link capacity.

2.2. Results

2.2.1 Experiment 1a

Conditions:

voice packet size:	128 bits
calls:	100
type:	16 kbps CVSD
activity:	45% total active talk, 55% silent
data packet size:	128 bits
number processors:	1 per node
processor speed:	5 microseconds / 16 bit word
simulation duration:	250 ms, statistics collected after 10 ms
maximum data traffic:	245.89 kbps
actual data traffic generated:	243.75 kbps
actual voice traffic generated:	720 kbps
data header and control traffic:	335.148 kbps
voice header and control traffic:	239.985 kbps
utilization:	.996

Results:

total voice packet delay:	.25 ms
variance:	.02
data packet delay:	8 ms
variance:	20
95% of data packets have delay under	17 ms
maximum node memory required:	8000 bits

2.2.2 Experiment 1b

Conditions as in experiment 1b except:

number processors:	4 per node

Results:

total voice packet delay:	.15 ms
variance:	.01
data packet delay:	7.8 ms
variance:	15 ms
95% of data packets have delay under	16 ms
maximum node memory required:	6000 bits

2.2.3 Experiment 2

Conditions:

voice packet size:	128 bits
calls:	100
type:	16 kbps CVSD
activity:	45% total active talk, 55% silent
data packet size:	128 bits
number processors:	1 per node
processor speed:	5 microseconds / 16 bit word
simulation duration:	250 ms, statistics collected after 10 ms
maximum data traffic:	245.89 kbps
actual data traffic generated:	187.5 kbps
actual voice traffic generated:	720 kbps
data header and control traffic:	257.8 kbps
voice header and control traffic:	239.985 kbps
utilization:	.91

Results:

total voice packet delay:	.25 ms
variance:	.0375
data packet delay:	2.3 ms
variance:	3
95% of data packets have delay under 6 ms	
maximum node memory required:	4096 bits

2.2.4 Experiment 3

Conditions:

voice packet size:	128 bits
calls:	100
type:	16 kbps CVSD
activity:	45% total active talk, 55% silent
data packet size:	128 bits
number processors:	1 per node
processor speed:	5 microseconds / 16 bit word
simulation duration:	250 ms, statistics collected after 10 ms
control packet size:	32 bits (100 bits in other experiments)
maximum data traffic:	373.76 kbps
actual data traffic generated:	337.5 kbps
actual voice traffic generated:	720 kbps
data header and control traffic:	225 kbps

voice header and control traffic:	239.985 kbps
utilization:	.986

Results:

total voice packet delay:	.4 ms
variance:	.025
data packet delay:	2.25 ms
variance:	1.7

95% of data packets have delay under 5.5 ms

maximum node memory required:	4096 bits

2.2.5 Experiment 4a

Conditions:

voice packet size:	128 bits
calls:	100
type:	16 kbps CVSD
activity:	45% total active talk, 55% silent
data packet size:	512 bits
number processors:	1 per node
processor speed:	5 microseconds / 16 bit word
simulation duration:	250 ms, statistics collected after 50 ms
maximum data traffic:	458.04 kbps
actual data traffic:	445.31 kbps
actual voice traffic:	720 kbps
data header and control traffic:	122.46 kbps
voice header and control traffic:	239.985 kbps
utilization:	.989

Results:

This network configuration is not stable, it appears that the memory required by the node constantly increases.

2.2.6 Experiment 4b

Conditions as in experiment (4a) except:

number processors:	4 per node

Results:

total voice packet delay:	.25 ms
variance:	.01
data packet delay:	10 ms
variance:	50

95% of data packets have delay under 25 ms

maximum node memory required:	15000 bits

2.2.7 Experiment 5a

Conditions:

voice packet size:	128 bits
calls:	100
type:	16 kbps CVSD
activity:	45% total active talk, 55% silent
data packet size:	1024 bits
number processors:	1 per node
processor speed:	5 microseconds / 16 bit word
simulation duration:	500 ms, statistics collected after 50 ms
maximum data traffic:	512.42 kbps
actual data traffic:	508.59
actual voice traffic:	720 kbps
data header and control traffic:	67.68 kbps
voice header and control traffic:	239.985 kbps
utilization:	.995

Results:

This network configuration is unstable, node memory requirements do not level off.

2.2.8 Experiment 5b

Conditions as in experiment (5a) except:

number processors:	4 per node

Results:

voice packet delay:	.4 ms
variance:	.05
data packet delay:	20 ms
variance:	20

95% of data packets have delay under 30 ms
maximum node memory required: 30000 bits

2.3. Example Simulation Statistics Output

The following tables are the summary of the statistics collected from the simulation trace file. Packets are grouped into exactly similar types; for example, the first line is a summary of all type 1 (voice) packets of subtype 2, (16 kbps CVSD as defined), for calls from node 2 to node 1. The table also indicates the total number of packets rejected by the network (by node memory overflow, for example), total packet measured of this type, average end-to-end delay, in milliseconds, and variance. The last two columns indicate the average number of nodes these packets have passed through since leaving the source node, and the average length of packets of this type (the possibility of having, say, data packets of various sizes was expected).

Figure 2.1 through 2.8 depict the memory requirements as a function of simulation time. The data gives the total memory a node needs because of input/output queues and holding queues for data packets during the time intervals.

2.3.1 Experiment 1a

type	subtype	node	to node
1	1	2	1
2	1	1	2
1	1	1	2
3	0	2	1
3	0	1	2
2	1	2	1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	1784	2.354E-01	2.481E-02	1.000E+00	1.280E+02
2	0	587	5.871E-01	1.938E-01	1.000E+00	1.000E+02
1	0	1798	2.606E-01	1.142E-02	1.000E+00	1.280E+02
3	0	583	9.009E+00	3.024E+01	1.000E+00	1.280E+02

3	0	593	6.806E+00	1.489E+01	1.000E+00	1.280E+02
2	0	600	9.795E-01	8.618E-01	1.000E+00	1.000E+02

----- Statistics Summary -----

2.3.2 Experiment 1b

type	subtype	node	to	node
1	1	1		2
1	1	2		1
3	0	1		2
2	1	1		2
2	1	2		1
3	0	2		1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	1746	1.540E-01	4.124E-03	1.000E+00	1.280E+02
1	0	1738	1.627E-01	1.230E-02	1.000E+00	1.280E+02
3	0	542	8.554E+00	4.000E+01	1.000E+00	1.280E+02
2	0	533	7.875E-01	5.568E-01	1.000E+00	1.000E+02
2	0	552	6.070E-01	2.198E-01	1.000E+00	1.000E+02
3	0	527	6.901E+00	3.183E+01	1.000E+00	1.280E+02

2.3.3 Experiment 2

type	subtype	node	to	node
3	0	1		2
1	1	2		1
1	1	1		2
3	0	2		1
2	1	2		1
2	1	1		2

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
3	0	472	2.683E+00	4.138E+00	1.000E+00	1.280E+02
1	0	1728	3.022E-01	4.036E-02	1.000E+00	1.280E+02
1	0	1824	2.164E-01	3.532E-02	1.000E+00	1.280E+02
3	0	452	2.057E+00	2.463E+00	1.000E+00	1.280E+02
2	0	482	6.208E-01	2.697E-01	1.000E+00	1.000E+02
2	0	448	7.988E-01	3.977E-01	1.000E+00	1.000E+02

2.3.4 Experiment 3

type	subtype	node	to	node
1	1	2		1
1	1	1		2
3	0	1		2
2	1	2		1
3	0	2		1
2	1	1		2

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	1762	3.590E-01	2.349E-02	1.000E+00	1.280E+02
1	0	1735	5.265E-01	3.195E-02	1.000E+00	1.280E+02
3	0	861	2.378E+00	1.793E+00	1.000E+00	1.280E+02
2	0	867	2.792E-01	6.045E-02	1.000E+00	3.200E+01
3	0	795	1.922E+00	1.609E+00	1.000E+00	1.280E+02
2	0	797	2.617E-01	6.833E-02	1.000E+00	3.200E+01

2.3.5 Experiment 4a

type	subtype	node	to	node
1	1	2		1
1	1	1		2
2	1	1		2
3	0	1		2
3	0	2		1
2	1	2		1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	1480	2.199E+01	1.344E+02	1.000E+00	1.280E+02
1	0	1435	2.359E+01	1.568E+02	1.000E+00	1.280E+02
2	0	223	7.411E-01	1.804E-01	1.000E+00	1.000E+02
3	0	192	2.858E+01	1.578E+02	1.000E+00	5.120E+02
3	0	175	3.600E+01	3.987E+02	1.000E+00	5.120E+02
2	0	231	7.637E-01	2.241E-01	1.000E+00	1.000E+02

2.3.6 Experiment 4b

type	subtype	node	to	node
1	1	2		1
1	1	1		2
2	1	1		2
3	0	1		2
2	1	2		1
3	0	2		1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	1504	2.475E-01	1.152E-02	1.000E+00	1.280E+02
1	0	1460	2.441E-01	1.049E-02	1.000E+00	1.280E+02
2	0	183	5.970E-01	1.407E-01	1.000E+00	1.000E+02
3	0	192	5.027E+00	5.869E+00	1.000E+00	5.120E+02
2	0	199	6.699E-01	2.057E-01	1.000E+00	1.000E+02
3	0	181	1.360E+01	6.693E+01	1.000E+00	5.120E+02

2.3.7 Experiment 5a

type	subtype	node	to	node
1	1	1		2
1	1	2		1
2	1	1		2
3	0	1		2
3	0	2		1
2	1	2		1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	2495	7.899E+01	1.895E+03	1.000E+00	1.280E+02
1	0	2498	7.957E+01	2.010E+03	1.000E+00	1.280E+02
2	0	245	1.355E+00	4.705E-01	1.000E+00	1.000E+02
3	0	179	1.087E+02	2.971E+03	1.000E+00	1.024E+03
3	0	164	9.071E+01	2.185E+03	1.000E+00	1.024E+03
2	0	256	1.280E+00	4.630E-01	1.000E+00	1.000E+02

2.3.8 Experiment 5b

type	subtype	node	to	node
1	1	1		2
1	1	2		1
2	1	1		2

3	0	1	2
3	0	2	1
2	1	2	1

type	pkts-rej	tot-pkts	avg-del	var-del	nds-thru	avg-len
1	0	2195	4.517E-01	7.700E-02	1.000E+00	1.280E+02
1	0	2201	4.007E-01	4.048E-02	1.000E+00	1.280E+02
2	0	161	1.046E+00	5.986E-01	1.000E+00	1.000E+02
3	0	147	9.530E+00	2.584E+01	1.000E+00	1.024E+03
3	0	153	1.973E+01	1.078E+01	1.000E+00	1.024E+03
2	0	147	1.044E+00	3.894E-01	1.000E+00	1.000E+02

2.4. Conclusions

The results of the simulation experiments indicate that a packetized voice/data network is able to handle the traffic load efficiently, and with acceptable delays. With the link utilization near 1, the variance in the voice traffic delay is low enough to be comparable with the SENET scheme.

Data traffic is handled acceptably; it is clear that as data packet size increases, the net amount of real data available is increased (since there is less network control data generated). This increase in size causes an increase in the delay in the voice packets; as a longer data packet is being transmitted, the next voice packet must wait. The possibility of allowing various data packet lengths and the way the node processor handles these requires investigation.

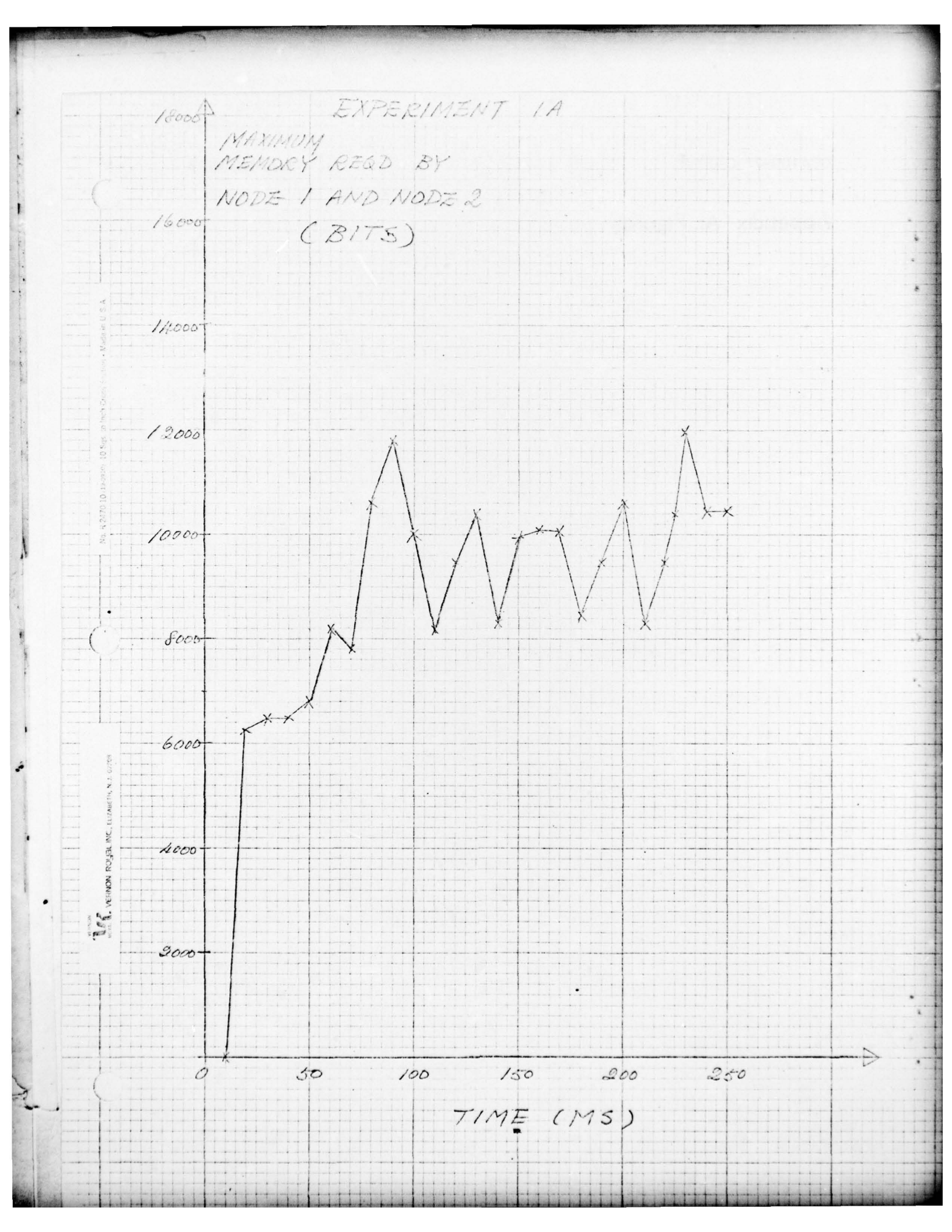
The network control protocols need to be examined in order to determine an efficient system. The effect of header and control packet size on the total usable data is clear.

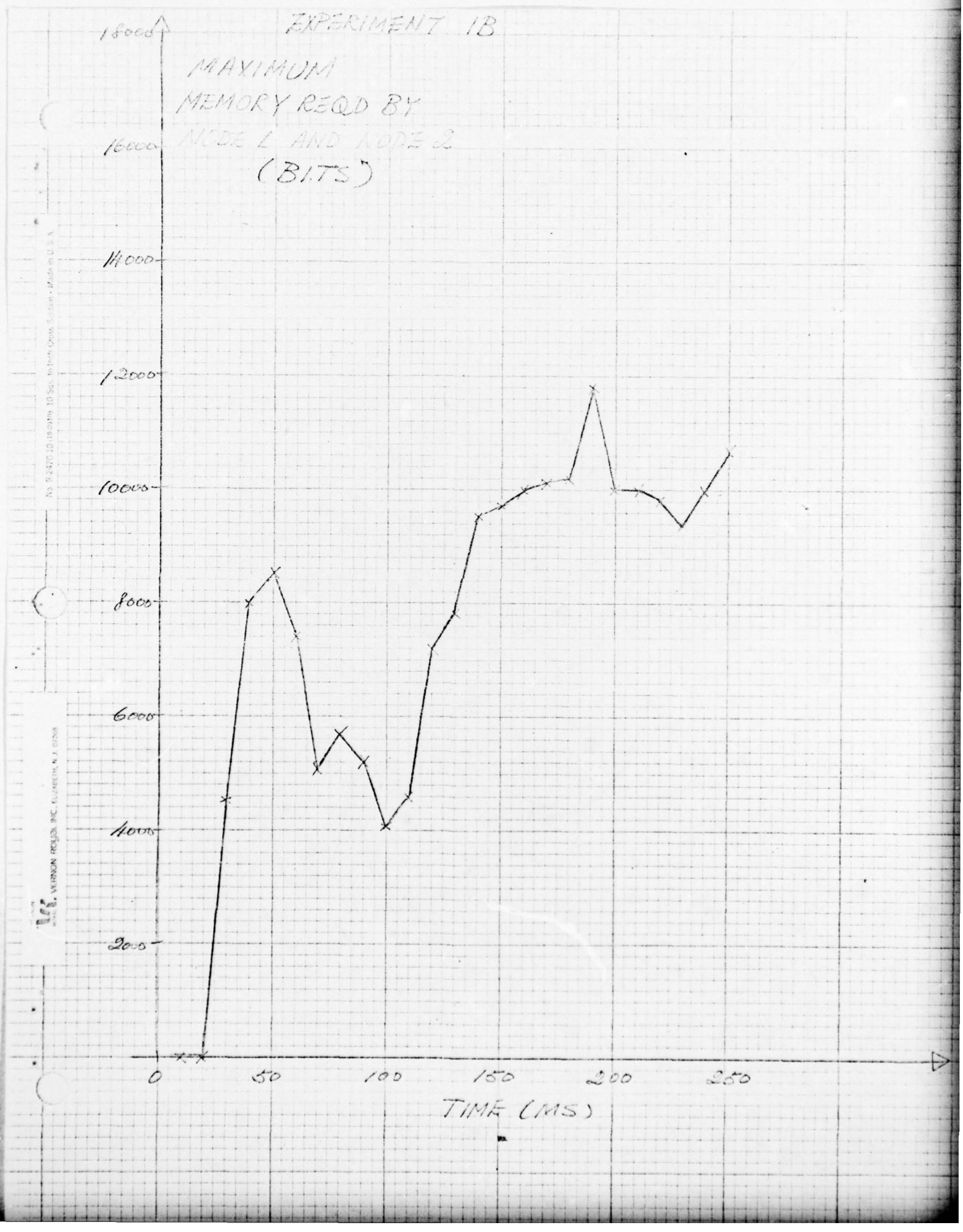
The processor capabilities within the node are critical in some experiments. Where one processor is insufficient, 4 are adequate. The network in which nodes have more

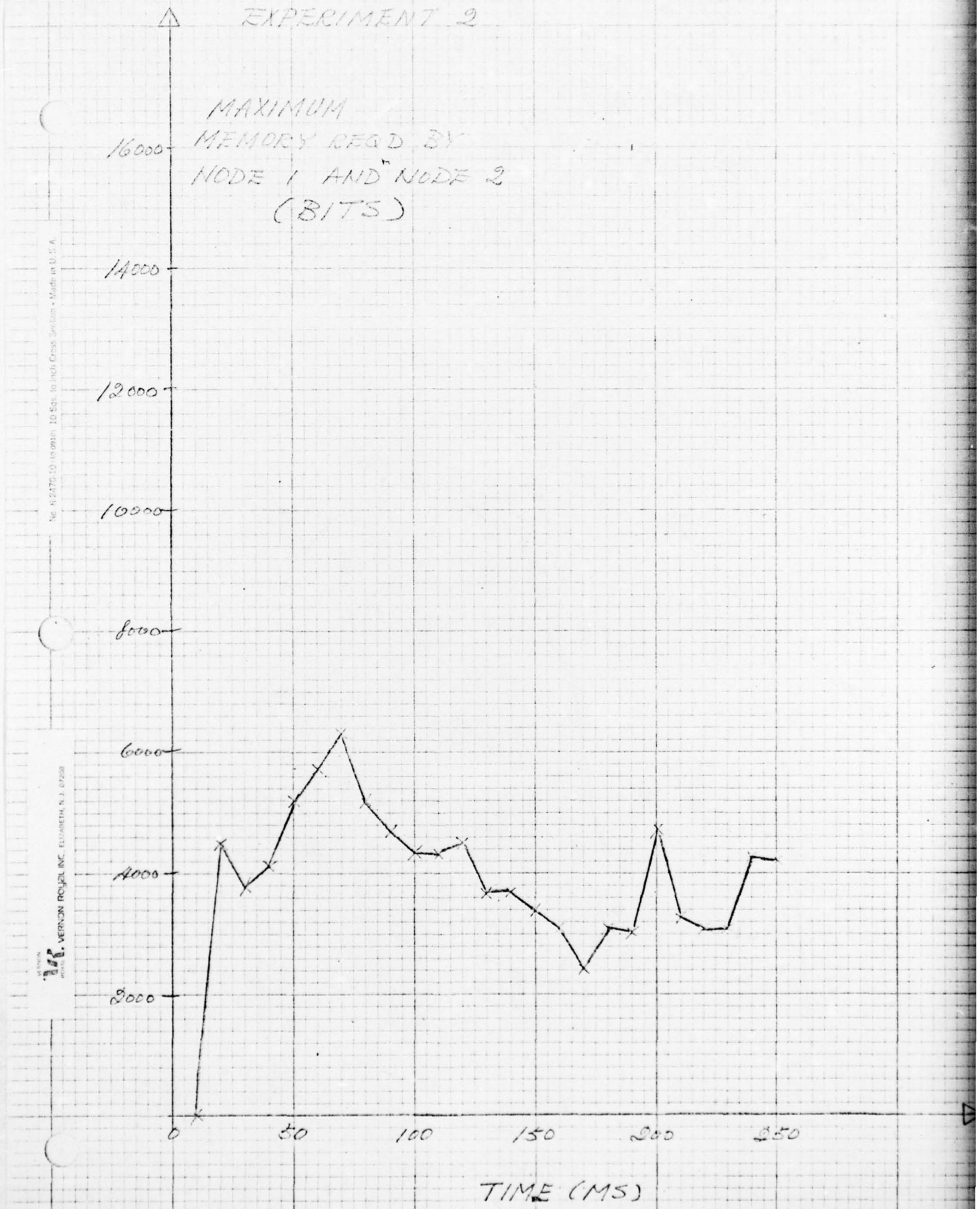
than one link would require greater processing power per node than for the network simulated here. The detailed processor requirements are not discussed here; the effect has been noticed.

A question remaining concerns the routing and packet forwarding mechanism. Whether the PVC flow control scheme is acceptable for large networks is not clear. The dynamic routing for a PVD network is not fully investigated. The possibility exists of a point in between these two which allows data traffic to be dynamically routed, while Class I traffic exists on virtual circuits. By limiting the total Class I traffic on a link to the maximum physically possible the network could accommodate instantaneous peaks without losses. Class II and III data traffic would be temporarily suspended on this link, and routed around on another path.

Appendix A: Figures

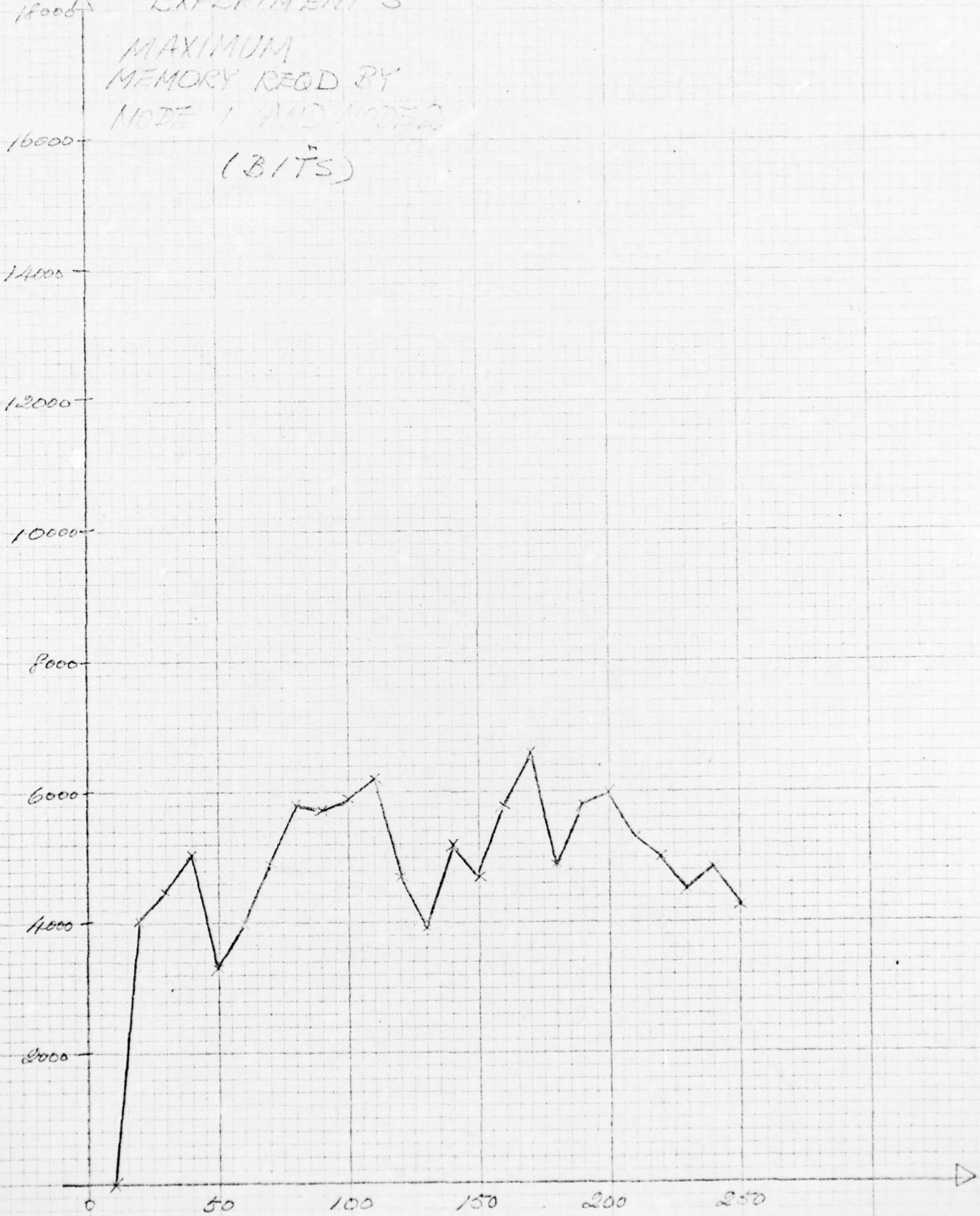

EXPERIMENT 1A
MAXIMUM MEMORY READ BY NODE 1 AND NODE 2 (BITS)
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
50
100
150
200
250
TIME (MS)

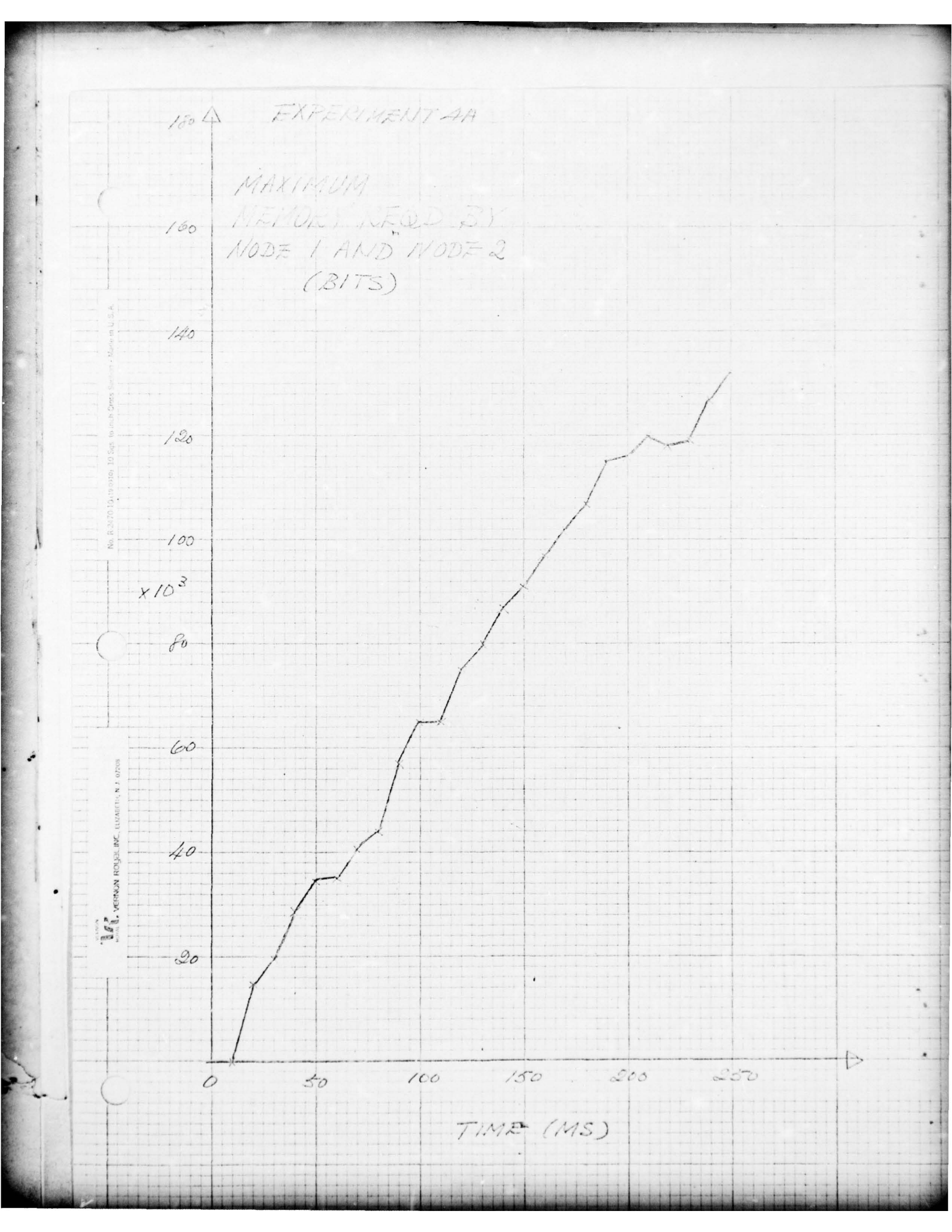

EXPERIMENT 1B
MAXIMUM
MEMORY REQD BY
NODE 1 AND NODE 2
(BITS)
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
50
100
150
200
250
TIME (MS)

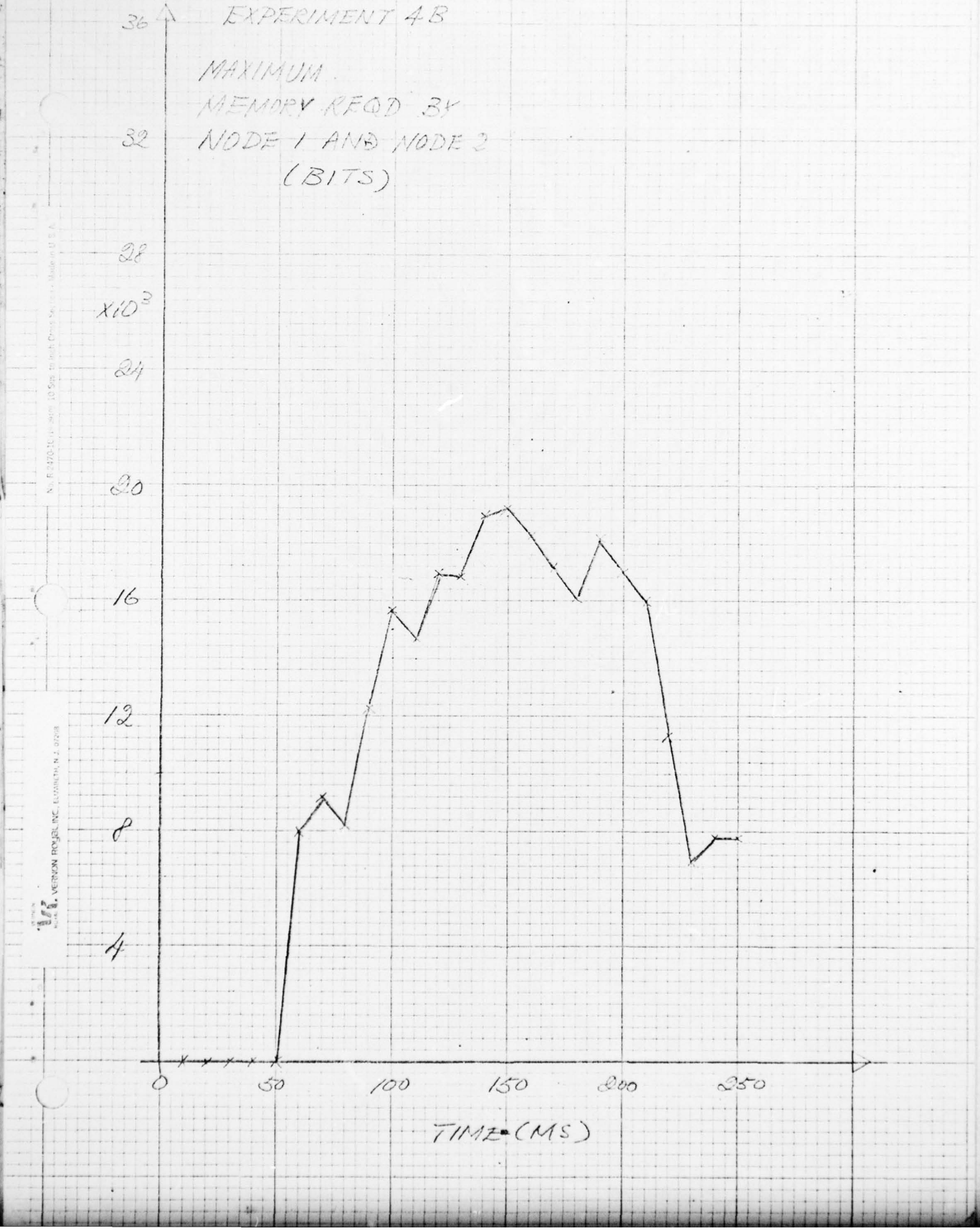

EXPERIMENT 2
MAXIMUM MEMORY REQD BY NODE 1 AND NODE 2 (BITS)
16000
14000
12000
10000
8000
6000
4000
2000
0
50
100
150
200
250
TIME (MS)

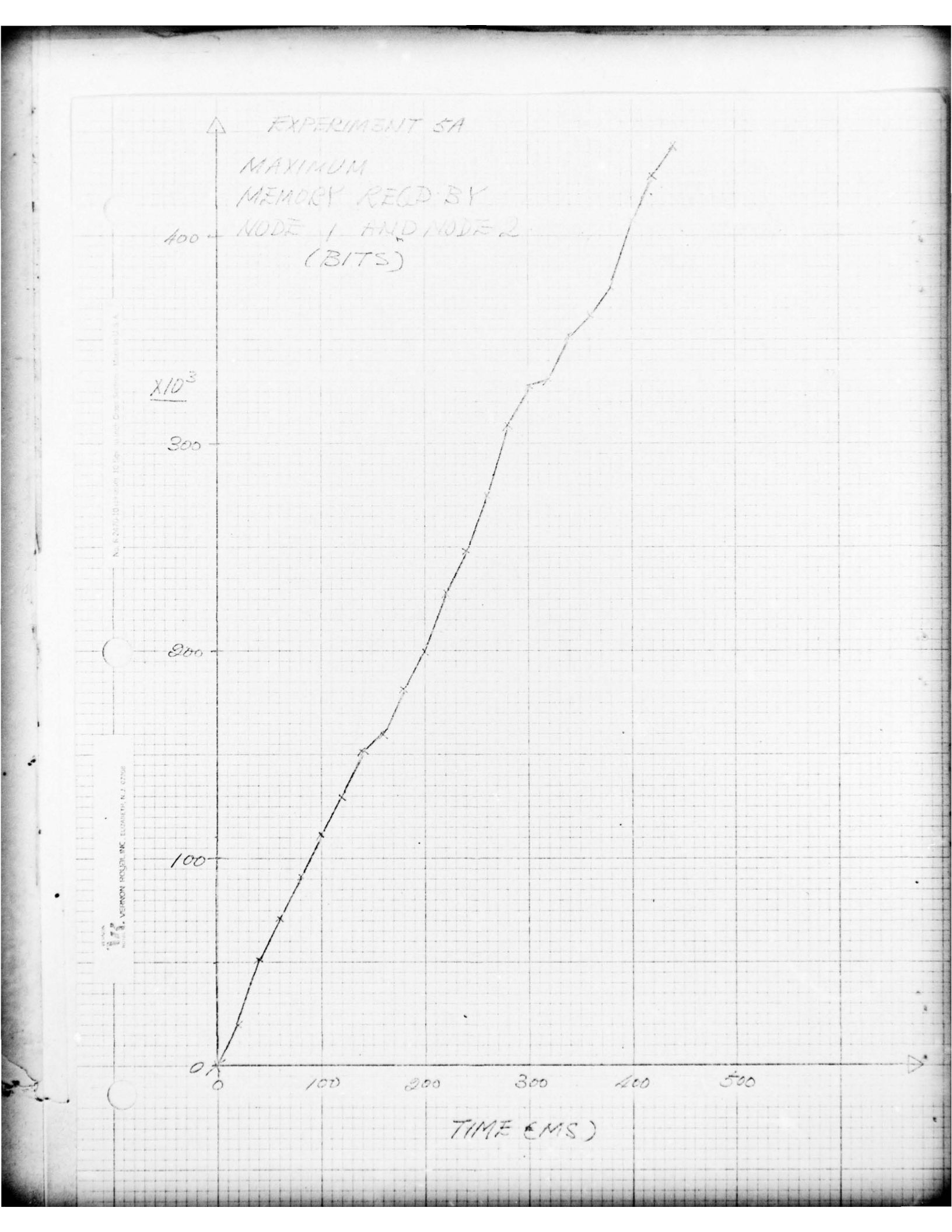
EXPERIMENT 3

MAXIMUM MEMORY READ BY NODE 1 AND NODE 2 (BITS)




EXPERIMENT AA
MAXIMUM
MEMORY USED BY
NODE 1 AND NODE 2
(BITS)
180
160
140
120
100
80
60
40
20
X10^3
0
50
100
150
200
250
TIME (MS)


EXPERIMENT 4B
MAXIMUM MEMORY REQD BY NODE 1 AND NODE 2 (BITS)
36
32
28
X10³
24
20
16
12
8
4
0
50
100
150
200
250
TIME (MS)


EXPERIMENT 5A
MAXIMUM
MEMORY REQD BY
NODE 1 AND NODE 2
(BITS)
X10^3
400
300
200
100
0
0
100
200
300
400
500
TIME (MS)

EXPERIMENT 5B

References

[ADC75] Advanced Data Communication Control Procedures (ADCCP), Independent Numbering. Proposed American National Standard. Fourth Draft, August 1975.

[Bar76a] Barbacci, M.R. and J.D. Oakley: "The Integration of Circuit and Packet Switching Networks: Toward a SENET Implementation". The 15th NBS-ACM Annual Technical Symposium, Gaithersburg, Md, June 1976.

[Bar76b] Barbacci, M.R. et al.:A Multiprocessor Implementation of an Integrated Switch. Report to the Defense Communications Agency from the Department of Computer Science, Carnegie-Mellon University. May 1976.

[Bra65] Brady, P.T.: "A Technique for Investigating On-Off Patterns of Speech". Bell System Technical Journal, Vol. XLIV, No. 1, January 1965.

[Bur72] Burton, H.O. and D.D. Sullivan: "Errors and Error Control". Proceedings of the IEEE, Vol. 60, No. 11, November 1972.

[Cov75] Coviello, G.J., and P.A. Vena: "Integration of Circuit/Packet Switching in a SENET (Slotted Envelope NETwork) Concept". National Telecommunications Conference (NTC), New Orleans, December 1975.

[Cro76] Crowther, W.R. F.E. Heart, A.A. Mckenzie, J.M. Mcquillan, D.C. Walden: "Issues in Packet Switching Network Design". AFIPS Proceedings, Vol. 44, NCC 1975.

[For75] Forgie, J.W.: "Speech Transmission in Packet Switched Store-and-Forward Networks". AFIPS Proceedings, Vol. 44, NCC 1975.

[For76] Forgie, J.W. and A.G. Nemeth: "An Efficient Packetized Voice/Data Network Using Statistical Flow Control". ?, 1976.

[Fra73] Frank, H. M. Gerla, W. Chou: "Issues in the Design of Large Distributed Computer Communication Networks". IEEE National Telecommunication Conference 1973.

[Fra75] Frank, H.: "Summary of Discussion Session". Symposium on Large Scale Networks 75', Network Analysis Corporation, Jan. 1975.

[Ger75] Gerla, M.: "Deterministic and Adaptive Routing Policies in Packet-Switching Computer Networks". Network Analysis Corporation, 3rd Data Communication 75', ACM/IEEE.

[Gor69] Gordon, G. : System Simulation, Prentice-Hall , Inc., Englewood Cliffs, N.J.

[Hea70] Heart, F.H. et al.: "The Interface Message Processors for the ARPA Computer Network,". AFIPS Proceedings, Vol. 36, SJCC 1970.

[Her76] Herrman, J.: "Flow Control in ARPA Network". Computer Networks, Vol. 1, No. 1, 1976.

[Kah71] Kahn, R.E. and W.R. Crowther, "Flow Control in a Resource Sharing Computer Network". 2nd Symposium on Problems in the Optimization of Data Communication Systems 71', ACM/IEEE.

[Kle75] Kleinrock, L. and W.E. Naylor, "On Measure Behavior of the ARPA Network". AFIPS Proceedings, Vol. 43, NCC 1974.

[Kle76] Kleinrock, L.: Queueing System, VOL. 2, Computer Application, Wiley-Interscience.

[Rob70] Roberts, L.G. and B.D. Wessler: "Computer Network Development to Achieve Resource Sharing". AFIPS Proceedings, Vol 36, SJCC 1970.

[Syl76] Sylvania Corporation: @[SENET-DAX]. Report to the Defence Communications Agency, 1976.

[Wul72] Wulf, W.A. and C.G. Bell:"C.mmp: A Multi-Mini-Processor". AFIPS Proceedings, Vol. 41,FJCC 1972