ROUTING AS A FLOW CONTROL STRATEGY WITHIN AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMMUNICATIONS NETWORK

A Dissertation

by

KENNETH RAY HEBERT

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the degree of

DOCTOR OF PHILOSOPHY

May 1986

Major Subject: Computing Science

ROUTING AS A FLOW CONTROL STRATEGY WITHIN AN INTEGRATED CIRCUIT/PACKET-SWITCHED COMMUNICATIONS NETWORK

A Dissertation

by

KENNETH RAY HEBERT

Approved as to style and content by:

Udo W Pooch
Udo Pooch
(Chairman)

Guy L. Curry
Guy Curry
(Member)

William Lively
William Lively
(Member)

Sallie Sheppard
Sallie Sheppard
(Member)

Bruce A. McCormick
Bruce McCormick
(Head of Department)

May 1986

ABSTRACT

Routing as a Flow Control Strategy
Within an
Integrated Circuit/Packet-Switched
Communications Network. (May 1986)
Kenneth Ray Hebert, B.S., University of Southwestern
Louisiana;
M.S., University of Southwestern Louisiana
Chairman of Advisory Committee: Dr. Udo Pooch

The introduction of economical voice digitization techniques and dramatic increases in the speed of electronic switching systems have now made possible the integration of voice and data traffic over the same communications network. Known as an "integrated" network, such a combined facility possesses characteristic advantages which make this technology the wisest choice for future communications systems. Recognizing this, researchers are pursuing design questions relevant to the integrated environment. Analysis of alternative architectures has not only convinced many of these researchers that integration of voice and data is a viable technique, but has also caused them to conclude that the Slotted Envelope Network (SENET) system described by Coviello and Vena is one of the most promising

alternatives offered thus far.

This research accepts the premise that the SENET-type integrated environment will eventually become commonplace. Leaping forward into that timeframe, however, one finds that many of the questions central to effective management of scarce communications resources remain unanswered. One of these is how to reduce congestion in this type of environment.

Focusing on that question, this effort extends current research by investigating the viability of routing strategy as a mechanism for reducing congestion in the integrated communications environment. In doing so, it exploits the notion that congestion occurs when an excessive volume of traffic is channelled down a given communications link.

A FORTRAN-based simulation of the integrated SENET environment is developed and then used to examine the characteristics of seven alternative routing strategies. The performance of these strategies is compared both amongst themselves and with "optimality". Performance is measured in terms of user-visible metrics.

Based on the experimental results obtained, this research concludes that fixed routing, the technique assumed by most models examined in the literature, is not as effective a tool for reducing congestions as would be a

strategy based on link utilization. The research further concludes that minimization of congestion can be realized only if the routing strategy is adjusted as workload varies. Detailed experimental data supporting these conclusions is presented. Finally, the dissertation concludes with an outline of related topics which merit further attention.

DEDICATION

This document is dedicated to the two people whose support made it possible, my wife, Pat, and son, Doug.

ACKNOWLEDGEMENTS

It is impossible to properly thank the many people who have touched my life over the last four years. To all of them I say "Thank you" and trust that each know it is truly heartfelt. There are, however, certain people without which this effort would not have come to fruition and whom deserve special thanks.

Dr. Udo Pooch, my major professor, who was always there when I needed direction and support but was astute enough to recognize my need for independence.

Colonel Robert Feingold, the man who made this effort possible and who has provided me with the best example of character and personal integrity I could have hoped for.

Michael and Carol Masser, two close friends who provided the sanity checks so often needed along the way.

Pat and Doug Hebert, my wife and son, who were always there to provide the support I needed and who kept me in touch with reality.

To these, and to the ones unmentioned but remembered, I say "Thank you".

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Computer-Communications Networks

A computer-communications network can be described as an interconnected group of independent computer systems capable of communicating with one another [16]. This definition is extremely broad and includes equipment configurations which run the gambit from pairs of micro-computers that occasionally exchange data to airline reservation systems linking thousands of real-time computers and terminals. It includes configurations where all components are located in the same room and those where the components are spread out around the world. The definition encompasses both systems designed for leisure and systems which transmit time-critical military command and control information.

In spite of their many varieties and diverse applications, all computer-communication networks possess two characteristic features. First, the component computer systems communicate with each other and second, the systems are independent.

The Communications of the ACM has been used as a pattern for both format and style.

Media commonly used to accomplish computer communications include twisted wire pairs, coaxial cable, microwave links, optical fiber and satellites. The combination of transmission media, the nodes which they connect and the necessary hardware/software needed to control them is commonly referred to as the "communications subnet" or the "data network" [88]. "Communications backbone" is another term sometimes used to refer to this combination [64].

The second characteristic feature of a computer-communications network, independence, is particularly significant. This characteristic not only distinguishes computer-communications networks from multiprocessor computing environments, but also has important implications concerning the volume and timing requirements of data flows between the component computer systems. Specifically, the cooperative nature of the multiprocessor environment would typically force a much larger flow of data to support task/job coordination. Further, depending on the extent of processor synchronization, the multiprocessor environment might require message response times measured in milliseconds or even microseconds.

Some authors have found it necessary to draw a distinction between the terms "computer-communications network" and "computer network." Elovitz and Heitmeyer

[30] distinguish the two by noting that in the former the user is responsible for managing the computer resources, while in the latter the resources are managed automatically by a network operating system. Since this research will focus on the management decisions which govern the network and not their origin, such distinctions become irrelevant and the above two terms will be used interchangeably along with the terms "communications network" and "network." For the purposes of this dissertation these terms will be taken to mean the communications subnet and all facilities necessary to gain access to it.

1.2 Historical Development

The last two decades have witnessed three developments destined to drastically change the telecommunications environment in this country. The first was political in nature and has been called the Carterphone Decision [74]. This Federal Communications Commission (FCC) ruling gave both common and non-common carriers the right to freely connect third party devices to common carrier lines, thereby dramatically expanding the types of services carriers could offer their customers [60, 118]. One such service was computer support with carriers immediately recognizing the potential for offering shared time on computer systems and specialized

peripherals to geographically dispersed users via common carrier lines [5, 32].

The second was an almost unbelievable drop in the cost of computers. Prior to 1970, the dominant model for computer support was a central computer supporting dispersed terminals. This configuration was motivated by the high cost of computer equipment and the relatively low cost of communications. However, since 1970, falling computer prices have caused designers to re-examine that common configuration. Medium, large and now very-large scale integration technologies have reshaped the economics of computer support, providing powerful desktop computer systems which often cost less than one tenth of the persons using them. Technological advances, making computers such inexpensive tools, have forced the old configuration to yield to a configuration where large numbers of computers are used, and are often essential to, the day-to-day operations of modern business. To be most effective, these computers (often geographically separate) must exchange data either periodically or continuously. Thus, the last few years have seen computer communications become an operational necessity for both industry and government [67, 111].

The third major development, also technological in nature, was the development of ARPANET, considered by many to be the father of modern computer networks [108]. Named

for its sponsor, the Defense Advanced Research Projects Agency (DARPA), ARPANET differed from previous communications networks in that it used a technique called "packet switching" to transfer data from one location to another. In the packet switching environment, transmissions between two locations are broken into small, typically fixed sized segments (1000 bits for ARPANET) called packets. Each packet is then individually routed to its destination where the packets are reassembled to form the original message. ARPANET has grown from its original 4-node experimental implementation to a mature operational network with well over 200 nodes scattered across three continents [2, 108, 111]. Today, ARPANET allows data and specialized equipment to be shared among hundreds of universities, research centers and government agencies. ARPANET has proved that packet switching is an important technology that is superior to circuit switching technology for certain classes of transmissions [41, 92]. Circuit switching is the older of the two techniques, developed to service voice calls when the telephone network was first implemented. It is used by all computer-communications networks which pre-date ARPANET. In the circuit switched environment, a complete path is established between the source and destination nodes for their entire session. This path is used to carry all transactions between the communicating nodes.

1.3 The Case for an Integrated Network

Experimentation with ARPANET and the many networks which have followed yielded a wealth of data about computer communications. In particular, packet switching has clearly demonstrated its advantages over circuit switching for the bursty traffic typical of human-computer interactions in a real-time environment [88]. Circuit switching has not been relegated to history, however, since it still seems to be the methodology best suited for high-data rate, steady-flow applications [12, 17, 18, 63, 111, 120, 122].

Current communications systems are generally designed to handle either voice or data traffic, but not both. Operational systems typically use separate facilities for these two traffic classes. Clearly this approach represents a duplication of effort and resources. In the early-1970's it was recognized that an integration of voice and data traffic into the same system would reap significant benefits from economies of scale, commonality of equipment and more efficient utilization of bandwidth [42, 52, 59, 61, 72, 75, 93, 95, 103].

The clear advantages of an integrated environment plus advances in technology which eliminate barriers to implementation of such a network have prompted many researchers to study techniques for implementing an integrated circuit/packet switched communications network.

Analyses of alternative architectures have been extensively reported in the literature [8, 23, 24, 28, 29, 33, 43, 44, 53, 55, 58, 81, 84, 94, 97, 99, 110, 114].

One of the most technically viable proposals appears to be the Slotted Envelope Network (SENET) proposed by Coviello and Vena [24]. The viability of an integrated network has been demonstrated by several authors and it is the general consensus in both industry and government that the computer-communications networks of the future will integrate both packet and circuit switching techniques [9, 13, 21, 31, 34, 54, 73, 83, 95, 96, 100, 111].

1.4 The Next Step

Researchers have demonstrated both the wisdom and feasibility of pursuing a SENET-type communications network. They have also investigated the critical issues of optimal topological design and link capacity assignment. Thus, previous research provides the tools necessary to design and implement an integrated circuit/packet switched communications network [4, 11, 15, 19, 25, 38, 40, 64, 68, 70, 71, 95, 117].

Once built, such a network will have a built-in limit to the amount of traffic it can carry. If demand should ever exceed this limit, data packets and/or voice calls will be delayed or rejected. Any network experiencing this condition is said to be "congested" [80, 104].

Clearly, congestion could be eliminated if enough resources were dedicated to the communications network. Such a network would, however, be prohibitively expensive. Realistically, computer-communications networks must be designed to accommodate peak traffic requirements and absorb reasonable load fluctuations, but must do so within the constraints of cost-effective operation. However, even in networks designed with substantial excess capacity, higher than expected traffic demands, unfavorable load patterns, component failures or any combination thereof, can create situations where the network becomes congested. Whatever the reason, congestion is clearly unacceptable and procedures must be developed to prevent or at least forestall this condition. These procedures are generally referred to as "flow control strategies" [56, 104].

This research investigates the utility of routing as a mechanism for implementing the network flow control strategy. Routing policy determines the path information (either voice or data) will follow in traversing the network. This research effort exploits the notion that congestion results when an excessive volume of traffic is channeled down a given communications link. The research will determine whether certain routing strategies are superior to others in preventing and/or forestalling congestion in a SENET-type network environment.

1.5 Research Objectives

The specific objectives of this research are to:

(1) Investigate existing routing strategies available for use in a circuit/packet switched communications network.

(2) Identify those routing strategies which will maximize throughput and reduce congestion under varying traffic conditions and response constraints.

(3) Simulate an implementation of routing which will demonstrate improved throughput and hold congestion to a minimum.

This dissertation summarizes the results of the stated objectives.

A survey of the relevent literature is presented in Chapter II. It serves to highlight applicable research and to provide the reader with an appropriate background. The experimental design is described, in detail, in Chapter III. The simulation model used to accomplish the stated objectives is described in the next chapter, while the results of the simulation experiments and analysis of the obtained results are given in Chapter V. Finally, in Chapter VI, the conclusions drawn from the simulation results are presented and recommendations for further study provided.

CHAPTER II

LITERATURE SURVEY

2.1 Overview

Technology has advanced to the point where it is now economically feasible to integrate voice and data communications over the same network. This research investigates questions relevant to that type of environment. Chapter II sets the stage for a discussion of the research and its conclusions by providing the reader with a review of previous research and necessary background material. The chapter begins with a description of the three communications technologies in widespread use today. Then a discussion of why an integrated network is both technically feasible and economically desirable is presented. SENET, one of the prime architectures for integrating voice and data traffic, is described. Finally it addresses the role of flow control in a computer-communications network and summarizes the principal routing strategies available, setting the stage for a discussion of the experimental effort.

2.2 Switching Technologies

In order for two nodes to communicate, three distinct steps are necessary. First, a physical path must be

established between the two; second, information must be transferred; and third, the path must be disconnected. The process of establishing and disconnecting the physical path is known as switching.

The three switching technologies currently in use can be distingushed by the methodology used to allocate and deallocate network resources [65]. The three technologies are:

(1) circuit switching,

(2) message switching, and

(3) packet switching

The following paragraphs describe each of these techniques in some detail.

2.2.1 Circuit Switching [51, 65]

In a circuit switched network, resources are allocated at the beginning of a user's session and reserved throughout its duration. When a request for service is received, a physical path (circuit) is located which connects the source and destination nodes. This path is retained until all communication is completed.

This arrangement, motivated by the telephone network, uses circuit switching exclusively. When the telephone plant was first installed the only way to transmit voice was via analog signals, which are not easily stored. This, plus slow switching times, forced design engineers

to use circuit switching techniques.

This system has both advantages and disadvantages. Clearly, it minimizes the effort needed to establish and disconnect the circuit, an operation which occurs only once for each session. Its biggest drawback is that expensive communications resources are tied up regardless of whether they are needed. During normal conversations each telephone link is idle, on average, fifty percent of the time. When this lack of utilizatation is combined with "idle periods," the result is a significant amount of wasted capacity. Some researchers have estimated this unused capacity as high as 60% [51]. Reservation prevents communications links from being used by other traffic during periods of inactivity.

The utilization problem is amplified when computers enter into the picture. Because computer communications are normally "bursty," i.e., the proportion of idle to busy time was much higher than that of voice traffic, the amount of wasted capacity increased dramatically.

At this juncture it is important to note that circuit switching does not have any inherent inefficiencies. Given the right application, namely a steady flow of data or voice traffic, it is the most efficient of the three switching techniques available. Circuit switching advantages include:

(1) Resources to establish and disestablish circuits are minimal.

(2) Traffic is delayed only by the time to propogate through the communication subnet. There are no additive accounting or switching delays.

(3) Only information (voice or data) need be transferred. There is no need for traffic management information.

(4) Congestion is impossible because there is only one user trying to tranfer data over each half duplex (HDX) line.

2.2.2 Message Switching [50, 85]

Message switching was proposed as a refinement to circuit-switching. Message switched networks address the inefficiencies of the first approach by dividing the user session into "messages," where each message is a logical unit of information from the viewpoint of the user [65]. Conceptually, messages should correlate with the "bursts" of data typically found in computer communications. Since a "message" is defined from the user's perspective it can vary in size from one bit to an entire textbook, depending on the application.

The principal idea behind this technique calls for each "message" to be transmitted individually, i.e.,

treated as a complete session. This implies that paths are to be established and disestablished for each message. This workload, plus the accounting necessary to keep user sessions together, creates a significant burden on the network switching systems. In return, however, costly network resources are no longer allocated for the entire user session, and can be shared by many users.

Because of variability in the size and number of messages to be routed over a given path, message switching can not guarantee that a path will be available when needed and messages may have to be stored at their origin. Variants of this technique, called store-and-forward systems, allows messages to be sent as far along their desired route as possible. If a desired link is already being used, the message is stored at the intermediate node until the desired link becomes free.

Message switching is clearly a tradeoff technique. In exchange for not holding network resources over (possibly long) idle periods, message switching gives up guaranteed message arrival times and adds a significant workload in terms of both switching and session management. It also opens the door to network congestion by allowing multiple users to compete from the same communications links. Finally, it artifically increases message delay since switching time, possibly as high as ten seconds [111], is added to each message.

2.2.3 Packet Switching [65, 88, 111]

Packet switching is the newest of the three available switching technologies. Introduced in the late 1960's and motivated by the Department of Defense and its Advanced Research Projects Agency, packet switching quickly gained acceptance.

Conceptually, packet switching is a further refinement of message switching. It attempts to address the problems associated with variable message size by arbitrarily breaking each message into smaller, equal-sized "packets". One of the best known computer-communications networks, ARPANET, uses this technique, dividing messages into 1000-bit packets. By breaking messages down even further, packet switched networks add an additional layer of overhead to the data transfer system since individual packets must be coordinated and reassembled into their original message format. The packet approach also dramatically increases the number of paths to be established and disestablished since this task must be accomplished for each packet.

In spite of the extra workload generated, packet switching, has been shown to be superior to both circuit and message switching for handling bursty traffic, especially in those applications where transmission errors are common [91].

2.3 The Case for an Integrated Environment

Until recently technological barriers have forced voice networks to be distinct from those designed to handle data traffic. The voice networks, using circuit switching, often paralleled those carrying data via message or packet switching, neither system capable of accommodating the workload of the other. Obviously this represented an inefficient use of resources. Gruber [47] stated:

> "Perhaps the ultimate objectives of integration are ... to realize the economies of equipment commonality, ..., higher resource utilization and combined network operations, maintenance and administration."

Technological advances on two fronts have now made integration a realistic alternative. First, economical voice encoders (vocoders) have been developed which allow voice traffic to be digitized. This is typically accomplished by using Pulse Code Modulation (PCM) to convert voice samples into binary codes [48, 76, 82, 87, 101]. In this form, voice can be transmitted over the Bell System's T-carrier family of digital links. The second development, switching equipment which allow nanosecond switching times, allows lines to be shared among users without making conversations discontinuous [7,

87]. Taken together, these advances make it possible to integrate data and voice traffic into the same network. According to Dysert et al [29]:

> "The future for fully integrated voice and data transmissions is very promising."

As evidence, consider recent Defense Communications Agency studies which show that the Department of Defense intends to implement an all-digital, integrated network that would be operational early in the next decade [8, 23, 72].

2.4 SENET - An Integrated Environment

Researchers have examined numerous scenarios for integrating voice and data. Of those examined, the one which appears to be the most technically viable is the Slotted Envelope Network (SENET), proposed by Vena and Coviello [61, 81].

SENET integrates voice and data traffic over the same network by packetizing all traffic. Voice is packetized via vocoders while data is accommodated through standard packet techniques. In the SENET-approach, a digital link is divided into fixed duration envelopes, each of which can accommodate a constant number of slots. The slots are partitioned into three traffic classes. The first partition is reserved for voice traffic, while the second

is used to carry data packets. The final partition is the unused capacity which results when there is insufficient data and/or voice traffic to fill the envelope. Envelopes begin with a header field to aid in synchronization and finish with a trailer field which is used to mark the end of the envelope. Either of these two fields might contain network management information. Figure 2-1 provides a conceptual view of the SENET approach. In this abstraction, envelopes are represented by three "wheels" each symbolizing a SENET partition and each subdivided into the number of "wedges" needed to accommodate the available traffic.

2.5 Flow Control Strategies

From a user perspective, the computer-communications network is a "black box," to satisfy the needs for transmitting information. Other than that, it serves no useful purpose. Consequently, the user evaluates the network in just those terms. Either it satisfies the requirement to communicate within an acceptable time frame or it does not. If the "black box" does not respond in the expected manner, it is labeled "congested". From this vantage point, "flow control strategies" would be defined as those procedures which are used to prevent or minimize congestion.

SENET Structure

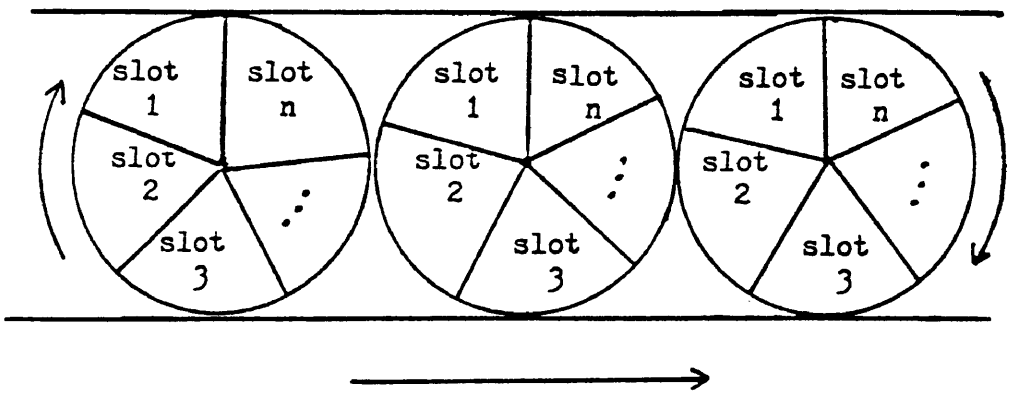


Figure 2-1

From a network design perspective, flow control has grown to take on a slightly different meaning, with flow contol being distinguished from "congestion control." The former is normally considered to be more inclusive since it would include those procedures/techniques aimed at insuring that offered traffic successfully traverses the network, and not simply those strategies which interact when a problem occurs.

This research proposes the idea that these concepts are essentially the same, differing only in viewpoint. The user viewpoint is motivated by the fact that flow controls are unimportant until they become either (a) restrictive or (b) ineffective. The second viewpoint is motivated by the fact that network designers and engineers use controls to prevent problems from occurring.

In view of this, it is appropriate to accept the first definition, i.e. flow control strategies are those procedures designed to prevent or forestall congestion. However, it must be clearly understood that this definition includes all procedures designed to insure the successful traversal of the network by offered traffic, since when these fail, congestion inevitably results.

Experience with ARPANET and other packet switched networks has shown that flow contol is a complex subject which can only be successfully addressed by a layered approach [3, 6, 39, 102, 106, 113, 115]. These layers

must work together in a synergistic fashion if problems are to be avoided. Gerla [39] identifies the four levels of flow control shown in Figure 2-2.

The first level, node-to-node, is the most basic. It involves the coordination of message traffic between two adjacent nodes of the communications subnet. Its objective is the prevention of buffer congestion and nodal deadlocks.

The second layer, source-node-to-destination-node, is an end-to-end approach to flow control. Its objective is to insure that traffic entering a source node is successfully transported to a destination node.

The third layer, host-to-node, deals with those controls necessary to insure that traffic is successfully transferred from the host (user) to the network access point (node). The final level, host-to-host, encompasses protocols between network users to insure the safe transmission of traffic.

Routing falls into the second category of flow control techniques. When used as a flow control strategy the objective of routing becomes the direction of traffic so congested links or nodes are avoided. Routing is unique among other flow control mechanisms in that it is not traffic constraining. Other mechanisms try to resolve congestion by reducing the flow of traffic between nodes and/or users. Routing attempts to alleviate congestion by

Flow Control Levels

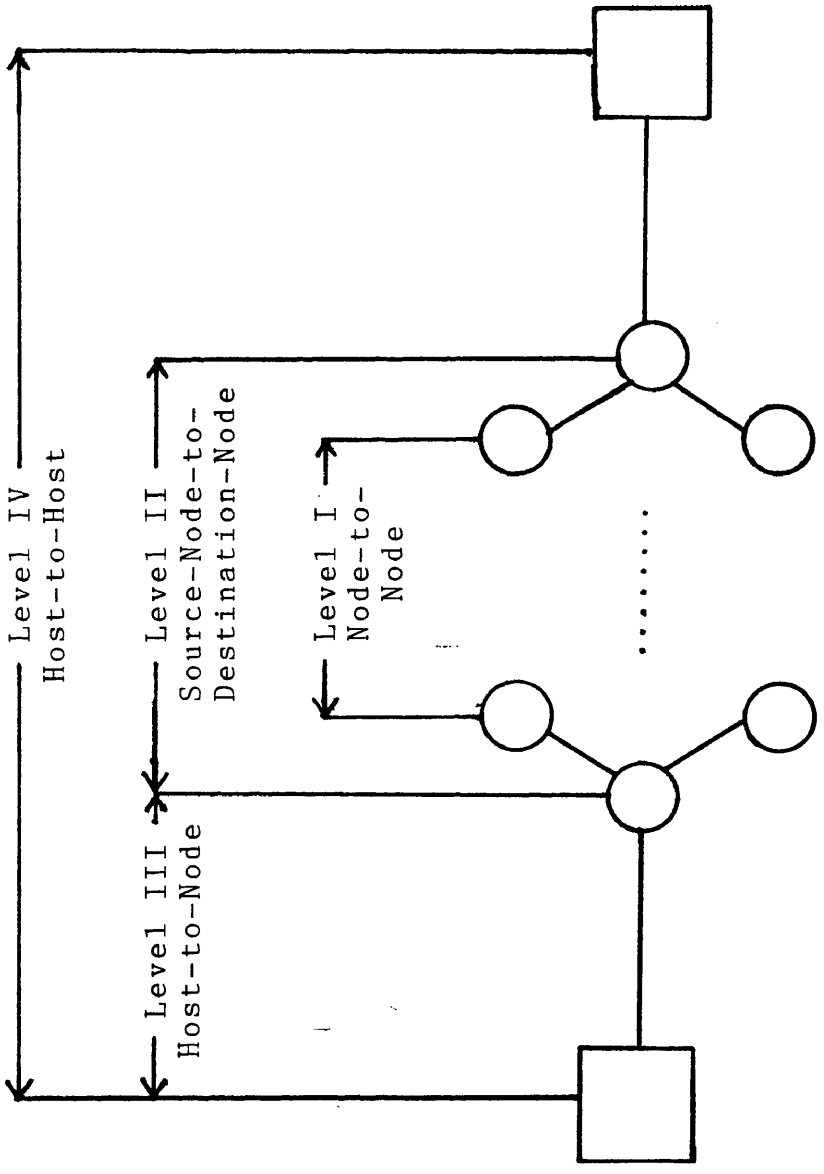


Figure 2-2.

redirecting traffic to less frequently used paths. This is sometimes referred to as "load leveling."

2.6 Routing Classifications [1, 20, 37, 69, 98, 105, 121]

A great deal of effort has gone into the design and modeling of routing algorithms. Some of the authors to have considered this subject are Prosser [89, 90], Boehm and Mobley [10], Metcalfe [79], McQuillan [77], Kleinrock [66], and Greene [45]. Fultz [35] provides an excellent classification scheme. The descriptions that follow are based primarily on his research, as summarized by Greene [45]. The classification scheme (Figure 2-3) broadly categorizes routing strategies into deterministic and stochastic algorithms.

2.6.1 Deterministic Algorithms

Deterministic algorithms derive routes based on a predefined set of static rules. This category, the simplest to implement, requires no information on the state of the network. Four basic strategies are described.

2.6.1.1 Flooding

Flooding is the simplest of all deterministic routing strategies [10]. It calls for all nodes to blindly transmit incoming messages over <u>all</u> outgoing

Routing Classifications

Class	Algorithm	Variant
Deterministic	Flooding	All Selective
	Fixed Routing Split Traffic Ideal Observer	
Stochastic	Random Routing	
	Isolated Routing	Local Delay Estimate Shortest Queue+Bias
	Distributed Routing	Periodic Update Asynchronous Update

Figure 2-3

links. Some variations of this strategy call for the link over which the message arrived to be exempted while others call for flooding to occur over only a portion of the outgoing links. The latter technique is referred to as "Selective Flooding." Under all of these variants, messages are removed from the system after some time quantum based on the assumption that at least one of the duplicate copies must have reached its destination.

Flooding has been proposed by some authors as a solution to information transfer problems in a hostile environment, e.g. military command and control information under wartime conditions [45]. It also has been proposed as a way to identify the shortest path to a destination node since flooding all paths guarantees that the shortest path will be selected. However, the duplication of traffic inherent in the strategy causes flooding to be quickly overcome by congestion, making it useless as a practical approach to routing.

2.6.1.2 Fixed Techniques

This approach to routing (called deterministic by some authors) calls for messages to follow a predetermined path between source and destination nodes. The predetermined routes, usually stored in tables at each node, are based upon an assumed network configuration and known traffic patterns. This allows the definition of

optimal routes to be reduced to a multi-commodity flow problem, solutions to which are well known [36, 86].

This approach works well as long as the configuration and traffic assumptions remain valid. Unfortunately, this strategy is unable to adapt itself to changes in the network or its offered workload.

Some authors have proposed a tiered approach whereby alternative tables are made available. When the primary route is busy/unavailable, the alternative route is selected. Though better than the single table scenario, this approach suffers from two significant weaknesses. First, the "fallback positions" are usually limited to one or possibly two backup tables. Thus, even the tiered approach is limited in its ability to adapt to changes in network status. The second drawback is that the assumptions for making the backup routes optimal may not be valid and could cause more problems than they solve. Since a node receives no information regarding network status, it has no way of determining whether the secondary or tertiary route is superior to the primary and must base the alternative path selection on nonavailability of primary routes.

2.6.1.3 Split Traffic Techniques

Split traffic techniques, also called traffic bifurcation techniques, are similar to the fixed

techniques just discussed in that alternate tables exist. The difference between the two techniques lies in the fact that all tables are used on a continuous basis. Tables are selected randomly with each table having a predefined probability of being selected. The selection probabilities are established ahead of time based on experience. Compared to fixed strategies, split techniques typically maintain a better balance of traffic throughout the network. However, the technique suffers from the same two drawbacks noted for the fixed techniques. Further, random selection makes performance heavily dependent on the probabilities selected.

2.6.1.4 Ideal Observer

This approach envisions a network overseer which has total knowledge of the entire network and its current status and workload. Each time a message enters a node, its route is recomputed to determine the optimal path to its destination. Investigated as an "ideal" situation, it is used primarily as a standard against which to compare other techniques.

2.6.2 Stochastic Algorithms

Stochastic algorithms operate on probabilistic decision rules rather than deterministic ones. Routes are selected based on an evaluation of network status

information. Some of the network measures used include current configuration, previous delay metrics and current workload. These techniques appear to have the greatest potential for accommodating network instabilities since they try to assess network status when determining routes. As a class, these techniques suffer from the overhead problems associated with exchanging status information on a timely basis. This becomes less of a problem with Common Channel Interswitch Signaling (CCIS) which provides for the constant exchange of network status and control information. This capability is rapidly being added to the current telecommunication system. A CCIS system could be implemented in a SENET-type environment by reserving the first few slots of the each envelope for information exchange [26, 45, 62, 107, 109].

2.6.2.1 Random Techniques

Random routing techniques assume that the node transmitting a message knows of only its own existence. Messages are sent out over a randomly selected link, following what Davies and Barber [27] call a "drunkard's walk." The central idea behind this approach is that random selections will eventually cause the message to arrive at its destination. Variants of this policy include incorporation of a "bias" to guide the message in the general direction of its destination and end-to-end

approaches where one of the multiple paths available from the source to the destination are selected. Random approaches suffer from inefficient path selection but are beneficial in networks where there is a high probability of node or link failure [10, 14, 89].

2.6.2.2 Isolated Techniques

Sometimes referred to as "Backward Learning techniques," isolated techniques are the local analog of the ideal observer strategy. Rather than rely on the network overseer, isolated techniques rely on information available to them locally, either from observed message delays or from local metrics such as queue length. As information is gathered, it is used to update a routing table which is then used to select paths for outgoing traffic [10].

The principle advantage and disadvantage of this approach stem from its local nature. Because all information is locally observed, there is no network overhead associated with information transfer. Conversely, the local visibility implies that adjustments cannot usually be made to non-local network failures. For example, assume node A desires to transmit data to node C. Its current routing table indicates the optimum path is via node B, therefore packets are sent to that node first. If the link from node B to C is out of order, node A would

have no way of knowing and data would simply stack up at node B.

2.6.2.3 Distibuted Techniques

Distibuted techniques are also related to the ideal observer technique discussed earlier. These techniques base routing decisions on either periodically or continuously exchanged data designed to reflect the status of the network. Types of data which might be exchanged include queue lengths, delay metrics, link and node status, and link utilization information.

When first devised this class of techniques was considered to involve too much overhead to be useful. Several authors have proposed variants which yield a compromise between the burden of information broadcasts and the weaknesses of isolated strategies. Fultz [35] proposed the use of a "minimum delay vector" which is continuously exchanged between adjacent nodes. McQuillan [78] proposed a partitioned approach where status information is only exchanged between small groups of nodes, with information routed between the groups as if they were single units.

This approach becomes more viable with the incorporation of CCIS techniques. By providing a convenient mechanism for distributing traffic management information, CCIS forces a re-evaluation of routing

strategies previously investigated and the tradeoffs characteristic of them.

2.7 Summary

As the above review demonstrates, routing strategy has been the subject of intense interest. A great deal has been learned about the tradeoffs involved in devising a routing strategy. Unfortunately, previous work has been focused primarily on the pure packet environment. This research extends these studies into the integrated environment where the added complexities of combined voice and data traffic make direct application of earlier results questionable. So far, research in the integrated environment has been limited to fixed routing strategies.

CHAPTER III

EXPERIMENTAL DESIGN

3.1 The Research Hypothesis

The literature survey in Chapter II clearly demonstrates both the wisdom and technical feasibility of constructing an integrated circuit/packet switched computer communications network. As a result of the research effort previously expended, network designers have the tools and techniques needed to build such a network. The dawning capability to build an integrated network combined with the benefits many expect to derive forces the conclusion that it is only a matter of time before integrated circuit/packet switched communications networks become commonplace.

Leaping forward into that timeframe, however, the researcher finds that the added complexities of the integrated environment bring into question many of the conclusions previously drawn with respect to purer environments. One area which merits further study and analysis is routing strategy and, in particular, the effect routing strategy has on congestion in an integrated circuit/packet switched communications network. This research effort is aimed at that topic.

Specifically, this research addresses the utility of routing as a flow control strategy in a SENET-type

integrated environment. Stated more formally, this research question becomes:

Given: (1) a SENET-type integrated circuit/packet switched communications environment exists, and

(2) that this environment was initially optimized to a specific traffic flow pattern using progressive alternate (fixed) routing.

Question: Can an alternative routing strategy be identified which reduces congestion as offered workload varies?

3.2 Measures of Congestion

In order to pursue this question it becomes necessary to clearly define the meaning of "reduced congestion" and to identify a means of measuring it. For the reasons cited in Chapter II, this research proposes to view these terms from a user's viewpoint.

From a user perspective, the details of how data and voice messages are transmitted are unimportant. The user is concerned only with two things: the cost to build the communications network and the performance it offers. These two features, unfortunately, are usually in direct conflict with the typical user striving to minimize cost

while staying within an acceptable performance threshold. At some point, however, cost/benefit studies are completed, management decisions are made, and the network resources are acquired. From that point on, the user's only concern is end-to-end performance with performance typically being measured in terms of volume or response time. Common measures include:

(1) Throughput - the number of messages or packets which successfully traverse the network during a given time quantum.

(2) Average message delay - the average length of time it takes for a message or packet to traverse the network.

(3) Blocking - the percentage of time voice calls are attempted but cannot be completed because resources are unavailable.

These metrics provide the user with a quantitative assessment of how well the network satisfies communication requirements .

Alternative measures of both congestion and performance have been proposed (e.g. relative queue sizes and line utilization figures). Researchers have also proposed combinations of the three metrics noted above [49, 57, 109, 112]. However, these measures are the ones which "stare the user in the face." These are the metrics which the user must contend with on a day-to-day basis and

these are the measures which must meet the specified "grade-of-service" around which the network was designed and built.

Congestion in the integrated environment can thus be defined as a network condition where throughput and/or average message delay rise above a specified standard or a condition where blocking rises above a specified acceptable minimum. The specific values for the performance standards should be established by management and should form a key design constraint for the network.

3.3 The Experiment

In order to accomplish the research objectives, it was necessary to develop a mechanism for studying routing strategy in an integrated circuit/packet switched communications environment. Clearly, the most direct approach would have been to build an integrated network, install it in a operational environment and study the impact of alternative routing strategies. This idea is obviously unrealistic on both economic and practical grounds. Given that an experimental system could not be built, the only option left was modeling, either mathematical or simulation.

Of the two remaining options, the most satisfying approach would have been to develop a mathematical model which succinctly expresses the relationships of the

network components. Unfortunately, the large number of components and their myriad interactions and inter-relationships made such an effort impossible, at least practically, if not theoretically [39, 119]. This leaves computer simulation as the only viable alternative and the avenue pursued by this research.

The research was given a boost by Carroll Clabaugh when he developed a simulator for a circuit/packet switched network as part of his dissertation [22]. The simulator was insufficient in its original form to accomplish the objectives of this research but did form the building block for a much more elaborate simulator developed as a part of this research. The new simulator, called FLO, is an extensively enhanced version of Clabaugh's simulator and is considerably more flexible in the problem range it can address. The simulator is described in detail in Chapter IV. Though originally envisioned as an "administrative effort," the enhancement of the simulator became one of the major focuses of attention and as implemented provides a powerful tool for network investigation.

Having established a suitable environment for experimentation, the next step was to specify the routing strategies to be examined. The literature survey turned up several routing algorithms, and classification schemes, one of which was described in Chapter II. Of those

presented, flooding and its variants were quickly eliminated when analysis showed them to cause, rather than avoid, congestion. Variants of the ideal observer techniques were rejected on the premise that complete re-evaluation of packet routes at every node would overload system resources. Traffic bifurcation techniques were eliminated on the grounds that these techniques were essentially the same as fixed methods when node/link failures and variable nodal arrival rates were ignored.

Even after elimination of these categories, a host of routing strategies and their variants were left. Closer analysis showed that the policies were quite similar and that an alternative classification, one focusing on the type of data used to evaluate routing alternatives, would reduce the strategies under consideration to a more workable number. The end result was the implementation of five "generic" routing strategies, each of which were examined experimentally. The five strategies, listed in Figure 3-1, are described in Chapter V along with the experimental results obtained from each.

Strategies using delay statistics were originally considered as a sixth alternative but were later dismissed since research in the pure packet environment has shown these strategies to be equivalent to those based on line utilization [67]. Dismissal of this category was further supported by the observation that delay statistics

Generic Routing

Strategies

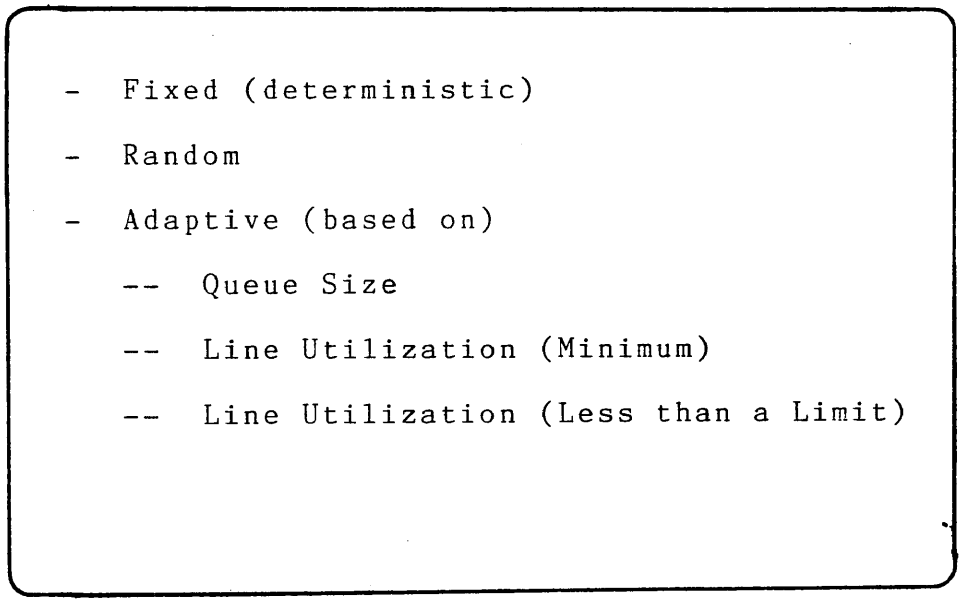


Figure 3-1

represent historical data rather than current network status.

An optimized network environment was then created via CIRPAC [64], a tool developed by Mark Kiemele as part of his dissertation. CIRPAC uses a progressive refinement technique to find the optimal network topology, i.e. least cost, satisfying a fixed set of performance constraints. This tool was used to optimize the 10-node network shown in Figure 3-2. This network configuration is based on the CYBERNET network [88]. The base network (before optimization) used 11 links to connect ten nodes, each of which was composed of a packet and a circuit switch. After several iterations, CIRPAC identified a refined topology which used 12 links to meet the specified grade of service (less than one minute average packet delay and less than ten percent blocking). The refined configuration, shown at Figure 3-3, then became the starting point for routing analyses.

Finally, the generic strategies were exercised against the established environment under varying workload conditions. The results of these experiments and their comparative analysis appears in Chapter V.

Original Topology

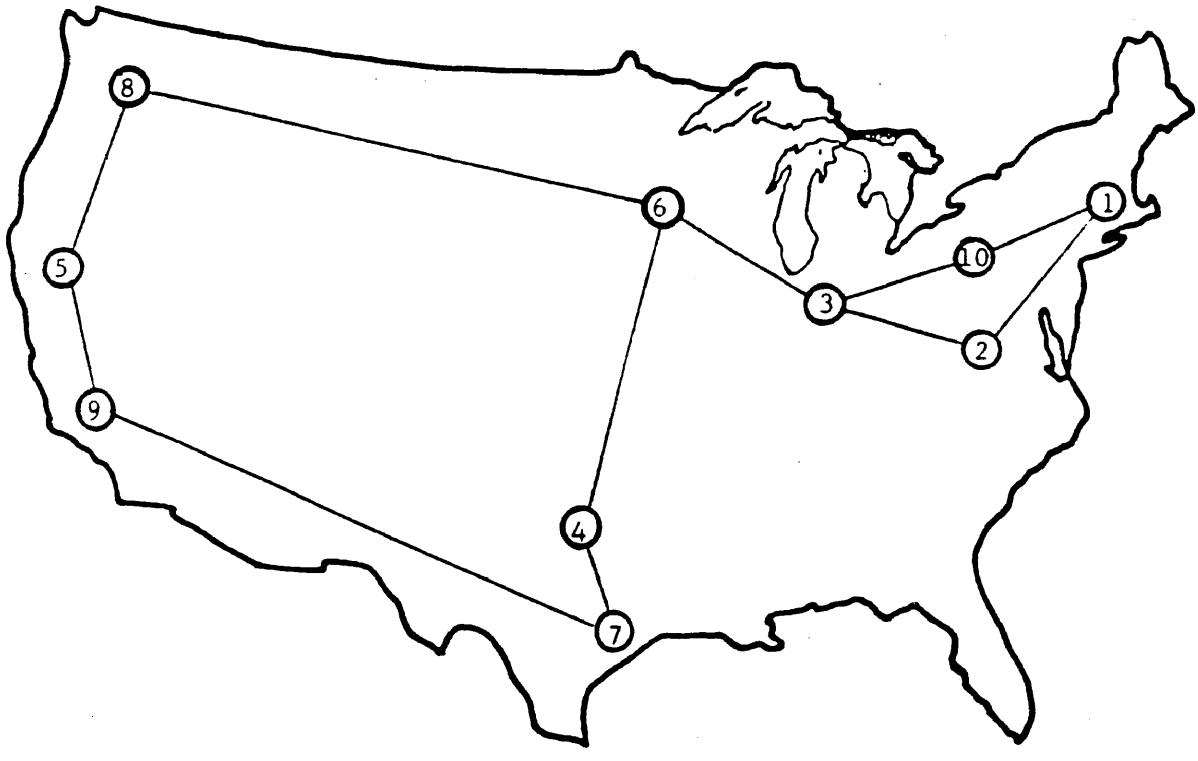


Figure 3-2

Optimized Topology

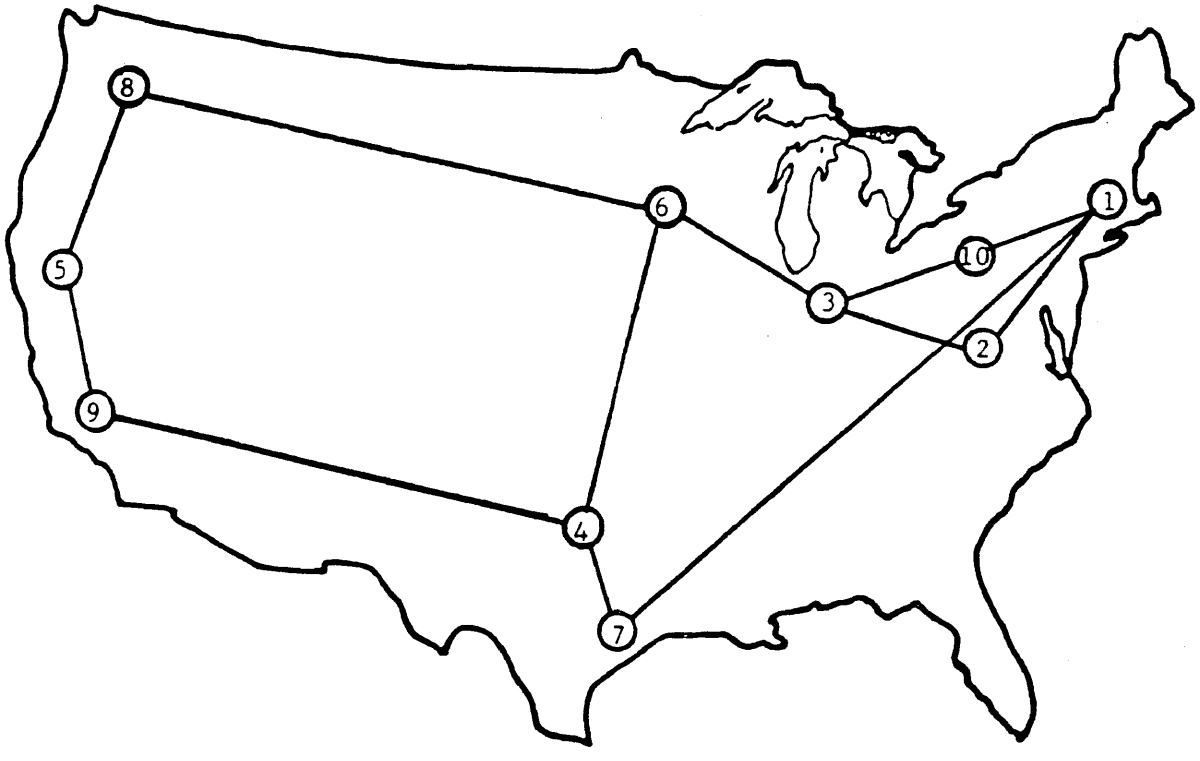


Figure 3-3

CHAPTER IV

THE NETWORK SIMULATION MODEL

4.1 Introduction

The simulator used in this research is based on one developed by Carroll Clabaugh as part of his dissertation, retaining the key simulation concepts but significantly enhancing its capabilities and flexibility [22]. This description is based on Clabaugh with changes made where appropriate.

The simulator is implemented on the Texas A&M University AMDAHL/470 computer system in VS FORTRAN. Optimization was used to obtain the maximum run-time efficiency. It is an event-oriented simulation with state changes occurring at zero-time events and the system clock advancing between these events. The permanent entities simulated include packet and circuit switching nodes and the T1 digital links connecting them. Temporary entities include voice calls and data packets. The original simulator was validated by Clabaugh [22] and Kiemele [64] via extensive sensitivity analysis. The enhanced version was validated by establishing identical outputs between the two versions for equivalent parameter settings. A listing of the enhanced simulator, FLO, is included in Appendix A.

4.2 The Research Model

The research model is based on the SENET circuit/packet switched network depicted in Figure 4-1. The principal model features include:

(1) Digital network composed of T1 carriers and digital switching nodes.

(2) Circuit switched backbone nodes (CS) with peripheral packet switched nodes (PS).

(3) Two classes of traffic:

  (a) Class I - time-critical traffic requiring uninterrupted service, e.g. voice, video, facsimile and sensor.

  (b) Class II - non-time critical traffic which can withstand reasonable delays such as interactive, query/response, and bulk data.

(4) Variable data rates.

(5) Switchable store-and-forward, furthest possible, or end-to-end operation.

(6) Checkpoint and restart capability.

(7) Alternative routing strategies.

(8) Flexible configuration definiton.

(9) Common channel interoffice signalling.

(10) Flexible routing definitions.

Circuit/Packet Switched Network

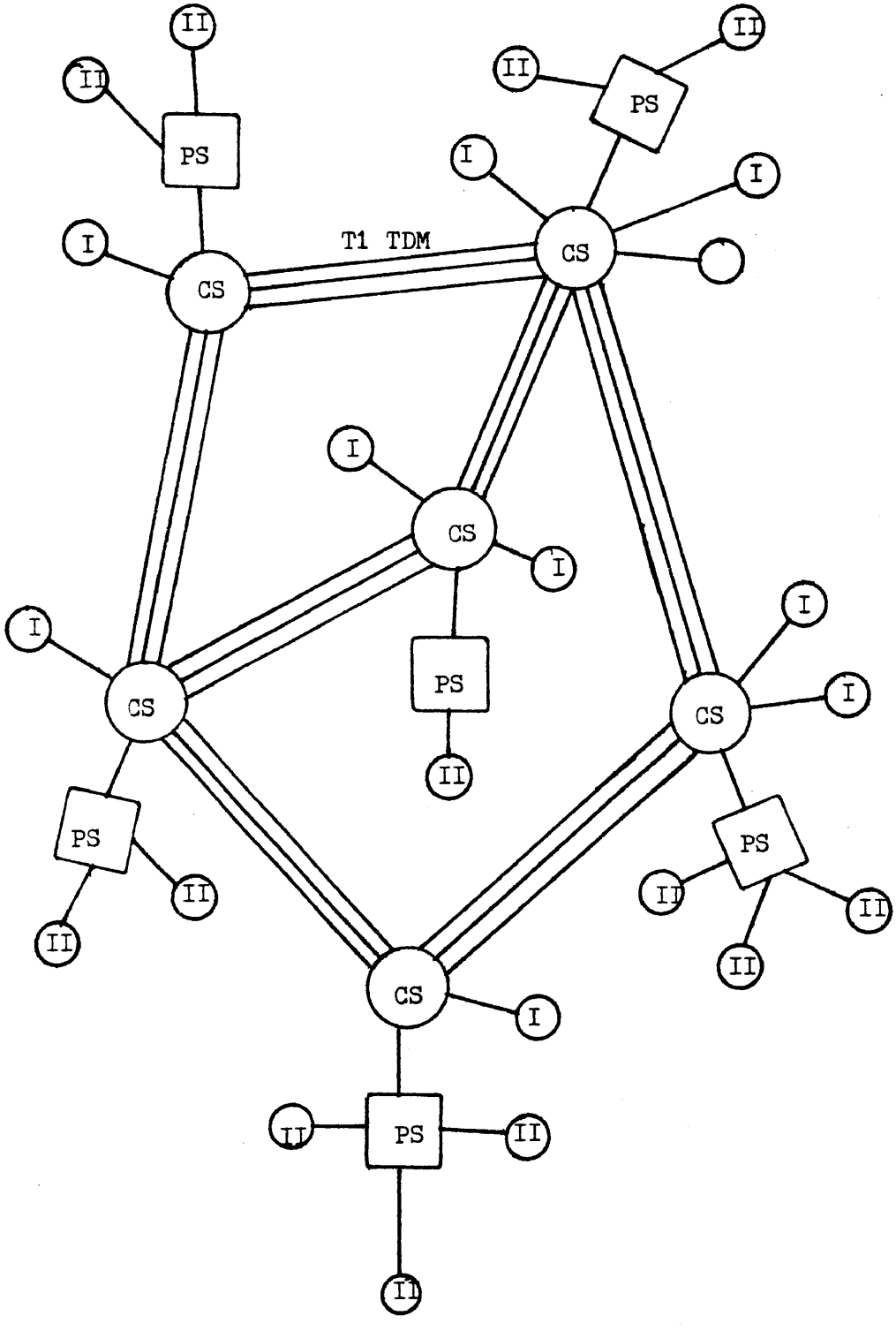


Figure 4-1

Network nodes are composed of Circuit Switches (CS) with peripheral Packet Switches (PS). Class I users are directly connected to the CS nodes while Class II users terminate at the PS nodes. The PS nodes queue all subscriber traffic until it can be serviced by the associated CS node. If the offered traffic can be handled immediately, the queuing mechanism is by-passed. For the purposes of the research model, the queues are considered to be infinite. In reality, they are sized appropriately by setting bounds on the FORTRAN arrays which hold them.

The nodes are inter-connected by digital T1 carriers with a SENET-type switching superstructure. The carriers are synchronously clocked into "envelopes," each of which is composed of header fields, trailer fields, and a series of time division multiplexed slots. The two traffic classes compete for these slots on a "first come, first served" basis.

Class I traffic (voice, sensor, etc.) requires uninterrupted full duplex service. This traffic class is accommodated by reserving a slot in each envelope along the route selected. When there are insufficient resources to accommodate a Class I call, it is "blocked" and the user is notified that service is unavailable. Queueing is not allowed for this traffic class.

The mechanisms available to handle class II traffic (interactive, query, etc.) are more varied since traffic

of this sort can withstand reasonable interruptions (delays) in service. Further, class II traffic is uni-directional, therefore, it only requires a half duplex line. Acknowledgements are either "piggybacked" on return traffic or carried by special purpose "ack packets." Transactions (or messages) are broken into fixed sized packets each of which is routed over the network. One of three routing philosophies can be selected by the user:

(1) end-to-end,

(2) furthest possible, or

(3) store-and-forward.

In the first mode, routes are selected only if they form a complete path from the source to the destination node. If a complete path is unavailable, traffic is queued at the source node.

Under the second philosophy, packets are transmitted as far along their selected path as possible. When packets reach an unavailable link, they are queued at the intermediate node. The packets are then granted priority service on future route selections.

The final mode of operation calls for packets to be transmitted only to the first node of their selected route where they are then queued for further service. The user can select one of two "visibility" levels to support the routing decision. The first causes the route selection to be based solely on locally available data, while the

second level bases route evaluation on global data.

In all cases, routes are evaluated and selected based on the strategy chosen by the user. The five "generic" strategies available are listed in Chapter III.

There are no explicit flow controls implemented in the model. Node-to-node controls become unnecessary because of the circuit switched backbone. Host-to-node controls are unnecessary because of the unlimited storage capacity of the PS nodes as are host-to-host controls. Finally, end-to-end flow control is realized via the routing strategy.

4.3 The Queuing Model

The myriad complexities and nodal interactions of integrating voice and data traffic within the same computer-communications network make an analytic solution impractical [39]. By decomposing the network, however, each node can be viewed as the queuing process shown in Figure 4-2 [46].

Each node can be visualized as a multi-server system with K sets of C independent parallel servers. The values of K and C are fixed for each node at network definition time. The first value, K, indicates the number of outgoing links attached to the node. The second value, C, varies for each of the outgoing links and reflects the capacity assigned to each of those outgoing links, i.e.

Queuing Model

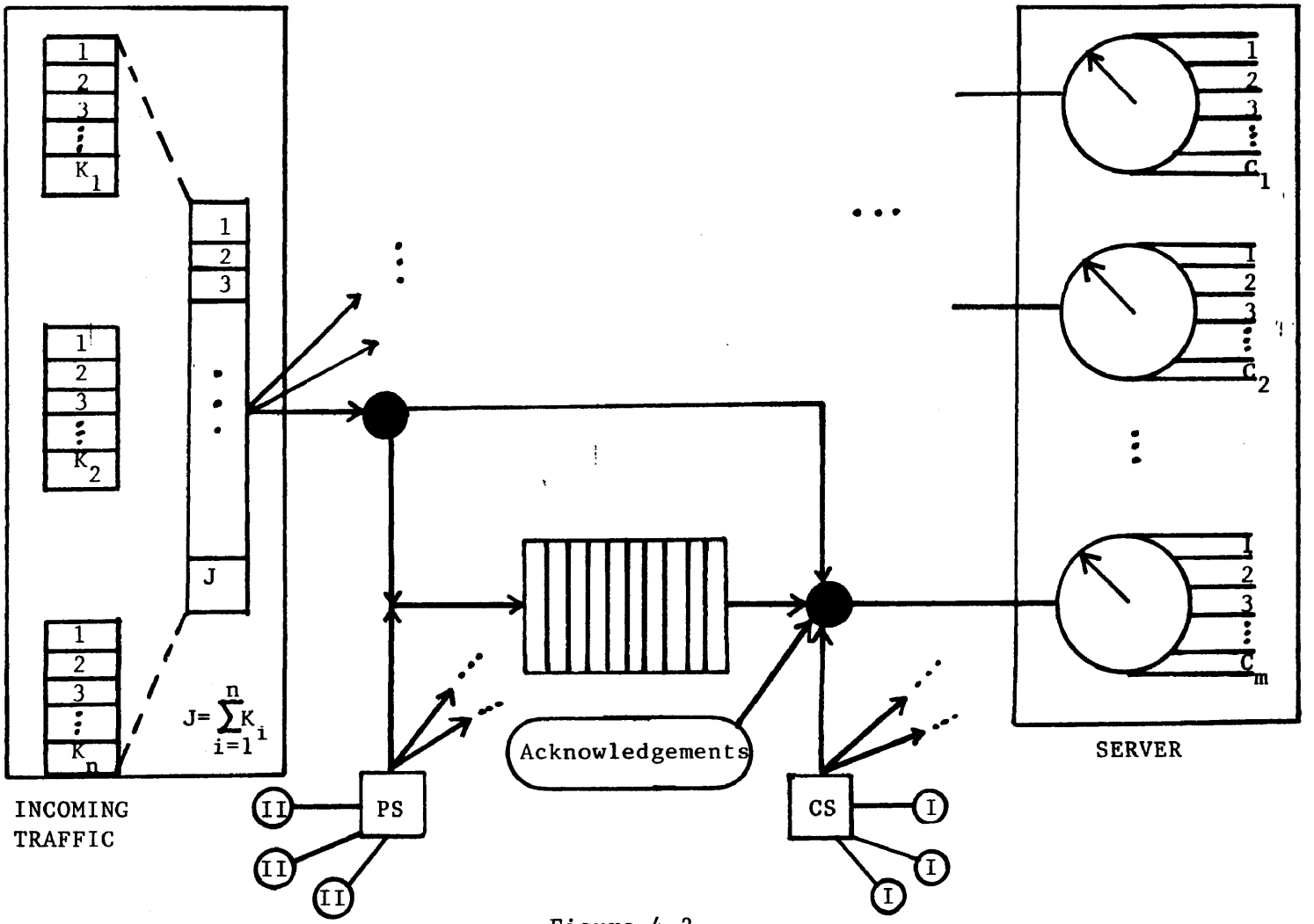


Figure 4-2.

the number of slots in each SENET envelope.

Each server is assumed to have a constant service time which reflects the time required to properly build and forward a SENET envelope. In actuality, the envelopes are not handled as a group. Rather, each slot is handled as it arrives (sequentially). Making this assumption helps to simplify the queuing model. The model is further simplified by assuming that all envelopes are synchronized, i.e. all envelopes exit the node simultaneously.

The server must satisfy the requirements of the following four service classes.

(1) Incoming links

(2) Circuit Switches

(3) Packet Switches

(4) Acknowledgements

4.3.1 Incoming links

The number of incoming links is a function of the network configuration and can vary from one to the size of the network. In the experimental case chosen each node has two. Each incoming link possesses a fixed number of slots, based on its defined capacity. For model simplicity, it is assumed that the incoming links are also synchronized. Letting J represent the total number of incoming slots, this subelement would be modeled as a bulk

input system with the random variable ranging between 1 and J. Unfortunately, the arrival process is not Markovian. Though it carries poisson generated packets, the random variable is heavily dependent upon the routing philosophy/strategy chosen and the network status. This significantly complicates the queuing model.

Incoming packets are handled by the node in one of three ways depending on the routing philosophy selected. If the packets are to be immediately forwarded, the nodal process places it on the correct link, i.e. acquires the appropriate server. If the packet is to be queued, it is entered into an infinite queue along with other packets that share its destination. The final option occurs when the current node is the destination for the traffic. In this case, packets are removed from the system. For modeling purposes, it is assumed that this decision occurs in a negligible amount of time.

This handling procedure further complicates the model. First, packets will be removed from the network only if the current node is their destination. The probability that a given packet is destined for the current node must reflect the probability that other nodes will generate packets destined for the current node, the routing philosophy/strategy in effect, and the status of the network.

Second, the queuing mechanism for incoming packets is not first-in, first-out (FIFO). Because packets are queued based on their original creation time, they could end up virtually anywhere in the queue depending on the decisions which were made at previous nodes. Finally, the immediate forwarding of certain packets causes this subelement to act like a priority-based system with incoming packets taking precedence over queued traffic.

4.3.2 Circuit Switches

The ciruit switch (CS) is the concentrator for all class I traffic, i.e. that requiring uninterrupted service. Because queuing delays are unacceptable, the hub has no storage capacity. Class I service requests are assumed to have a poisson distribution and exponential interarrival times and lengths. Since the CS hub can only satisfy users when a slot is available, it can be viewed as having a capacity of L slots. The added constraint of limited server capacity, however, causes the value of L to vary from 0 to the total number of slots on the appropriate outgoing link. The value of L is determined by the formula:

$$L = C - \overline{P}$$

where C = the number of slots available on the proper link.

P = the number of incoming packets to be immediately forwarded

The destination of each service request is a random variable with each network node having equal selection probability. The outgoing link required is determined by the routing strategy based on the destination and network status.

4.3.3 Packet Switches

The third server category, the Packet Switch (PS), is assumed to have an infinite storage capacity and is fed from two sources. The first source is the collection of all class II subscribers terminated at the current node. Transactions are assumed to arrive according to a poisson distribution and to have exponential inter-arrival times. Transaction lengths (number of packets) are assumed to be geometrically distributed [35]. The second source of class II traffic is the set of the incoming links. Depending on the routing philosophy selected, incoming class II packets are removed and queued to their appropriate destination along with nodal subscriber traffic. The queues are assumed to have infinite lengths. However, as mentioned above, the sorting mechanism is based on the packet's entry into the system. Thus, incoming packets could easily advance to the front of a queue if it entered the network prior to entries already queued. Note also that each destination and, thus, each subset of servers has its own queue. Entries from the

Class II queue are accepted for service whenever there is available capacity in the appropriate outgoing SENET envelope.

4.3.4 Acknowledgements

The final service category is called acknowledgement packets or "ackpacs." Since class II traffic is unidirectional, there is no need to reserve a full duplex line to support it. Rather, a half duplex line is adequate. The only exception to this rule is the "overhead" required to acknowledge successful transmission. Normally ackpacs can be "piggybacked" on packets traveling to the original source. If there is no traffic available to "piggyback" on, however, a single packet message called an ACKPAC will be generated and sent from the destination to the source.

4.3.5 Model Complexity

Clearly, even a simplified version of the nodal process is difficult to model analytically. In lieu of a closed-form solution, a simulation model has been used to examine the network. The next section describes the simulator in detail.

4.4 The Network Simulator

The modified simulation model, hereafter referred to as FLO, is hierarchically structured into four major functional modules. This section describes each of these modules and its major subelements.

4.4.1 FLO

As the driver for the simulator, FLO is responsible for the three global management functions of model definition/initialization, simulation execution and experiment termination. FLO accomplishes these duties via subroutine calls to one or more support procedures, the calling hierarchy of which is depicted in Figure 4-3.

At the highest level of abstraction, FLO is designed to accept as input a network configuration, a routing strategy, and the parameters needed to control the experimental run. FLO will then configure the network as requested and, applying the selected strategy, determine the largest workload which the network can accommodate without exceeding a specified grade of service. FLO achieves this objective by using the input control parameters to systematically increment offered workload until a suitable termination condition is reached.

FLO views initialization as a three-phase process, each of which is accomplished by a different subroutine.

Calling Hierarchy of
FLO

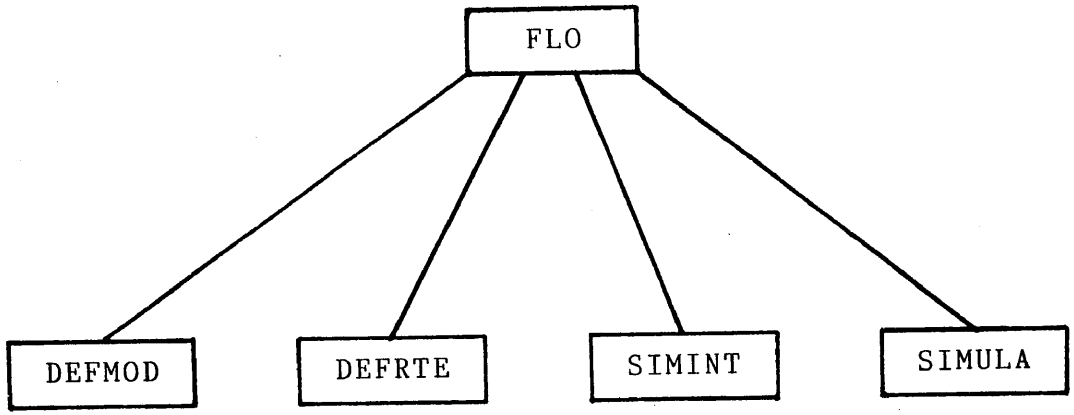


Figure 4-3.

DEFMOD implements the first phase by defining the network configuration to be used in the experimental run. Phase II is realized by a call to DEFRTE which accepts as input, the routing strategy definition data and control parameters. The last phase of the initialization is accomplished by a call to SIMINT. This subroutine is charged with using the previously input definition and control data to initialize the tables and variables needed to support simulation execution. On the intial pass, initialization is restricted to copying permanent data into temporary tables. On subsequent passes, however, SIMINT is charged with using the control parameters to increment the offered workload.

Having initialized the necessary variables and tables, FLO requests simulation of the defined network environment via a call to SIMULA. This subroutine will execute until its parametrically defined termination condition is reached. During execution, network performance metrics are captured for analysis. One aspect of that analysis is the determination of whether the network has become overloaded or if the network has become unstable, i.e. summary performance measures do not indicate steady state is reached. Either of these conditions would cause FLO to halt the experimental run. The experimental run will also be halted if it exceeds the iteration limit. The following sections describe each of

these modules.

4.4.2 DEFMOD

DEFMOD realizes the first portion of the three-phase initialization process required to support simulation execution. As mentioned earlier, this subroutine collects the information which defines the network configuration to be used for the experimental run. Flexibility is provided by allowing this data to take two forms, either a standard configuration definition or a snapshot from an earlier experimental run. The variable RESTRT is used to distinguish between these two cases and if set forces a subroutine call to SSREAD thereby bypassing the standard initialization process.

If RESTRT is not set, DEFMOD proceeds to fill the various arrays and tables needed to support the simulation process. A detailed description of each array/table can be found at Appendix B. The first array to be filled is ORPRMS. This is a seventeen element array containing the initial parameter settings required by SIMULA. These values are used to initiate the simulation process and then form the basis for the systematic increase in offered workload.

The second array filled is ORSDTB. This array contains the seeds used to support independent pseudo random number generation for each node.

The third array filled is KONECT, a square matrix which details the connectivitity of the experimental nodes. The link between any two nodes X and Y is described by the matrix with a -1 indicating there is no link from X to Y, a 0 indicating that it is illegal to connect nodes X to Y, and a positive value indicating the number of the link which connects them.

The last four arrays read by DEFMOD are all linear arrays which describe the characteristics of the channels which make up the network configurations. The first, SORCHL, identifies the node at which each channel originates while NODCHL provides the node at which each channel terminates. PARM3 specifies the number of slots available over each link while JARM3 contains the total number of slots available in the first n circuits.

Taken together, these tables give the user the flexibility to define virtually any network configuration conceivable.

4.4.3 DEFRTE

The second phase of the initialization process is accomplished by DEFRTE, a subroutine responsible for gathering both the parameters which control the experimental run and the data needed to define and support the routing strategy used. The control parameters gathered are described below.

(1) STOFWD - A switch indicating whether class II traffic is to use end-to-end or store-and-forward techniques in traversing the network.

(2) WCINCR - The factor by which class I workload is to be incrementally advanced through experimental runs.

(3) WPINCR - The factor by which class II workload is to be incrementally advanced through experimental runs.

(4) SWITCH - A variable used to select the routing routine to be used.

(5) GLOBAL - A switch defining the "visibility" to be used in routing strategy analysis. If the switch is "off" then only information from directly connected nodes is used in the analysis. If "on" then data from the entire network is made available.

(6) STPPRI - A switch used in conjunction with STOFWD to achieve the "furthest possible" routing philosophy.

Specific data items must also be collected to support the routing alternative chosen. These include:

(1) DESTAB - The primary routing table, DESTAB is used for fixed routing strategies. It

contains in matrix form the primary link choice for traffic which is to be routed from any node, X, to any other node, Y.

(2) DSTALT - The secondary routing table; it is used by fixed routing strategies whenever the primary route is unavailable.

(3) NMTRYS - A variable supporting the random routing technique which determines how many randomly selected routes will be tried before the strategy indicates that no route is available.

4.4.4 SIMINT

The final phase of the initialization process is accomplished by SIMINT, a procedure which initializes the global variables and arrays used by SIMULA. This step is necessary because these variables/arrays will change over the course of a simulation run. SIMINT resets these to their appropriate values for incrementing the offered workload and then reinitializes the simulation for the next experimental cycle.

The variable/arrays initialized include:

(1) PARAM - A copy of the parameters stored in ORPRMS.

(2) SEEDTB - A copy of the seeds used to randomly

generate traffic.

(3) NLINES - An array showing the number of slots available on each channel.

(4) WCFACT - The factor used to increment the class I traffic.

(5) WPFACT - The factor used to increment class II traffic.

4.4.5 SIMULA

SIMULA realizes the simulation of the experiment defined by the previous three modules. It is an enhanced version of the simulator implemented by Clabaugh [22] and it simulates the SENET-type environment described earlier.

The main routine (SIMULA) initiates the simulation by performing necessary initialization and echoing the network configuration and parameter settings. The initialization is accomplished by a subroutine call to BDATA and a sequence where the initial class I and II arrivals are generated for each node. SIMULA then enters a tight loop where repeated calls are made to EVENT which selects and handles the next simulation event, causing the simulation to advance. The loop is interrupted periodically to record performance statistics for later analysis and summary. The loop stops at an experimentally determined point.

SIMULA then proceeds to analyze the performance statistics captured to determine whether the network over stabilized or has become overloaded. This information is returned to FLO for appropriate action. At specified intervals SIMULA will capture snapshots of the network operation to support later restarts if needed. The calling hierarchy for SIMULA is shown in Figure 4-4.

The simulator is table-driven in that all pertinent data is held in system tables (arrays). Changes in state are realized by updating these tables. The tables are described in detail in Appendix B. The major tables include the following.

(1) CHANTB - An array reflecting current status of all network channels including ones which are occupied/free, the class I/class II traffic carried, and its utilization.

(2) EVTBL - An array showing the next event scheduled to occur for each switch.

(3) QUEUE1 - An array containing data associated with all class II transactions queued for service or scheduled to occur in the future.

(4) QUEUE2 - An extension of QUEUE1.

Calling Hierarchy
of
SIMULA

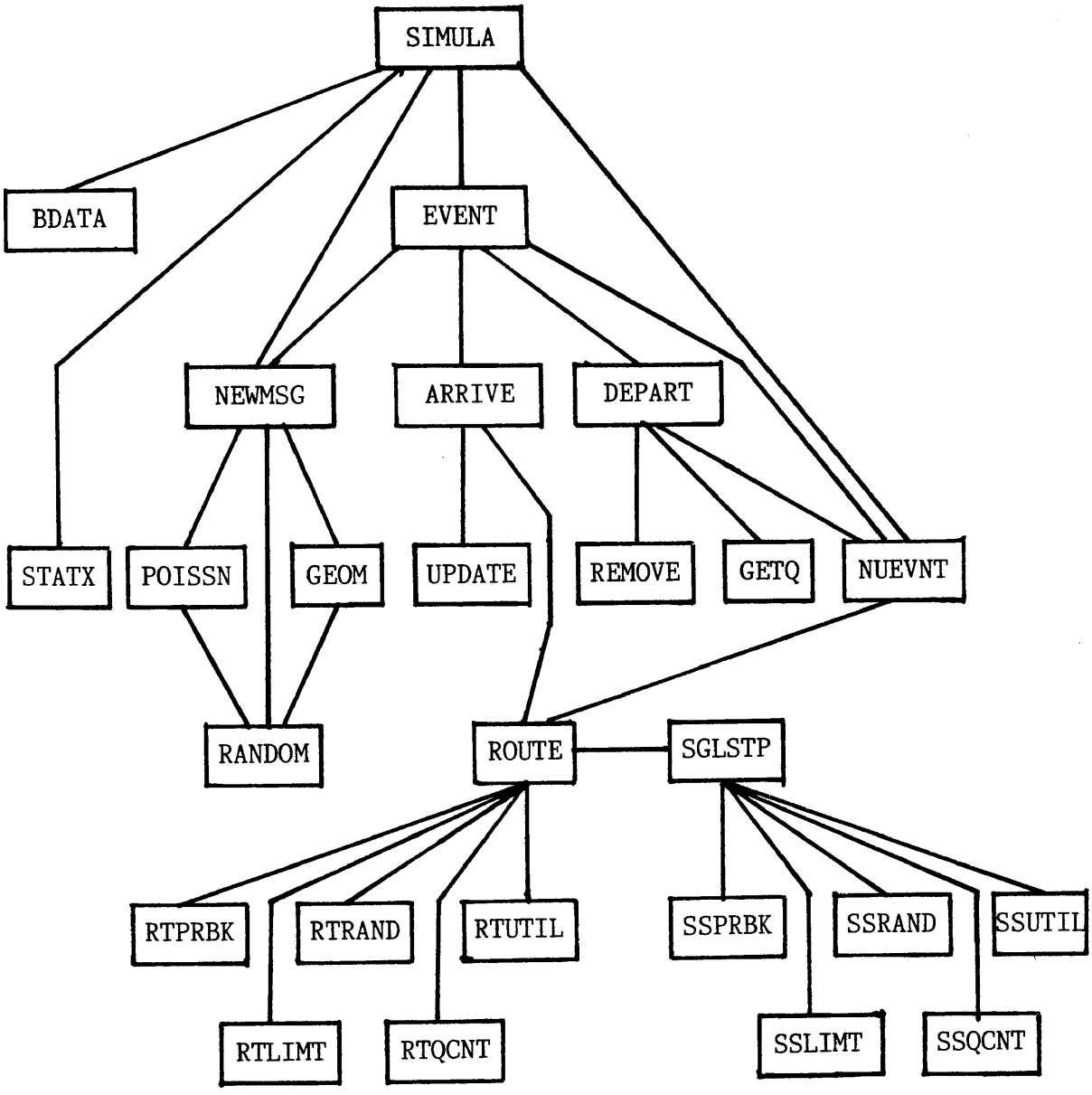


Figure 4-4.

(5) CALQ1 - An array containing data associated with each class I transaction queued for service or scheduled to occur in the future.

(6) CALQ2 - An extension of CALQ1.

4.4.5.1 EVENT

This procedure is responsible for determining which system event is next, advancing the system clock to that event, handling it properly, and then generating the next event if appropriate. EVENT first examines system counters to determine if one of the periodic data recording points has been reached. If so, a call is made to STATK which examines current performance metrics and saves selected information. EVENT also determines whether the experimentally determined steady state point has been reached. If so, various summary values are re-initialized for data collection.

EVENT's second major task is to select the next system event. This is accomplished by perusing EVTBL. Once found,EVENT updates summary statistics and then handles the event via a call to ARRIVE (to handle transaction arrivals) or DEPART (to handle transaction departures). These routines are responsible for updating the system tables to reflect either an arrival or a departure. Manipulation of these tables is based on the

path selected by ROUTE, if the end-to-end routing philosophy is being enforced. Updates are based on the path selected by SGLSTP if the furthest-possible or store-and-forward philosophies are in effect.

Finally, EVENT completes the simulation cycle by generating a replacement for the system event which was just handled.

4.4.5.2 ROUTE and SGLSTP

One of the key features of the enhanced simulator is the isolation of the routing strategy into a common set of routines. ROUTE and SGLSTP form the central access points for those routines. ROUTE accepts as input a set of control parameters defining the path needed and the evaluation strategy to be used in selecting it. ROUTE returns the selected path via the global arrays TCHNLS, FTRACE, and RTRACE.

The routing routines implemented include the following:

(1) RTUTIL

(2) RTPRBK

(3) RTRAND

(4) RTQCNT

(5) RTLIMT

Each of these are associated with a specific generic strategy. Detailed descriptions of each are provided in

the next chapter.

SGLSTP is functionally equivalent to ROUTE. The difference between the two is that SGLSTP is designed for use when store-and-forward routing is in effect. SGLSTP accesses the following subroutines, each of which apply a "generic" technique to the store-and-forward or furthest-possible mode of operation.

(1) SSUTIL

(2) SSPRBK

(3) SSRAND

(4) SSQCNT

(5) SSLIMT

4.5 SUMMARY

The enhanced simulator is an extremely complex facility, comprising just over 5,000 lines of FORTRAN code (as opposed to 1800 for the original version). With this complexity comes a flexible capability to experiment with routing and flow control issues in an integrated environment. The next chapter details the results of the experiments conducted.

CHAPTER V

EXPERIMENTAL RESULTS

5.1 Introduction

The network simulation model described in Chapter IV was used to investigate the generic routing strategies identified in Figure 3-1. These basic strategies and three of their variants were examined under progressively increasing workloads. This chapter contains detailed descriptions of the routing strategies explored, the conditions under which they were exercised, and the experimental results obtained.

5.2 Routing Strategies Explored

The ability to investigate alternative routing strategies is enhanced by the structure of FLO. During the initialization phase, all potential paths are generated and stored in a path table, P, along with characteristic information such as source node, destination node and path length. Whenever a route is required, alternative paths connecting the desired source and destination nodes are evaluated and the "best" route is chosen. The path table index which references the selected route is used as a label and is associated with the transaction requesting service via either CALQ2 for class I traffic or QUEUE2 for class II traffic. The

label, and hence the selected path, is retained until the routing philosophy determines that the choice should be re-examined. To support transaction processing, the path selected is also returned via the global arrays TCHNLS, FTRACE, RTRACE. These arrays support simulation management functions and are reset for each route request.

Specifying the route a transaction will follow via a reference into the path table is essential to the flexible and efficient operation of the network simulation model. Since the definition of "best" can easily vary with network conditions, the choice of an optimal route becomes a "point in time" decision, with no guarantee that later evaluations will yield the same result. The reference approach allows the routing decision to be retained without vastly increasing the simulation storage requirements.

The evaluation of alternative routes and the decision as to which to select is the responsibility of either ROUTE or SGLSTP depending on the routing philosophy under consideration. These procedures select the optimal path based upon the desired source and destination nodes, the condition of the network, and the routing stategy opted for. Detailed descriptions of the routing strategies examined follow.

5.2.1 Fixed (RTPRBK and SSPRBK)

The fixed routing strategy was implemented as the standard by which to judge the relative merits of the other strategies. The specific routing tables used were those for which the network was optimized via CIRPAC.

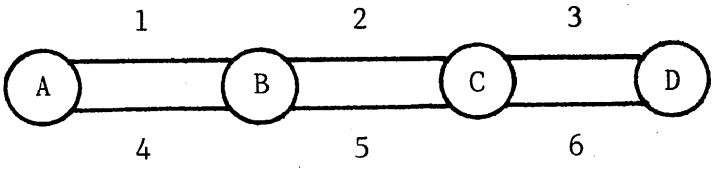
RTPRBK selects a route by perusing the DESTAB and ALTCHN tables. Defined at initialization and fixed for the duration of the experiment, these square arrays identify both a primary (DESTAB) and secondary (ALTCHN) route from any source node to any destination node. The (x,y) coordinates of these tables reference link identifiers. Traversing the indicated link will cause a transaction to reach either node Y or some intermediate node, I. If the identified link does not terminate at the desired destination, the link at coordinates (i,y) is investigated. Repeated examinations will eventually yield a link connected to the ultimate destination. At each stage, the available capacity of the selected link is examined. If there is insufficient capacity to accommodate the requested traffic, the link identified by the ALTCHN table is examined for potential use. If neither of the tables can provide a suitable link, a bottleneck condition is noted and returned in lieu of the requested route. Alternatively, if a route with sufficient capacity is identified a reference to that route is returned.

The fixed strategy outlined above, called "Progressive Alternate Routing" by Clabaugh [22], is significantly more robust than fixed strategies which choose a path from one or more pre-defined alternatives. Since each link of a path can take on at least two values, as demonstrated by Figure 3-3, the number of possible paths which are considered during route selection is at least 2^n where n is the number of links in a route. Consider the example depicted in Figure 5-1. Assume that a message is to travel from node A to node D with the primary and secondary routing tables defined as shown. Depending on the availability of the links at the time service is requested any of the eight routes listed in the figure might be selected.

SSPRBK follows a procedure analogous to that used by RTPRBK for selecting the "best" route. The principal difference between the two procedures is that the evaluation process is limited to the first step of the route since this is the only element which will be used.

A variant of the progressive alternate routing technique, called "Primary-Only Routing" was also considered. Identified as RTPRON and SSPRON in the discussion of experimental results, this variant was implemented by resticting the progressive alternate strategy to the primary routing table (DESTAB).

Progressive Alternative Routing



	A	B	C	D
A	0	1	1	1
B	1	0	3	3
C	2	2	0	3
D	3	3	3	0

	A	B	C	D
A	0	4	4	4
B	4	0	6	6
C	5	5	0	6
D	6	6	6	0

1-2-3
1-2-6
1-5-3
1-5-6
4-2-3
4-2-6
4-5-3
4-5-6

Figure 5-1.

5.2.2 Random (RTRAND and SSRAND)

Conceptually, random route selection is made in the complete absense of information, including the desired destination. This strategy is possible in FLO's "single-step" mode of operation, where routes are re-evaluated at each intermediate node. Under this routing philosophy, transactions repeatedly request routes until the desired destination is reached. However, the "end-to-end" routing philosophy has, as a basic tenet, the notion that traffic will reach its destination and be removed from the system after traversing the selected route. Thus, application of random routing concepts to the alternative philosophies afforded by FLO yields two variants of this basic strategy.

RTRAND represents the directed variant of random routing. When a request to traverse the network is received RTRAND randomly selects one of the routes which connect the desired source and destination nodes. Selection is based on a uniformly distributed random variable therefore all paths connecting the requested endpoints have an equal chance of being selected. Once selected, the route is evaluated to see if it possesses sufficient available capacity to handle the offered traffic. If there is, the index of the selected path is returned. If not, the variable NMTRYS is examined to see if an alternate route should be examined. NMTRYS is a

user-specified parameter fixed at system initialization time. It indicates the number of routes to be examined before RTRAND returns a bottleneck condition. For the experiments supporting this analysis, NMTRYS was assigned the value one (1).

SSRAND represents the second variant of random routing and realizes a true "drunkards walk". When asked to identify the "best route", or in this case, the "best next step", this procedure randomly selects one of the links emanating from the source node as the next link to traverse. SSRAND then ensures that there is sufficient available capacity on the selected link and, if there is, returns that "route". If there is insufficient capacity available, SSRAND examines NMTRYS to determine whether alternative routes should be examined before a bottleneck condition is noted.

5.2.3 Adaptive Procedures

Adaptive procedures share a common approach to route selection. These strategies periodically examine network status information, using it to guide the route selection process in an effort to adjust to network conditions.

An important feature of any routing strategy is its ability to adapt to ever changing network conditions. This capability, sometimes called "adaptiveness", is a function of both the routing strategy under consideration

and the routing philosophy in effect. While the former identifies the data and logic to be used in evaluating alternative paths, the latter determines the frequency with which route selections are re-examined. As the frequency of re-evaluations increases, so does the capability of the routing strategy to react to changes in network status.

5.2.3.1 Link Utilization (RTUTIL and SSUTIL)

Link utilization procedures evaluate alternative paths based on the percentage of link capacity currently in use. RTUTIL responds to a service request by identifying all paths which connect the desired source and destination nodes and then evaluating them according to the formula shown below:

$$\text{MIN} \left[\sum_{i=1}^{k} u_i^{\,p} \right]$$

where: k = length of the path

u_i = percentage of available capacity in use

p = user specified exponent

This procedure calls for the percentage utilization of each link along a candidate path to be computed and then raised to a user specified exponent, provided via the variable POWER. The resultant values are then summed to

form a "utization metric" for that path. This metric forms the basis for path selection with the path with the lowest metric being selected.

This algorithm is varied slightly for those situations where one of the component links of a path does not have sufficient available capacity to accommodate the requested workload. This situation is resolved by assigning that link an arbitrarily high value, one which is large enough so that the any route using that link is unlikely to be selected.

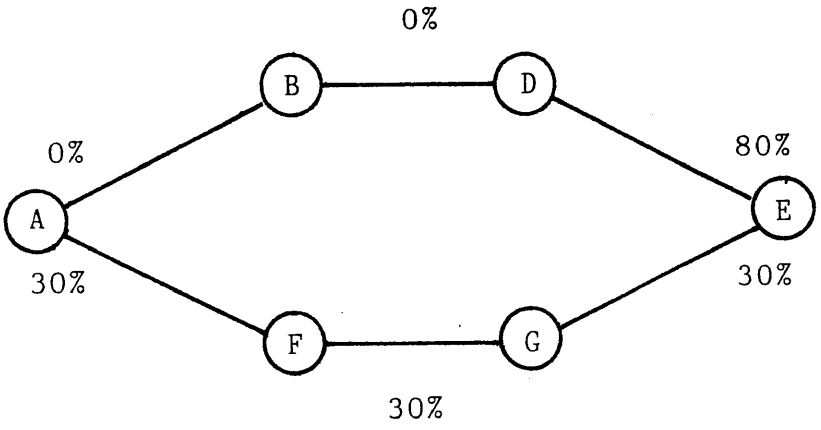
The utilization metric reflects several key features of a given path. Certainly, it reflects the relative line utilization of competing routes. Given two paths, both of length one, the route with the lowest utilization will be chosen. If two routes of unequal length are compared, the situation becomes more complex, with the shortest typically being selected. This rule of thumb fails, however, whenever the combined utilization of the longer route is less than that of the shorter one. For example if a route of length one has a link utilization of 75% and is compared to a route of length two where each link is 40% used, the first route would be chosen. However, the second route would be selected if the utilization of each of its links were 37% or less.

The above weighting technique becomes even more complex when the effects of unequal link utilization along

a path are considered. One could easily postulate examples where it would be imprudent to blindly select the path with the lowest utilization metric. This realization prompted the use of exponentiation to aid in path discrimination. Experimentally, it was determined that exponents greater than one (1) yield essentially the same relative values, therefore only one (1) and two (2) were used as exponents in the experimental runs supporting this analysis. These variants are identified as UTIL-1 and UTIL-2 in the discussion of experimental results. Figure 5-2 gives an example of how exponentiation can effect route selection.

The logic used by SSUTIL is analogous to that just described for RTUTIL. SSUTIL differs, however, from its counterpart in two ways. First, as do all route selection procedures which support "single-step" operation, SSUTIL returns only the first step of the selected route. The second difference is more significant and deals with the amount of data available to SSUTIL during route selection. Based on the user-provided setting for GLOBAL, SSUTIL will adjust its "visibility". If GLOBAL is set to one (1), SSUTIL will select a route based on link utilization data for all remaining links between the current node and the desired destination. If GLOBAL is set to zero (0), SSUTIL will base its decision on only the status of links directly connected to the current node.

Effect of Power
on
Link Utiization Scores



.00	.00
.00	.00
+.80	+.64
.80	.64

.30	.09
.30	.09
+.30	+.09
.90	.27

➡ If POWER = 1, select upper route

If POWER = 2, select lower route

Figure 5-2

5.2.3.2 Queue Count (RTQCNT and SSQCNT)

RTQCNT and SSQCNT base their evaluation of alternative routes on the size of the queues which have built up at intermediate nodes along a given path. Using an evaluation mechanism similar to that described for the link utilization procedures, RTQCNT evaluates each link based on the queue size at the link's source as a percentage of that link's capacity.

When a route request is received, RTQCNT first identifies all potential paths connecting the desired source and destination nodes. A queuing metric for each candidate path is then computed by summing the individual values for each of the path's component nodes. The individual values are computed by dividing the queue size by the capacity of the link to be used. This weighted value is then raised to the power provided in the variable, POWER. The path with the minimum queuing metric is then selected.

Just as in the link utilization procedures, the evaluation mechanism used to support routing decisions based upon queue size must resolve several important factors. Analyzing an increasingly complex sequence of examples, the resolution techniques become more apparent.

Consider first the case where two paths of equal length (assume one) are compared. If the links have the same capacity, the path with the smaller queue will be

favored. If, however, the queues are of the same size, it will be the link with the largest capacity which is deemed optimal. As the simplifying assumptions of equal capacity and queue size are withdrawn the possible comparison scenarios mushroom.

Further complexity is added by eliminating the assumption of equal path length. As in the link utilization procedures, the queuing metric for each path is a summation of the individual link values. In general, this comparison technique favors the shorter paths, however examples can be constructed which cause the longer path to be selected. Exponentiation was again used to help resolve situations analagous to that demonstrated by Figure 5-2. One (1) and two (2) were the exponents used in this series of experiments. The results are distinguished in the discussion of experimental results by the labels "QCNT-1" and "QCNT-2".

The workings of SSQCNT closely parallels that of SSUTIL. It also differs from its counterpart, described above, by first, providing only the first "step" of the route selected, and second, allowing the restriction of its visibility based upon the value found in GLOBAL.

5.2.3.3 Link Utilization Limit (RTLIMT and SSLIMT)

The basis for analysis in the limiting procedures is also link utilization. However a significant difference

exists in the way in which the raw link utilization data is combined. Rather than pursue the path with the lowest weighted utilization, these procedures select the shortest path such that all component links possess a progressively increasing quantum of excess capacity.

Upon receipt of a request for routing service, RTLIMT first identifies candidate paths and then examines these paths from shortest to longest, investigating the utilization of each link. The first path to be identified with utilization factors of 85% or less for all its component links is selected. If all candidate routes are examined and none meets the "less than 85%" criteria, the procedure will start over with 90% being used as the comparison figure. Assuming no routes are identified, the comparison figure would progressively increase to first 95% and then 100%. This last figure would simply cause the shortest path to be selected.

The motivation for this technique stems from a desire to retain open capacity on lines as long as possible. The tradeoff, in this case, concerns by-passing shorter paths with high utilization factors (though possibly sufficient capacity to handle the requested traffic) to use a longer path with lower utilization metrics.

Again, SSLIMT is analogous to RTLIMT, differing from the latter in the ability to forward only the first step of the selected route, and the capacity to limit its

visibility to only information available about directly connected nodes.

5.3 Resource Constraints

The flexibility of FLO provides the analyst with a virtually unlimited capability to investigate the integrated network environment. Not only can any network configuration be realized but given a particular configuration, the possible combinations of traffic arrival patterns/rates, routing philosophies and routing stategies are endless. These features, taken together, imply that experimentation is bounded only by the analyst's imagination and curiosity.

Unfortunately, the reality of limited computer resources and time quickly forces compromises in an effort to obtain the most cost-effective information. In this project, four major compromises were made to reduce the computing budget.

The first two were introduced in Chapters III and IV. They involve the use of "generic" routing strategies to reduce the number of alternative stategies to be tested and restriction of the experiment to the 10-node configuration shown in Figure 3-2.

The third compromise was a restiction of the workload settings to be examined. This was accomplished in three ways. First, only balanced workload conditions were

examined even though FLO allows the analyst to vary the arrival rate of voice (Class I) and data (Class II) traffic independently.

Secondly, FLO was not allowed to iterate until it identified the largest workload which could be accommodated by a given routing strategy. Rather, a more economical approach was taken, whereby each strategy was examined under several smoothly increasing workload settings. Kiemele's sensitivity analysis [64] demonstrated that the simulation model yields accurate results for workloads ranging from one to six Class I transactions per minute and from 100 to 600 Class II packet arrivals per second. Based on a desire to examine the alternative routing strategies over the widest possible conditions, it was decided that both Class I and Class II workload settings should be varied over their entire range. The requisite number of experimental runs was then reduced by spanning these ranges in increments of one Class I transaction per minute and 100 Class II packet arrivals per second. This resulted in six workload settings abbreviated 1/10 through 6/60.

Finally, workload settings which provided only marginal information were eliminated. This resulted in the elimation of the first two workload settings for all stategies (1/10 and 2/20) and the elimination of all workload settings for the random strategy. The result was

a significant reduction in the required number of simulation runs.

The final compromise called for the experimental runs to be limited to only one routing philosophy, even though FLO makes it possible to examine three different philosophies. The philosophy examined was "end-to-end" routing, where routes are determined based on an evaluation at the source node and then remain constant for the duration of the voice call or message. This philosophy is the one closest to the fixed routing stategy under which the network configuration was optimized via CIRPAC and thus represents the most difficult scenario under which to demonstrate the advantages of alternative routing strategies.

5.4 Experimental Data

This section contains a comparative analysis of the data collected during the experimental runs. Each subsection addresses a different perspective for comparing the data along with appropriate summary statistics. The complete set of collected data can be found in Appendix C.

5.4.1 Packet Delay

Chapter III motivates an analysis based on "user-visible" performance metrics. In the data transmission world, one of the most readily visible

metrics is packet delay. Two forms of this metric are available for consideration. The first, average packet delay, estimates the amount of time it will take a packet to traverse the network. The second metric could easily be termed the "aggravation factor" since it represents the percentage of packets which experience excessive delay. Though potentially more meaningful, the second form is less precise since the definition of excessive will often vary among users and even among applications.

5.4.1.1 Average Packet Delay

Figures 5-3 and 5-4 present the average packet delay (APD) statistics observed during the experimentation. The data is presented in both graphical and tabular form for each of the routing strategies examined.

There are three readily available points of reference from which to evaluate the observed data. The first is the performance observed for PRBK, Clabaugh's Progressive Alternate Routing strategy. This is the strategy around which the network configuration was optimized.

A second frame of reference is the design criteria used by CIRPAC in optimizing the network configuration. During optimization, CIRPAC used one (1) second as the maximum allowable delay.

A third basis for comparison is optimality, i.e. what is the best average packet delay statistic a user can

Average Packet Delay - Graphical

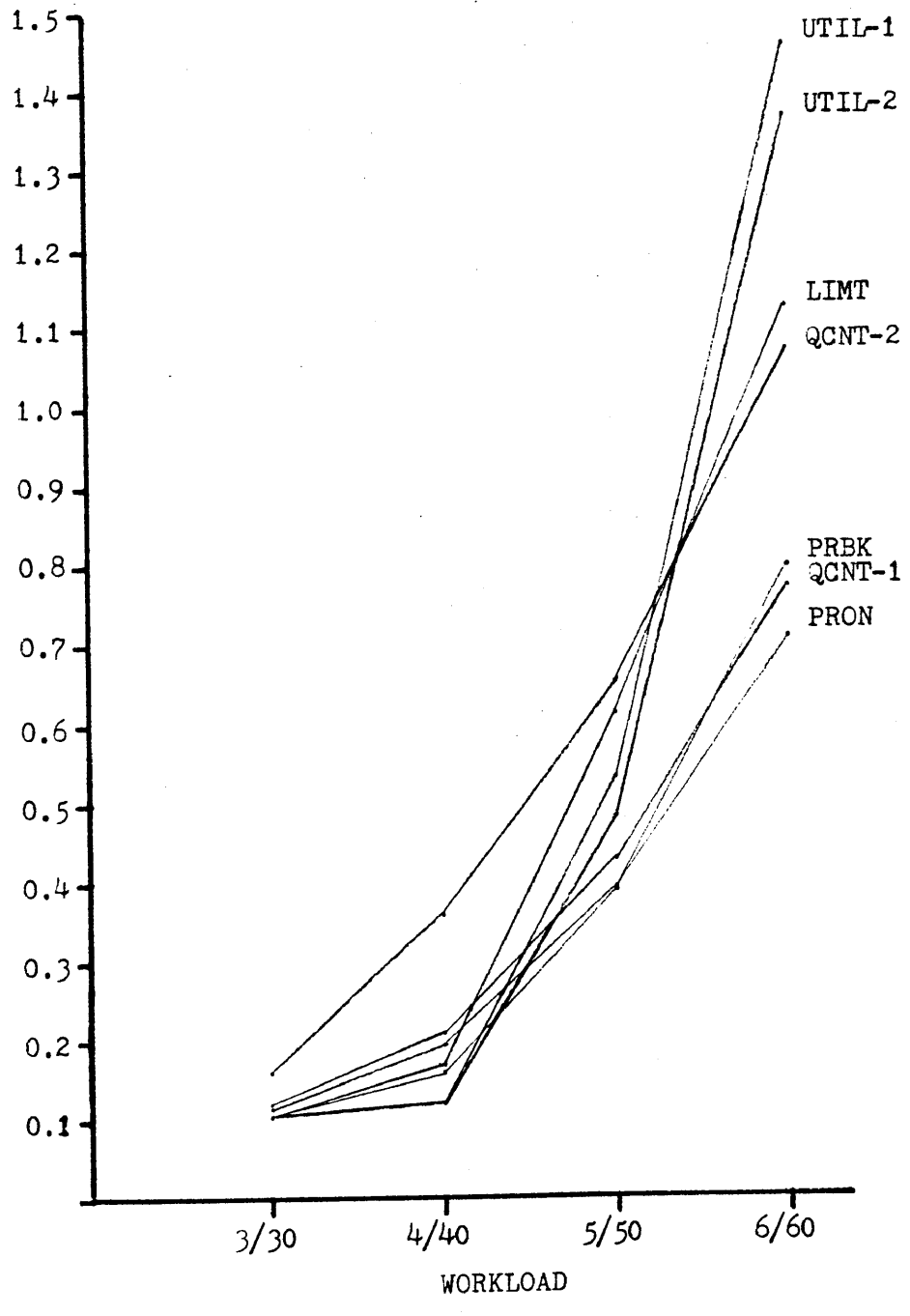


Figure 5-3

Average Packet Delay - Tabular

Strategy	Workload			
	3/30	4/40	5/50	6/60
PRBK	0.106	0.160	0.391	0.799
PRON	0.117	0.194	0.394	0.711
LIMT	0.105	0.172	0.616	1.131
QCNT-1	0.119	0.210	0.431	0.775
QCNT-2	0.158	0.361	0.655	1.073
UTIL-1	0.104	0.121	0.536	1.467
UTIL-2	0.104	0.119	0.485	1.375

Figure 5-4

expect. Appendix D contains the details of an analysis which demonstrates that, given the nodal processing times and the modelled network configuration, the best possible average packet delay statistic is 0.103.

At the initial workload setting, 3/30, the average packet delay observed for four strategies approaches the optimum. UTIL-1 and UTIL-2 both recorded the lowest packet delay at 0.104, only 1 percentage point over the optimum value. The metrics observed for two others, LIMT and PRBK, were also extremely close to the optimum with the former 2 percent above and the latter 3 percent above. The other strategies did not fare as well, recording average packet delays from 12% to 52% higher than that of UTIL-2.

For the next workload setting, the smallest average packet delay observed was that of UTIL-2 at 0.119, with UTIL-1 yielding approximately the same value at 0.121. The performance of the utilization strategies indicates an effective use of network resources since the 25% increase in workload only resulted in a 15% increase in average delay. This compares to a 50% increase for the next best strategy, PRBK, which recorded an average packet delay of 0.160. The four other strategies yielded values from 45% (LIMT) to 203% (QCNT-2) higher than that observed for UTIL-2.

At a workload setting of 5/50, PRBK demonstrated the best performance, recording an average packet delay of 0.391. It variant, PRON, yielded similar results with an observed metric of 0.393. The remaining strategies turned in performances ranging from 24% (UTIL-2) to 58% (QCNT-2) worse than that for PRBK.

For the final workload setting tested, PRON demonstrated its superiority with an observed packet delay of only 0.711. The next best statistic observed was that for QCNT-1, which was 9% higher at 0.775, followed by PRBK which was 13% higher at 0.799. These were the only strategies which satisfied the design criteria at this workload. LIMT, UTIL-1 and UTIL-2, which demonstrated excellent performance at lower workloads, all recorded delay statistics significantly larger than the one (1) second criterion.

5.4.1.2 Excessive Delay

The most readily available statistic for defining "excessive" is the one (1) second average packet delay criterion used by CIRPAC to optimize the network. Using this criterion as a basis for analysis, Figures 5-5 and 5-6 present in both graphic and tabular form the percentage of packets taking longer than 1 second to traverse the network.

Percentage Delay > One Second - Graphical

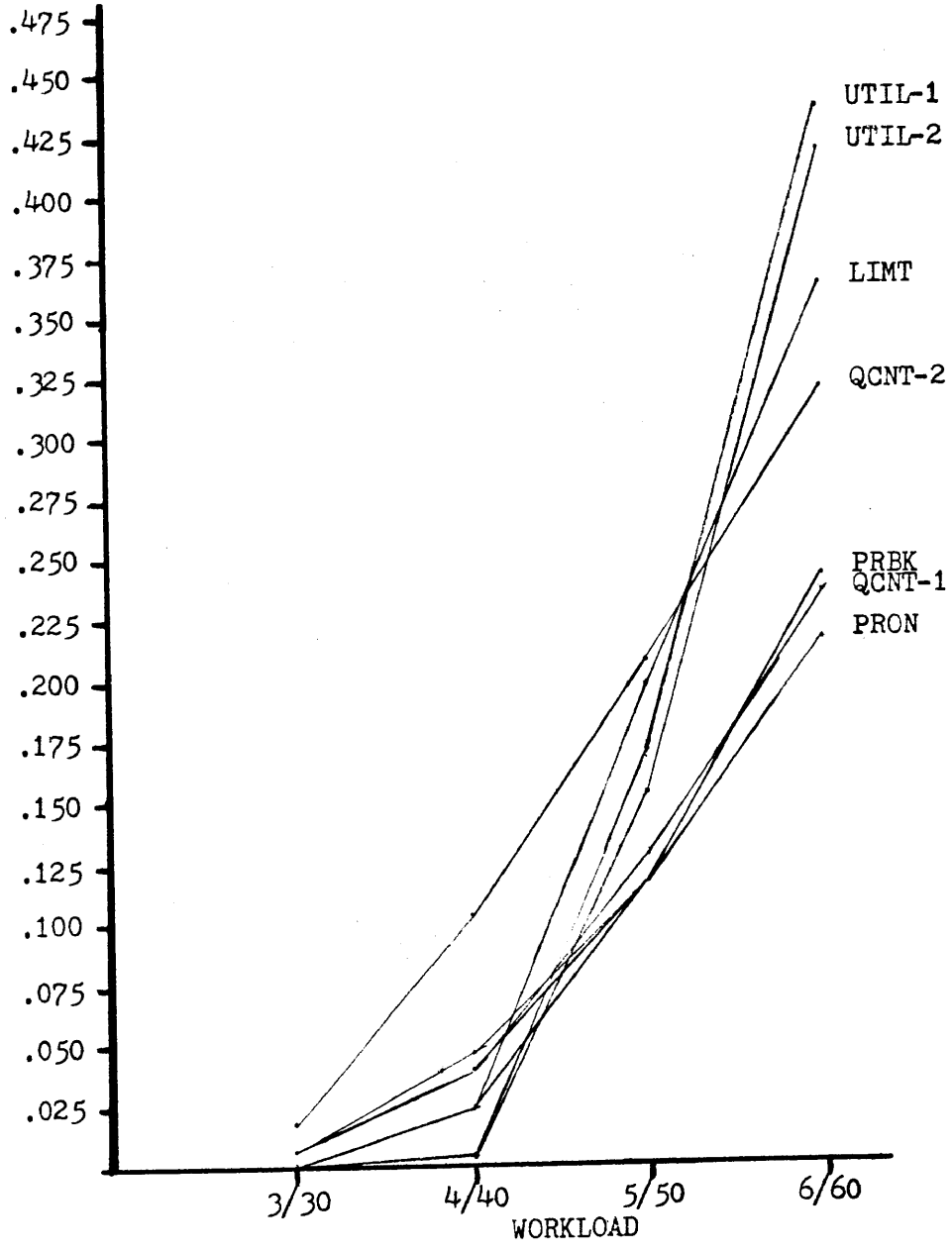


Figure 5-5

Percentage Delay > One Second - Tabular

Strategy	Workload			
	3/30	4/40	5/50	6/60
PRBK	0.001	0.025	0.116	0.243
PRON	0.007	0.040	0.116	0.216
LIMT	0.000	0.025	0.198	0.365
QCNT-1	0.007	0.046	0.129	0.237
QCNT-2	0.018	0.104	0.209	0.321
UTIL-1	0.000	0.005	0.171	0.438
UTIL-2	0.000	0.004	0.154	0.420

Figure 5-6

Review of the data shows that at the 3/30 workload setting, excessive packet delay is negligible with the worst case 1.8% and the majority of strategies recording no excessive delay at all. At the 4/40 workload setting the superiority of the utilization strategies begins to accentuate itself. At that setting, less that 1% of all packets experienced excessive delay (.4% and .5%) under the utilization strategies. By comparison, the next best performance was recorded by LIMT and PRBK at 2.5% followed by PRON and QCNT-1 at approximately 4%.

At the 5/50 workload setting, the best performance was jointly recorded by PRBK and PRON. Although a 12% statistic is not exceptional, it is clearly superior to the 15% and 17% figures experienced by the utilization strategies.

At the last workload level, PRON had the best performance just as it did with respect to average packet delay. Reporting an "aggravation factor" of 22%, this strategy performed at least 12% better than any other strategy and almost twice as well as UTIL-2.

5.4.2 Blocking Factor

Blocking factor is defined as the percentage of class I service requests which are rejected due to insufficient resources along the path selected by the routing strategy under consideration. This traffic class prohibits

queuing, therefore service requests cannot be stored until resources are freed. The analog of a class I rejection in the telephone network is a "busy signal".

The optimum value for blocking factor is 0, i.e. no class I service requests are rejected. Such a figure is highly optimistic and either indicates a large reserve of communications facilities or exceptional management of resources. In practice, a communications network would be designed to keep the blocking factor below some threshold. The threshold used by CIRPAC in optimizing the network was 10%.

Figures 5-7 and 5-8 summarize the blocking factors observed during the experimental runs. Review of the data clearly points out two facts. First, as would be expected, the blocking factor increases with traffic flow. Second, the observed data shows that an alternative strategy, specifically one based on utilization yields a significantly lower blocking factor.

At the 3/30 workload level, there is virtually no blocking. Four of the seven strategies under consideration observed .1% or less blocking. The other three strategies experienced between .6% and 1.5% blocking, significantly worse than the first four, but far less than the criteria for which the network was designed.

At the 4/40 workload setting, blocking is still not a significant problem, but the statistics observed for the

Blocking Factor - Graphical

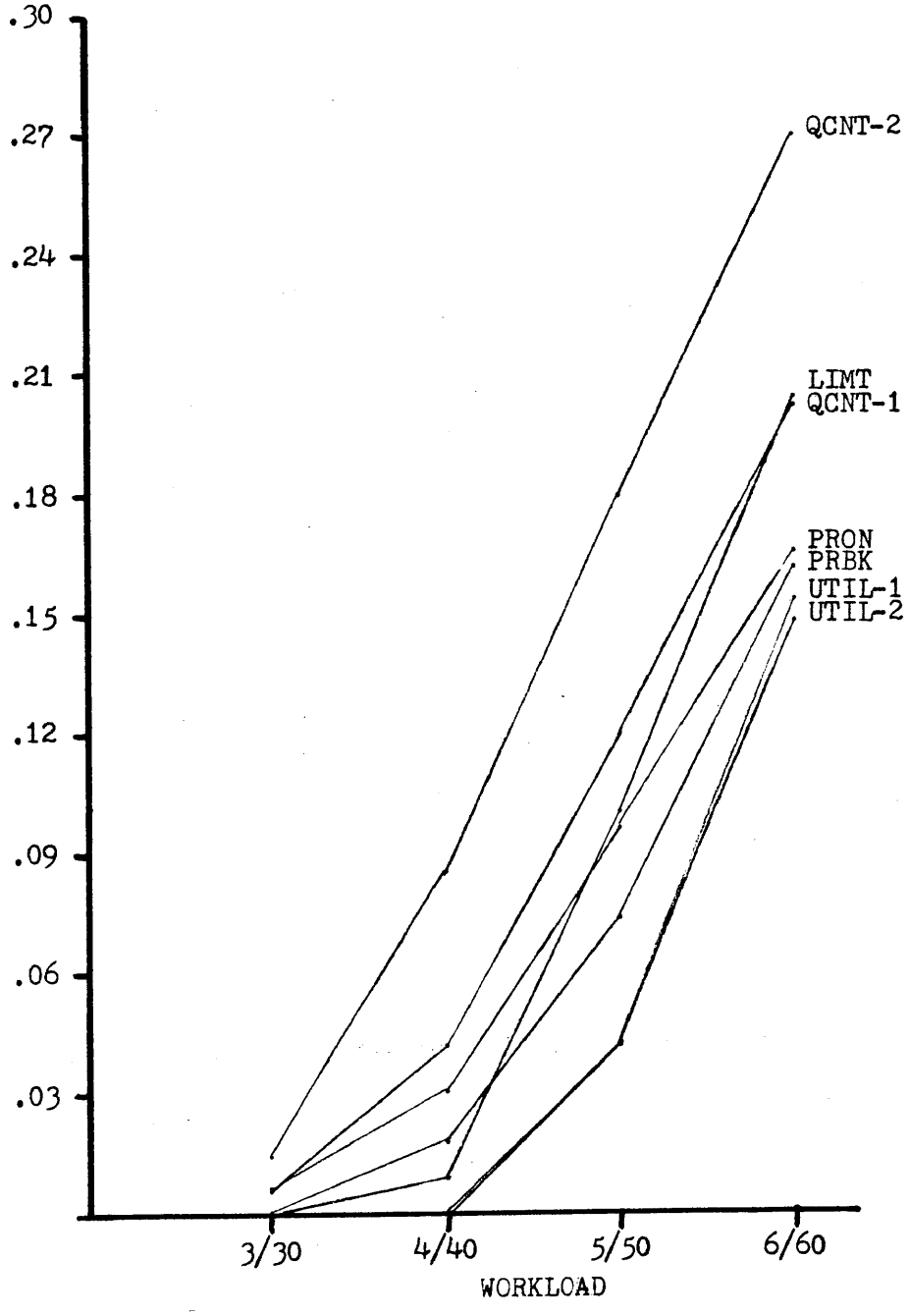


Figure 5-7

Blocking Factor - Tabular

Strategy	Workload			
	3/30	4/40	5/50	6/60
PRBK	0.001	0.018	0.074	0.162
PRON	0.007	0.031	0.097	0.166
LIMT	0.000	0.009	0.101	0.205
QCNT-1	0.006	0.042	0.120	0.203
QCNT-2	0.015	0.086	0.180	0.271
UTIL-1	0.000	0.000	0.043	0.154
UTIL-2	0.000	0.001	0.042	0.149

Figure 5-8

different routing strategies are diverging. For example, the blocking factor for the utilization strategies are still .1% or less, but that for PRBK is now 1.8% and that for QCNT-2 is at 86% of the design criterion.

At the 5/50 workload level, blocking has become a significant factor. At this level, the two queue-based strategies are well over the 10% design criterion(12% for QCNT-1 and 18% for QCNT-2) and two others are at the design criterion (LIMT and PRON). The blocking factor for PRBK has reached 0.074, approximately three-quarters of the design criterion,while the utilization strategies are at 4.2% and 4.3%, clearly superior to all other strategies.

At the 6/60 workload setting, all strategies have exceeded the network design criterion. At this workload, the most desireable blocking factor is demonstrated by the utilization strategies with both yielding approximately 15% blocking. PRBK and PRON experienced approximately 16% blocking at this workload. Blocking factors for the other three strategies were significantly higher.

5.3.3 Throughput

The third "user-visible" metric identified in Chapter III is throughput. Throughput is a measure of the volume of traffic which successfully traverses the network over a set time period. The emphasis here is on the word

"traverse" since the typical user only views the end-to-end results.

The throughput statistics collected during the experimental runs are presented in Figure 5-9. These statistics take two forms. The first, link throughput, must be cautiously interpreted since this figure reflects the average number of packets flowing through a node during a given time period. Though this value would rise with increased "end-to-end" throughput, it would also rise if the routing strategy under consideration selected extremely long routes.

The second form, message throughput, reflects the average number of messages (Class II transactions) which successfully traverse the network per second. Analysis of the data does not however, demonstrate significant differences between the alternative strategies.

5.4.4 Average Queue Size

As noted above, Chapter III motivates the notion that the performance of flow control mechanisms should be judged based on "user-visible" performance metrics. Diverging slightly from this concept, average queue size is examined. Though not directly available to the user, it is important since it is a key influences on packet delay and blocking factor.

Throughput

Workload	PRBK	PRON	LIMT	QCNT-1	QCNT-2	UTIL-1	UTIL-2	
3/30	643.	640.	649.	651.	743.	644.	646.	
4/40	868.	851.	967.	865.	982.	907.	907.	Link Thru-put
5/50	1,113.	1,064.	1,344.	1,080.	1,232.	1,327.	1,319.	
6/60	1,347.	1,275.	1,611.	1,295.	1,483.	1,658.	1,669.	
3/30	2,686.821	2,686.013	2,688.358	2,685.397	2,687.376	2,688.358	2,688.358	
4/40	3,580.856	3,582.875	3,583.154	3,966.497	3,579.925	3,582.536	3,582.297	Pkt Thru-put
5/50	4,474.478	4,474.840	4,475.972	4,472.276	4,473.924	4,471.745	4,475.481	
6/60	5,357,443	5,357.671	5,360.960	5,365.313	5,359.349	5,362.365	5,363.899	

Figure 5-6

Average Queue Size - Graphical

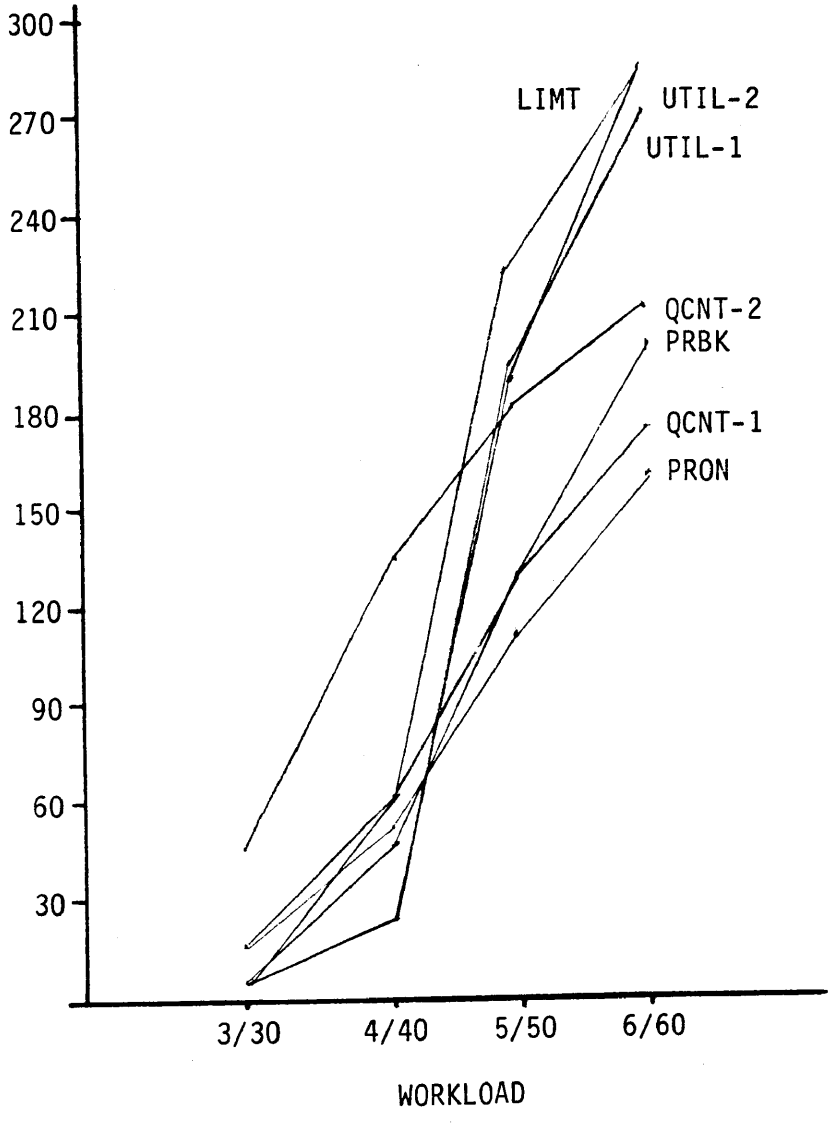


Figure 5-10

Average Queue Size - Tabular

Strategy	Workload			
	3/30	4/40	5/50	6/60
PRBK	5.4	44.0	129.9	201.5
PRON	16.0	52.3	108.7	157.5
LIMT	4.3	61.2	223.7	285.7
QCNT-1	15.7	63.3	128.3	173.3
QCNT-2	44.2	132.5	184.5	211.6
UTIL-1	4.3	19.7	195.4	273.9
UTIL-2	4.3	18.2	191.4	282.9

Figure 5-11

Review of the data presented in Figures 5-10 and 5-11 demonstrates that packet delay and average queue size are strongly related. At the lowest workload level, the minimal queueing found under four of the strategies is consistent wth the extremely low average queue size found under these strategies. The only strategy with a large queue size is QCNT-2 which was also noted earlier as having excessive packet delay statistics. This correlation is consistent for all workload levels tested.

5.5 Summary

The experimental effort described here produced a significant volume of technical data. This chapter described the origins of that data and summarized into a concise format, comparing the data observed for each of the alternative routing strategies investigated. The last chapter presents the conclusions which can be drawn from the experimental results described here.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

6.1 Overview

The introduction of economical voice digitization technologies combined with dramatic increases in the speed of electronic switching systems has opened the door to the creation of an environment long recognized for its characteristic advantages, the "integrated" communications network. Such an environment integrates both voice and data traffic over the same communications network, allowing the implementor to reap the benefits of equipment commonality, more efficient resource utilization and streamlined operations, maintenance and administration. Recognizing the substantial advantages of an integrated communications environment and technology's relentless progress toward it, researchers have examined the benefits of alternative network architectures. The conclusion reached by many is that the Slotted Envelope Network (SENET) proposed by Coviello and Vena is one of the most promising approaches offered thus far. Researchers have also investigated the critical issues of optimal topological design and link capacity asssignment. Thus, previous research provides the tools necessary to design and implement an integrated communications network. This

research accepts the premise that a SENET-type integrated environment will eventually become commonplace.

Once built, such a network will have a built-in limit to the amount of traffic it can carry. If demand should ever exceed this limit, data packets and/or voice calls will be delayed or rejected. Any network experiencing this condition is said to be "congested".

Clearly, congestion could be eliminated if enough resources were dedicated to the communications network. Such a network would, however, be prohibitively expensive. Realistically, computer-communications networks must be designed to accommodate peak traffic requirements and absorb reasonable load fluctuations, but must do so within the constraints of cost-effective operation. However, even in networks designed with substantial excess capacity, higher than expected traffic demands, unfavorable load patterns, component failures or any combination thereof, can create situations where the network becomes congested. Whatever the reason, congestion is clearly unacceptable and procedures must be developed to prevent or at least forestall this condition. These procedures are generally referred to as "flow control strategies".

This research investigates the utility of routing as a mechanism for implementing the network flow control strategy. Routing policy determines the path information

(either voice or data) will follow in traversing the network. This research effort exploits the notion that congestion results when an excessive volume of traffic is channeled down a given communications link. It investigates the characteristics of alternative routing strategies with respect to preventing and/or forestalling congestion in a SENET-type network environment.

Specifically, this research effort poses the following research question:

Given: (1) a SENET-type integrated circuit/packet switched communications environment exists, and

(2) that this environment was initially optimized to a specific traffic flow pattern using progressive alternate (fixed) routing.

Question: Can an alternative routing strategy be identified which reduces congestion as offered workload varies?

Pursuit of this question requires a clear definition of the phrase "reduced congestion" and identification of a procedure for measuring it. This research opts to view these terms from a user's viewpoint.

From a user perspective, the details of how data and voice messages are transmitted are unimportant. The user is concerned only with two things: the cost to build the

communications network and the performance it offers. These two features, unfortunately, are usually in direct conflict with the typical user striving to minimize cost while staying within an acceptable performance threshold. At some point, however, cost/benefit studies are completed, management decisions are made, and the network resources are acquired. From that point on, the user's only concern is end-to-end performance with performance typically being measured in terms of volume or response time. Common measures include:

(1) Throughput - the number of messages or packets which successfully traverse the network during a given time quantum.

(2) Average message delay - the average length of time it takes for a message or packet to traverse the network.

(3) Blocking - the percentage of time voice calls are attempted but cannot be completed because resources are unavailable.

These metrics provide the user with a quantitative assessment of how well the network satisfies his communications requirements. They are the ones he must contend with on a daily basis and are the ones fostered by this research as most appropriate for measuring network performance.

Congestion in the integrated environment is thus be defined as a network condition where throughput and/or average message delay rise above a specified standard or a condition where blocking rises above a specified acceptable minimum. The specific values for the performance standards are established by management and form a key design constraint for the network.

Simulation was the vehicle selected for investigating the research question. A FORTRAN-based simulator which models the integrated SENET environment was developed and then used to examine the effect of seven alternative routing strategies on network congestion. The strategies were examined at progressively increasing workloads against a sample network which was based on the CYBERNET system. The sample network was originally optimized via CIRPAC, a network topology optimization tool which assumes the use of progressive alternate (fixed) routing.

The experimental results obtained were examined from two perspectives. The first one, proximity of observed metrics to "optimality", provides the reader with insight into how well each strategy performs with respect to a network blessed with unlimited communications resources. The second perspective, proximity of observed metrics to that experienced by progressive alternative routing, provides the reader with a comparison of how well each strategy performs relative to the one for which the

network was originally optimized.

6.2 Conclusions

The results obtained via experimentation conclusively demonstrate that the research question should be answered affirmatively. The results clearly show that there are routing strategies which reduce congestion as workload is varied. Unfortunately, the results also show that none of the strategies investigated yield consistently superior performance over all workloads. Rather, it appears that the routing strategy should be adjusted as workload varies. Finally, the data indicates that a strategy based on link utilization outperforms other strategies in most cases. Specifically, the following conclusions can be drawn.

Based on a throughput analysis, all strategies performed essentially the same. Though, throughput for the utilization strategies, UTIL-1, UTIL-2 and LIMT, were slightly better than that for other strategies, the percentage improvement demonstrated was so small that any claim to congestion reductions is at best dubious.

Based on an analysis of the blocking characteristics demonstrated by the alterative strategies, UTIL-2 is clearly the superior strategy. At all workload levels this strategy resulted in significantly reduced blocking. At the final workload level, the benefits of UTIL-2 appear

to fade, as the blocking metrics converge. However, even at this workload, UTIL-2 offers substantially less blocking than other alternatives considered.

If packet delay is the figure of merit to be used in judging network performance, the recommended routing strategy must change with workload. Below the 5/50 setting, UTIL-2 is clearly superior. However, at 5/50 and 6/60, PRON yields superior performance. These results are consistent for both excessive delay and average packet delay. The results are further supported by the average queue size statistics gathered.

6.3 Recommendations for Future Research

The stated goal of this project was to determine whether routing could be used to reduce congestion in an integrated circuit/packet-switched computer communications network. This objective was clearly satisfied with the affirmative answer to the research question posed. In the course of addressing that question an extremely powerful experimentation facility was produced. This facility opens the way for further investigation into the characteristics of the integrated environment. This researcher recommends that the following areas be pursued in future efforts.

(1) Queue priority schemes. One of the simplifying assumptions of FLO is that queues will always be handled first-in, first-out. There are applications, such as military command and control systems, where this is not only undesirable but potentially disastrous. The effect of queue prioritization on network performance should be addressed.

(2) Security issues. There are no provisions in the current model to effect network security except through encrytion of data. The network model further assumes that data can and will flow between any nodal pair and can take any route. Transmission of classified data often must be accomplished without encrytion. Known as "RED" networks, various techniques are used to insure that data is not tapped from the transmission media. The end result is a system which funnels specific data classes over specialized links, i.e. those which are suitably protected. The effect of security issues on network performance is an increasingly relevant topic.

(3) Routing disciplines. This project merely scraped the surface of the capabilities of FLO. One of the major capabilities not fully investigated, but deserving attention, is the effect of alternative routing disciplines on network performance.

(4) Strategy Adjustment. This research effort concluded that to keep congestion minimal, the routing strategy must be adjusted as workload increases. A key topic for future exploration centers around the issues associated with altering the routing strategy. Open questions include: When should the strategy be changed? How often should changes occur? How long will the system take to stabilize? Will performance be degraded in the interim? Each of these questions merit further study.

6.4 Summary

The clear advantages of an integrated communications environment have motivated a general consensus among industry and government experts that future communications systems will combine voice and data traffic over the same network. Technology is steadily progressing toward the realization of this environment. The research summarized

here extends previous efforts by investigating the use of routing as a flow control strategy within the integrated environment, concluding that congestion can be reduced by the judicious selection of routing strategies.

REFERENCES

1. Agnew, C.E. On quadratic adaptive routing algorithms. Communications of the ACM 19. 1 (January 1976), 118-122.

2. Ahuja, V. Design and Analysis of Computer Communication Networks. McGraw-Hill, Inc., New York, NY, 1982.

3. Andrews, Frank B., and Cooper, Chris G. Probing NCR's distributed network architecture. Data Communications. (April 1978), 49-59.

4. Avellaneda, O.A., Hayes, J.R., and Nassehi, M.M. A Capacity Allocation Problem in Voice-Data Networks. IEEE Transactions on Communications (Com-30), 7 (July 1982), 1967-1772.

5. Bell, G.C. More power by networking. IEEE Spectrum 11, 2 (February 1974), 40-45.

6. Belsnes, D. Flow control in the packet switching networks. Communications Networks, (1975), 349-361.

7. Berberi, N. VLSI pares T1 problems down to size. Data Communications, 7 (June 1984), 133-140.

8. Bially, T., and McLaughlin, A.J. Voice communications in integrated digital voice and data networks. IEEE Transactions on Communications (Com-28), 9 (September 1980), 1478-1488.

9. Black, P. How ISDN services could make or break the big network. Data Communications, 7 (June 1984), 247-252.

10. Boehm, B.W., and Mobley, R.L. Adaptive routing techniques for distributed communications systems. IEEE Transactions on Communications (Com-17), 3 (June 1969), 340-349.

11. Boorstyn, R.R. and Frank, H. Large-scale network topological optimization. IEEE Transactions on Communications (Com-25), 1 (January 1977), 29-47.

12. Boorstyn, R.R., and Livne, A. A Technique for Adaptive Routing in Networks. IEEE Transactions on Communications (Com-29), 4 (April 1981) 474-480.

13. Bourgonje W. Twisted-pair bus carries speech, data, text and images. Electronic Design, (July 26, 1984), 171-178

14. Chatterjee, A., Georganas, N.D., and Verma, P.K. Analysis of a packet-switched network with end-to-end congestion control and random routing. IEEE Transactions on Communications (Com-23), 12 (December 1977), 1485-1489.

15. Chou, W. ACK/TOPS - an integrated network design tool. 1981 IEEE International Conference on Communications (ICC-81), Denver, CO (June 14-18, 1981), 4.1.1-4.1.7.

16. Chou, W. (Ed.). Computer Communications, Volume I[Principles. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

17. Chou, W., Bragg, A.W., and Nilsson, A.A. The need for adaptive routing in the chaotic and unbalanced traffic environment. IEEE Transactions on Communications (Com-29), 4 (April 1981), 481-490.

18. Chou, W., Nilsson, A.A., and Bragg, A.W. The need for dynamic routing in a network spanning several time zones. 1981 IEEE National Telecommunications Conference, New Orleans, LA, (November 29 - December 3), 3.1.1-3.1.5.

19. Chou, W., and Sapir,D. A generalized cut-saturation algorithm for distributed computer communications network optimization. IEEE 1982 International Conference on Communications (ICC-82), Philadelphia, PA (June 13-17, 1982), 4C.2.1-4C.2.6

20. Chu, P.H.N., Boorstyn, R.R., and Kershenbaum, A. A simulation study of a dynamic routing scheme. 1981 IEEE National Telecommunications Conference, New Orleans, LA, (November 29 - December 3), 3.4.1 - 3.4.11.

21. Cicchetti, G.B., and Lubarsky, A.R. Hybrid integrated digital network. World Telecommunications Forum, Geneva, Switzerland, (1975), 2.3.7.1-2.3.7.5.

22. Clabaugh, C.A. Analysis of flow behavior within an integrated computer-communication network. Ph.D. dissertation, Texas A&M University (May 1979).

23. Coviello, G.J., and Lyons, R.E. Conceptual approaches to switching in future military networks. IEEE Transactions on Communications (Com-28), 9 (September 1980), 1491-1498.

24. Coviello, G.J., and Vena, P.A. Integration of circuit/packet switching by a SENET (Slotted Envelope Network) concept. 1975 IEEE National Telecommunications Conference (NTC-75), New Orleans, LA (December 1-3, 1975), 42.12-42.17.

25. Cravis, H. Communications Network Analysis. D.C. Heath and Co., Lexington, MA, 1981.

26. Dahlbom, C.A., and Ryan, J.S. History and description of a new signalling system. Bell System Technical Journal, 2 (February 1978), 225-250.

27. Davies, D.W., and Barber, D.L.A. Communications Networks for Computers. John Wiley and Sons, London, England, 1973.

28. Dejean, J.H., and Campagno, H. Packet switching multiservice network. 1981 International Switching Symposium, Montreal, Canada (September 21-25, 1981), Session 32-C, Paper 1.

29. Dysart, H., Krone, M., and Fielding, J. Integrated voice/data private network planning. 1981 IEEE International Conference on Communications (ICC-81), Denver, CO (June 14-18, 1981), 4.2.1-4.2.5.

30. Elovitz, H.S., and Heitmeyer, C.L. What is a computer network? 1974 IEEE National Telecommunications Conference (NTC-74), San Diego, CA (December 2-4, 1974), 1007-1014.

31. Esterling, R., and Hahn, P. A comparison of digital data network switching alternatives. 1975 IEEE National Telecommunications Conference (NTC-75), New Orleans, LA (December 1-3, 1975), 42.8-42.11.

32. Farber, D.J. Networks: an introduction. Datamation 18, 4 (April 1972), 36-39.

33. Forgie, J.W. Voice conferencing in packet networks. 1980 IEEE International Conference on Communications (IEEE-80), Seattle, WA (June 12-18,1980), 21.3.1-21.3.4.

34. Frank, H. Plan today for tomorrow's data/voice nets. Data Communications 7, 9 (September 1978), 51-62.

35. Fultz, G.L. Adaptive routing techniques for message switching computer-communications networks, Ph.D. Disseration, University of California, Los Angeles, CA, June 1972, 256-257.

36. Fultz, G.L., and Kleinrock, L. Adaptive routing techniques for store-and-forward computer communications networks. 1971 IEEE International Conference on Communications (ICC-71), Montreal, Canada (June 14-16, 1971), 39.1-39.8.

37. Gallagher, R.G. A minimum delay routing algorithm using distributed computation. IEEE Transactions on Communications (Com-25), 5 (May 1977), 73-85.

38. Gallagher, R.G. Distributed network optimization algorithms. IEEE 1979 International Conference on Communications (ICC-79), Boston, MA (June 10-14, 1979), 43.2.1-43.2.2.

39. Gerla, M., and Chou, W. Flow control strategies in packet switched computer networks. 1974 IEEE National Telecommunications Conference (NTC-74), San Diego, CA (December 2-4, 1974), 1032-1037.

40. Gerla, M., and Kleinrock, L. On the topological design of distributed computer networks. IEEE Transactions on Communications (Com-25), 1 (January 1977),48-60.

41. Gerla, M., and Mason, D. Distributed routing in hybrid packet and circuit data networks. IEEE Conference on Computer Communications Networks (COMPCON-78), Washington, DC (September 5-8, 1978), 125-131.

42. Gitman, I., Hsieh, W., and Occhiogrosso, B.J. Analysis and design of hybrid switching networks. IEEE Transactions on Communications (Com-29), 9 (September 1981), 1290-1300.

43. Gitman, I., Occhiogrosso, B.J., Hsieh, W., and Frank, H. Sensitivity of integrated voice and data networks to traffic and design variables. Sixth IEEE Data Communications Symposium, Pacific Grove, CA (November 1979), 181-192.

44. Gordon, R.D., Alles, H.G., and Bergland, G.D. An experimental digital switch for data and voice. 1981 International Switching Symposium, Montreal, Canada (September 21-25, 1981), Session 21-B, Paper 3.

45. Greene, W.H. Optimal routing within large scale distributed computer-communications networks. Ph.D. dissertation, Texas A&M University (May 1978).

46. Gross, D., and Harris, C.M. Fundamentals of Queueing Theory. John Wiley & Sons, Inc., New York, NY, 1974.

47. Gruber, J.G. Delay related issues in integrated voice and data networks. IEEE Transactions on Communications (Com-29), 6 (June 1981), 786-800.

48. Gruber, J.G., and Strawczynski, L. Subjective effects of variable delay and speech clipping in dynamically managed voice systems. IEEE Transactions on Communications (Com-33), 8 (August 1985), 801-808.

49. Haenschke, D.G., Kettler, D.A., and Oberer, E. Network management and congestion in the U.S. telecommunications network. IEEE Transactions on Communications (Com-29), 4 (April 1981), 376-385.

50. Hasegawa, Hideo, Miyahara, Teshigawara, Tushihara, and Yoshimi. A comparative evaluation of switching methods in computer communications networks. 1975 IEEE International Conference on Communications (ICC-75), San Francisco, CA (June 16-18, 1975), 6.6-6.10.

51. Heggestad, H. M. An overview of packet-switching communications. IEEE Communications Magazine, 4 (April, 1984), 24-31.

52. Hilal, W., and Liu, M.T. Local area networks supporting speech traffic. Computer Networks, 8 (August 1984), 325-337.

53. Hiramatsu, Y., Mase, K, and Kajiwara, M. A packet transfer control method using circuit switching function. Electronics and Communications in Japan, 7 (July 1983), 44-53.

54. Hoard, B. Integrating voice and data - sharing the lines. Computerworld 15, 52 (December 28, 1981), 33-35.

55. Hsieh, W., and Gitman, I. How good is your network routing protocol? Data Communications, (May 1984), 231-248.

56. Hsieh, W., and Gitman, I. How to prevent congestion in computer networks. Data Communications, 7 (June 1984), 209-216.

57. Hsieh, W., and Gitman, I. Routing strategies in computer networks. Computer, June 1984, 46-56.

58. Hsieh, W., Gitman, I., and Occhiogrosso, B.J. Design of hybrid-switched networks for voice and data. IEEE 1978 International Conference on Communications (ICC-78), Toronto, Canada (June 4-7, 1978), 20.1.1-10.1.9.

59. Ilyas, M., and Mouftah, H.T. Quasi cut-through: new hybrid switching technique for computer communication networks. IEE Proceedings, Part. E, 1 (January 1984) 1-8.

60. James, R.T., and Muench, P.E. AT&T facilities and services. Proceedings of IEEE-60, 11 (November 1972), 1342-1349.

61. Janakiraman, N., Pagurek, B., and Neilson, J.E. Performance analysis of an integrated switch with fixed or variable frame rate and movable voice/data boundary. IEEE Transactions on Communications (Com-32), 1 (January 1984), 34-39.

62. Joffe, J.M., and Moss, F.H. A responsive distributed routing algorithm for computer networks. IEEE Transactions on Communications (Com-30), 7 (July 1982), 1758-1762.

63. Kermani, P., and Kleinrock, L. A tradeoff study of switching systems in computer communication networks. IEEE Transactions on Computers (Com-29), 12 (December 1980), 1052-1060.

64. Kiemele, M. Adaptive topological configuration of an integrated circuit/packet-switched computer network. Ph.D. Dissertation, Texas A&M University, 1984.

65. Kimbleton, S.F., and Schneider, M.G. Computer communication networks: Approaches, objectives, and performance considerations. ACM Computing Surveys, 3 (Sept. 1975), 129-179.

66. Kleinrock, L. Analytic and simulation methods in computer network design. 1970 AFIPS Spring Joint Computer Conference (SJCC), Atlantic City, NJ (May 5-7, 1970, 569-579.

67. Kleinrock, L. A decade of network development. Journal of Telecommunications Networks, 1 (Spring, 1982), 1-11.

68. Kleinrock, L., and Kamoun, F. Optimal clustering structures for hierarchical topological design of large computer networks. Networks 10, 3 (Fall 1980),221-248.

69. Komatsu, M., and Mayeda, W. Usage Characteristics of detour routes in store-and-forward switching networks. Electronics and Communications in Japan, 1 (January 1983), 78-85.

70. Konheim, A.G., and Pickholtz, R.L. Analysis of integrated voice/data multiplexing. IEEE Transactions on Communications (Com-32), 2 (February 1984) 140-147.

71. Kozicki, Z., and McGregor, P.V. An approach to computer-aided network design. IEEE 1981 International Conference on Communications (ICC-81), Denver, CO (June 14-18, 1981, 4.4.1-4.4.7.

72. Li, S., and Majithia, J.C. Performance analysis of a DTDMA local area network for voice and data. Computer Networks, 8 (August 1984), 81-91.

73. Li, S., and Mark, J.W. Performance of voice/data integration in a TDM system. IEEE Transactions on Communications (Com-33), 12 (December 1985), 1265-1273

74. Mathison, S.L., and Walker, P.M. Regulatory and economic issues in computer communications. Proceedings of IEEE-60, 11 (Nov. 1972), 1254-1272.

75. McAuliffe, D.J. An integrated approach to communications switching. 1978 IEEE International Conference on Communications (ICC-78), Toronto, Canada (June 4-7, 1978), 20.4.1-20.4.5.

76. McDonald, J. C. Local digital switching - a successful new technology. Telecommunications 10, 4 (April 1976), 43-48.

77. McQuillan, J.M. Adaptive routing algorithms for distributed computer networks. Report AD-781467, NTIS (May 1974).

78. McQuillan, J.M. Routing algorithms for computer networks - a survey. 1977 IEEE National Telecommunications Conference (NTC-77), (December 1977).

79. Metcalfe, R.M. Packet communications. Report AD-771-430, NTIS (Dec. 1973).

80. Niznik, C.A. Performance evaluation of the computer network dynamic congestion table algorithm. IEEE Transactions on Communications (Com-33), 2 (February 1984), 150-159.

81. Occhiogrosso, B.J., Gitman, I., Hsieh, W., and Frank, H. Performance analysis of integrated switching communications systems. 1977 National Telecommunications Conference (NTC-77), Los Angeles, CA (December 5-7, 1977), 12:4.1-12:4.13.

82. Ogino, N., Numao, M., Saito, T., and Inose, H. Design of time division switching networks considering amount of control. Electronics and Communications in Japan, 3 (March 1983), 43-51.

83. Okada, H. Delay behavior of data traffic in an integrated voice/data multiplex structure: multi-capacity-limits (MCL) property. IEEE Transactions on Communications (Com-34), 3 (March 1986), 300-307.

84. Ozarow, L., and DeRosa, J. A combined packet and circuit-switched processing satellite system. IEEE 1979 International Conference on Communications (ICC-79), Boston, MA (June 10-14, 1979), 24.5.1-24.5.5.

85. Paoleth, L.M. AUTODIN. Computer Communications Networks, Noordhoff International Publication, Leyden, The Netherlands, 1975, 345-372.

86. Phillips, D.T., and Garcia, A. Fundamentals of Network Analysis. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

87. Pitroda, S.G. A review of telecommunications switching concepts - part two. Telecommunications 10, 3 (March 1976), 24-30.

88. Pooch, U.W., Greene, W.H., and Moss, G.G. Telecommunications and Networking. Little, Brown, and Company, Boston, MA, 1983.

89. Prosser, R.T. Routing procedures in communications networks - part I: Random procedures. IRE Transactions on Communications Systems (December 1962), 322-329.

90. Prosser, R.T. Routing procedures in communications networks - part II: Directory procedures. IRE Transactions on Communications Systems (December 1962), 329-335.

91. Roberts, L.G. The evolution of packet switching. Proceedings of IEEE-66, 11 (November 1978), 1307-1313.

92. Rosner, R.D. Packet switching and circuit switching: a comparison. 1975 IEEE National Telecommunications Conference (NTC-75), New Orleans, LA, (December 1-3, 1975), 42.1-42.7.

93. Ross, M.J. System engineering of integrated voice and data switches. 1978 IEEE International Conference on Communications (ICC-78), Toronto, Canada (June 4-7, 1978), 20.5.1-20.5.4.

94. Ross, M.J. Alternatives for integrating voice and data. 1981 International Switching Symposium (ISS-81), Montreal, Canada, (September 21-25, 1981), Session 41-B, Paper 4.

95. Ross, M.J., and Mawafi, O.A. Performance analysis of hybrid (circuit/packet) switching concepts. 1981 IEEE National Telecommunications Conference (NTC-81), New Orleans, (November 29 - December 4, 1981), 4.2.1-4.2.5.

96. Ross, M.J., and Sidlo, C. Approaches to the integration of voice and data telecommunications. 1979 IEEE National Telecommunications Conference (NTC-79), (November 1979).

97. Ross, M.J., Tabbot, A.C., and Waite, J.A. Design approaches and performance criteria for integrated voice/data switching. Proceedings of IEEE-65, 9 (September 1977), 1283-1295.

98. Rudin, H. On routing and "delta-routing": a taxonomy and performance comparison of techniques for packet switched networks. IEEE Transactions on Communications (Com-24), 1 (January 1976).

99. Rudin, H. Studies on the integration of circuit and packet switching. 1978 IEEE International Conference on Communications (ICC-78), Toronto, Canada (June 4-7, 1978), 20.2.1-20.2.7.

100. Rudov, M.H. Marketing ISDNs: reach out and touch someone's pocketbook. Data Communications, 7 (June 1984), 239-245.

101. Saint-Remi, J. The many sounds of voice digitization. Data Communications, 1 (January 1984), 169-170.

102. Schneider, G.M. The VANS system. 1978 IEEE Conference on Computer Communications Networks (COMPCON 78), Washington, DC (September 5-8, 1978), 166-174.

103. Schneider, K.S. Integrating voice and data on circuit-switched networks. IEEE Transactions on Aerospace Electronic Systems (AES-15), 4 (July 1979), 481-493.

104. Schwartz, M. Computer-communication network design and analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

105. Schwartz, M., and Cheung, C.K. The gradient projection algorithm for multiple routing in message-switched networks. IEEE Transactions on Communications (Com-24), 4 (April 1976), 449-456.

106. Schwartz, M., and Soad, S. Analysis of congestion control techniques in computer communication networks. Proceedings on flow control in computer networks, IFIP North-Holland, 1979, 113-130.

107. Segall, A. Advances in verifiable fail-safe routing procedures. IEEE Transactions on Communications (Com-29), 4 (April 1981), 491-497.

108. Sharma, R.L., de Sousa, P.T., and Ingle, A.D. Network Systems. Van Nostrand Reinhold Company, New York, NY, 1982.

109. Sproule, D.E., and Mellor, F. Routing, flow, and congestion control in the DATAPAC network. IEEE Transactions on Communications (COM-29), 4 (April 1981).

110. Takehiko, Y., and Shimasaki, N. A study of future integrated service digital networks. 1975 IEEE National Telecommunications Conference (NTC-75), New Orleans, LA (December 1-3, 1975), 7.1-7.6.

111. Tanenbaum, A.S. Computer Networks. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

112. Tanno, K., Hidekazu, T., Nakamura, T., and Sato, R. A flow control analysis based on a measure of power in packet switching networks. 1981 IEEE National Telecommunications Conference (NTC-81), New Orleans, LA (November 29 - December 3, 1981), E5.7.1-E5.7.6.

113. Thaker, G.H., and Cain, J.B. Interactions of routing and flow control. IEEE Transactions on Communications (Com-34), 3 (March 1986), 269-277.

114. Thurber, K.J. Circuit switching technology: a state-of-the-art survey. 1978 IEEE Conference on Computer Communications Networks (COMPCON 78), Washington, DC (September 5-8, 1978), 116-124.

115. Tymes, L.R.W. Routing and flow control in tymnet. IEEE Transactions on Communications (Com-29), 4 (April 1981), 392-398.

116. Wang, J.W., Queueing network modeling of computer communications networks. Computing Surveys, 10, 3 (September 1978), 343-352.

117. Weinstein, C., McLaughlin, A., and Bially, T. Efficient multiplexing of voice and data in integrated digital networks. 1980 IEEE International Conference on Communications (ICC-80), Seattle, WA (June 8-12, 1980), 21.1.1-21.1.7

118. Worley, A.R. The DATRAN system. Proceedings of IEEE-60, 11 (November 1972), 1357-1368.

119. Wunderlich, E.F. An analysis of dynamic virtual circuit routing. 1981 National Telecommunications Conference (NTC-81), New Orleans, LA, (November 29 - December 4, 1981), 3.3.1-3.3.6.

120. Yum, T. The design and analysis of a semidynamic deterministic routing rule. IEEE Transactions on Communications (Com-29), 4 (April 1981), 498-504.

121. Yum, T., and Schwartz, M. The join-biased queue rule and its application to routing in computer communication networks. IEEE Transactions on Communications (Com-29), 4 (April 1981), 505-511.

122. Yum, T.S., and Schwartz, H. Comparison of adaptive routing algorithms in computer communication networks. 1978 IEEE National Telecommunications Conference (NTC-78), Birmingham, AL.

APPENDIX A

SOURCE LISTING FOR FLO

The following pages contain the source listing for the enhanced simulator, FLO. FLO is implemented in VS/FORTRAN on the AMDAHL 470 at Texas A&M University.

```
C     ************************************************************
C     *                        FLO                               *
C     *  THIS IS THE MAIN ROUTINE FOR A SIMULATION TO STUDY THE  *
C     *  EFFECT OF ROUTING STRATEGIES ON FLOW IN AN INTEGRATED   *
C     *  CIRCUIT/PACKET SWITCHED NETWORK ENVIRONMENT.            *
C     ************************************************************
C
C
C
      INTEGER IPASS,OVRLOD,STABLE
C
C
C     ************************************************************
C     *  SET THE FLAG VALUES NEEDED TO CONTROL REPITITIONS.      *
C     ************************************************************
C
      IPASS=1
      STABLE=1
      OVRLOD=0
C
C     ************************************************************
C     *  CALL THE MAJOR MODULES ITERATIVELY UNTIL ONE OF THREE   *
C     *  TERMINATION CONDITIONS IS REACHED.  TERMINATION CONDITIONS*
C     *  INCLUDE:                                                *
C     *    (A) TOO MANY ITERATIONS (OVER 20 PASSES).             *
C     *    (B) NETWORK WILL NOT STABILIZE (REACH STEADY STATE).  *
C     *    (C) NETWORK HAS EXCEEDED MINIMUM PERFORMANCE CRITERIA.*
C     ************************************************************
C
      CALL DEFMOD
      CALL DEFRTE
  100 CALL SIMINT
      CALL SIMULA(IPASS,STABLE,OVRLOD)
      STOP
  200 IPASS=IPASS+1
      IF (IPASS.GT.20) GOTO 999
      IF (OVRLOD.EQ.1) GOTO 998
      IF (STABLE.EQ.1) GOTO 100
      WRITE(6,1000) IPASS
      STOP
  998 WRITE(6,1050) IPASS
      STOP
  999 WRITE(6,1100)
      STOP
C
C
 1000 FORMAT (' SIMULATION HAS BECOME UNSTABLE AT IPASS=',I2)
 1050 FORMAT (' NETWORK HAS BECOME OVERLOADED AT IPASS=',I2)
 1100 FORMAT (' SIMULATION PASS LIMIT REACHED')
C
C
      END
```

```
C     ************************************************************
C     * SUBROUTINE DEFMOD WILL READ IN THE VALUES NEEDED TO DEFINE *
C     * NETWORK CONFIGURATION.  THESE VALUES WILL NOT CHANGE FROM  *
C     * ITERATION TO ITERATION.  THE VALUES TO BE READ IN INCLUDE: *
C     *     (A) ORPRMS -- THE INITIAL PARAMETER SETTINGS FOR SIMULA*
C     *     (B) KONECT -- SHOWS NODE TO NODE CONNECTIONS           *
C     *     (C) ORSDTB -- THE INITIAL SEED TABLES FOR EACH ITER.   *
C     *     (D) SORCHL -- THE BEGIN POINT OF EACH CHANNEL          *
C     *     (E) NODCHL -- THE END POINT OF EACH CHANNEL            *
C     *     (F) PARM3  -- THE NUMBER OF SLOTS IN EACH CHANNEL      *
C     *     (G) JARM3  -- THE TOTAL SLOTS IN ALL PRIOR CHANNELS    *
C     ************************************************************
C
C
C
      SUBROUTINE DEFMOD
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
C
C     ************************************************************
C     * READ IN THE RESTART FLAG.  IF IT IS ON (1) THEN THE DATA   *
C     * WHICH FOLLOWS REPRESENTS A SNAPSHOT OF THE NETWORK AT      *
C     * STEADY STATE.  ALSO READ IN THE NEW SETTINGS FOR CERTAIN   *
C     * CONTROL VARIABLES.                                         *
C     ************************************************************
C
      READ(5,1012) RESTRT,NUSTFD,NUPOWR,NUSWCH,NUGLOB,NUSTPP
```

```
      READ(5,1013) WCFCNU,WPFCNU
      IF (RESTRT.EQ.1) CALL SSREAD
      IF (RESTRT.EQ.1) RETURN
C
C     ***************************************************************
C     * READ THE VALUES FOR THE SEVEN ARRAYS LISTED ABOVE.  THESE   *
C     * VALUES ARE NOT CHANGED ON EACH OF SIMULA'S ITERATIONS.      *
C     ***************************************************************
C
      READ(5,1010) (ORPRMS(I),I=1,10)
      READ(5,1011) (ORPRMS(I),I=11,16)
      ORPRMS(17)=0
      NODES=ORPRMS(1)
      CHNLS=ORPRMS(2)
      DO 79 I=1,NODES
         READ(5,1012) (KONECT(I,J),J=1,NODES)
   79    CONTINUE
      DO 80 I=1,NODES
         READ(5,1015) (ORSDTB(I,J),J=1,4)
   80    CONTINUE
      DO 81 I=1,CHNLS
         READ(5,1020) SORCHL(I),NODCHL(I),PARM3(I),JARM3(I)
   81    CONTINUE
C
C     ***************************************************************
C     * ESTABLISH THE ASSUMED RELATIONSHIP BETWEEN PACKET NODES     *
C     * (1 THRU N) AND THE CIRCUIT NODES.                           *
C     ***************************************************************
C
      S1=NODES/2
      DO 82 I=1,S1
         PKLINK(I)=I+S1
         PKLINK(I+S1)=0
         CSLINK(I+S1)=I
         CSLINK(I)=0
   82    CONTINUE
C
C     ***************************************************************
C     * ECHO THE NETWORK CONFIGURATION DATA TO INSURE ACCURACY.     *
C     ***************************************************************
C
      WRITE(6,2000)
      WRITE(6,2011)
      WRITE(6,2012)
      WRITE(6,2015) (ORPRMS(I),I=1,16)
      WRITE(6,2020)
      DO 300 I=1,NODES
         WRITE(6,2100) (KONECT(I,J),J=1,NODES)
  300    CONTINUE
      WRITE(6,3001)
      WRITE(6,3002) ((ORSDTB(I,J),J=1,4),I=1,NODES)
      WRITE(6,3006)
```

```
      DO 400 I=1,CHNLS
         WRITE(6,3007) I,SORCHL(I),NODCHL(I),PARM3(I),JARM3(I)
  400    CONTINUE
      WRITE(6,3008)
      DO 500 I=1,NODES
         WRITE(6,3009) I,PKLINK(I),CSLINK(I)
  500    CONTINUE
C
C
C
 1010 FORMAT (8(I4,1X),2(I10,1X))
 1011 FORMAT (2(I5,1X),I7,1X,I3,1X,I4,1X,I2)
 1012 FORMAT (52I2)
 1013 FORMAT (2F6.3)
 1015 FORMAT (4(I5,1X))
 1020 FORMAT (4(I10,1X))
C
C
C
 2000 FORMAT ('1',40X,'SYSTEM PARAMETERS')
 2011 FORMAT ('0',1X,'NODES LINKS SLOTS RATIO SLOT NODE  CS',6X,
     1        'PS  MSG START TIME END TIME PACKET  VDR RATES ',
     2        'Q SIZE   CS      PACKET PACKETS')
 2012 FORMAT (' ',25X,'TIME DELAY ARRIVAL ARRIVAL',21X,
     1        'LOADING',19X,'SERVICE SIZE   PER MSG')
 2015 FORMAT (' ',1X,3(I5,1X),I2,2X,I4,'MS ',I2,' MS',3X,I2,
     1        'MIN',3X,I2,'SEC',2X,I5,' MS',I8,'MS ',I7,1X,
     2        I5,'KBS ',I7,3X,I3,'SEC ',I4,'B',2X,I2)
 2020 FORMAT (///,' NETWORK CONNECTION TABLE',///)
 2100 FORMAT (' ',20I3)
 3001 FORMAT ('1',///,5X,'SEED TABLES:')
 3002 FORMAT (4(1X,I15))
 3006 FORMAT ('1',/,5X,'SOURCE, DESTINATION, AND CAPACITY (IN SLOTS)',
     1        'FOR EACH CHANNEL:'//5X,'CHAN  SORCHL(I)  NODCHL(I)  ',
     2        'PARM3(I)  JARM3(I)')
 3007 FORMAT (5X,I4,4(I7,4X))
 3008 FORMAT (///,'   I   PKLINK(I)   CSLINK(I)')
 3009 FORMAT (' ',' ',I2,I6,I11)
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * SUBROUTINE DEFRTE WILL READ IN THE VALUES NEEDED TO DEFINE *
C     * THE ROUTING STRATEGY TO BE USED. THESE VALUES WILL NOT     *
C     * OVER SIMULA'S ITERATIONS AND INCLUDE:                      *
C     *      (A) STOFWD                                            *
C     *      (B) PRINCR                                            *
C     *      (C) WKINCR                                            *
C     *      (D) WPINCR                                            *
C     *      (E) POWER                                             *
C     *      (F) SWITCH                                            *
C     *      (G) GLOBAL                                            *
C     *      (H) STPPRI                                            *
C     *      (I) DESTAB                                            *
C     *      (J) DSTALT                                            *
C     *      (K) NMTRYS                                            *
C     *      (L) RANDSD                                            *
C     ************************************************************
C
C
C
C
      SUBROUTINE DEFRTE
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
```

```
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      IF (RESTRT.EQ.1) RETURN
C
C     ************************************************************
C     * READ IN THE ROUTING DEFINITION DATA                      *
C     ************************************************************
C
      NODES=ORPRMS(1)
      READ(5,1000) STOFWD
      READ(5,1001) PRINCR
      READ(5,1002) WCINCR,WPINCR
      READ(5,1000) POWER,SWITCH,GLOBAL,STPPRI
      READ(5,1004) ((DESTAB(I,J),J=1,NODES),I=1,NODES)
      READ(5,1004) ((DSTALT(I,J),J=1,NODES),I=1,NODES)
      READ(5,1005) NMTRYS,(RANDSD(I),I=1,NODES)
C
C
C
      WCFACT=1.0-WCINCR
      WPFACT=1.0-WPINCR
C
C     ************************************************************
C     * SOME STRATEGIES PERFORM ANALYSIS ON THE FEASIBLE ROUTES  *
C     * BETWEEN TWO NODES.  SUBROUTINE PATHS WILL GENERATE THOSE *
C     * ROUTES FOR LATER ANALYSIS.                               *
C     ************************************************************
C
      CALL PATHS
C
C     ************************************************************
C     * IF SWITCH = 6 THEN CREATE THE PRIMARY ONLY STRATEGY BY   *
C     * MAKING THE TWO ROUTING TABLES (DESTAB AND DSTALT) THE SAME.*
C     ************************************************************
C
      IF (SWITCH.NE.6) GOTO 200
      SWITCH=2
      DO 160 I=1,NODES
         DO 150 J=1,NODES
            DSTALT(I,J)=DESTAB(I,J)
  150       CONTINUE
  160    CONTINUE
  200 CONTINUE
C
C
C
```

```
      IF (STOFWD.EQ.0) WRITE(6,2050)
      IF (STOFWD.EQ.1) WRITE(6,2051)
C
C
C
      WRITE(6,3001) POWER,SWITCH,GLOBAL,STPPRI
      WRITE(6,3002) NMTRYS,(RANDSD(I),I=1,NODES)
      WRITE(6,3003)
      WRITE(6,3004) ((DESTAB(I,J),J=1,NODES),I=1,NODES)
      WRITE(6,3005)
      WRITE(6,3004) ((DSTALT(I,J),J=1,NODES),I=1,NODES)
C
C
C
 1000 FORMAT (I2)
 1001 FORMAT (I10)
 1002 FORMAT (2F6.3)
 1004 FORMAT (13(I2,1X))
 1005 FORMAT (I6)
C
C
C
 2050 FORMAT (' ',///,5X,'STORE AND FORWARD MODE FOR PACKETS IS OFF')
 2051 FORMAT (1H0,///,5X,'STORE AND FORWARD MODE FOR PACKETS IS ON')
 3001 FORMAT (///,5X,'POWER IS EQUAL TO ',I3,
     *        ///,5X,'SWITCH IS EQUAL TO ',I3,
     *        ///,5X,'GLOBAL IS EQUAL TO ',I3,
     *        ///,5X,'STPPRI IS EQUAL TO ',I3)
 3002 FORMAT ('1',///,5X,'RANDOM TRIES =',I3,
     *        ///,5X,'RANDOM SEEDS ARE',52(/,10X,I10))
 3003 FORMAT ('1',///,5X,'PRIMARY ROUTING TABLE:')
 3004 FORMAT (' ',20I3)
 3005 FORMAT (///,5X,'ALTERNATE ROUTING TABLE:')
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * THIS ROUTINE WILL GENERATE ALL POSSIBLE PATHS BETWEEN ANY  *
C     * TWO NODES.  THE PATHS WILL BE PLACED IN THE TABLE P AND    *
C     * THE ARRAYS FSTPTH AND LSTPTH WILL POINT TO THE FIRST AND   *
C     * LAST ELEMENTS OF A LIST OF PATHS BETWEEN ANY TWO NODES.    *
C     ************************************************************
C
C
C
      SUBROUTINE PATHS
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      DO 50 I=1,800
         DO 40 J=1,10
            P(I,J)=0
   40       CONTINUE
         F(I)=0
         L(I)=0
         S(I)=0
         SELCNT(I)=0
   50    CONTINUE
```

```
C
C
C
      ND=ORPRMS(1)
      CT=0
C
C
C
  200 DO 300 I=11, ND
         DO 250 J=11, ND
            IF (KONECT(I,J).LE.0) GOTO 250
            CT = CT+1
            P(CT,1) = I
            P(CT,2) = J
            F(CT)   = I
            L(CT)   = J
            S(CT)   = 2
  250       CONTINUE
  300    CONTINUE
C
C
C
      K=1
  400 CALL EXPAND(K,CT)
      K=K+1
      IF (K.LE.CT) GOTO 400
C
C
C
      DO 500 I=1,CT
         DO 450 J=I,CT
            IF (F(I).LE.F(J)) GOTO 450
            CALL SWAP(I,J)
  450       CONTINUE
  500    CONTINUE
C
C
C
      OLDTOP = 1
      DO 610 K=1,CT
         IF (K.EQ.CT) GOTO 540
         IF (F(K).EQ.F(K+1)) GOTO 610
  540    DO 600 I=OLDTOP, K
            DO 550 J=I,K
               IF (L(I).LE.L(J)) GOTO 550
               CALL SWAP(I,J)
  550          CONTINUE
  600       CONTINUE
         OLDTOP = K+1
  610    CONTINUE
C
C
```

```
C
      OLDTOP = 1
      DO 810 K=1,CT
         IF (K.EQ.CT) GOTO 740
         IF (L(K).EQ.L(K+1)) GOTO 810
  740    DO 800 I=OLDTOP, K
            DO 750 J=I,K
               IF (S(I).LE.S(J)) GOTO 750
               CALL SWAP(I,J)
  750          CONTINUE
  800       CONTINUE
         OLDTOP = K+1
  810    CONTINUE
C
C
C
      DO 850 I=1,ND
         DO 840 J=1,ND
            FSTPTH(I,J)=0
            LSTPTH(I,J)=0
  840       CONTINUE
  850    CONTINUE
C
C
C
      FSTPTH(11,12)=1
      S1=CT-1
      DO 900 I=1,S1
         IF (L(I).EQ.L(I+1)) GOTO 900
         LSTPTH(F(I),L(I))=I
         FSTPTH(F(I+1),L(I+1))=I+1
  900    CONTINUE
      LSTPTH(F(CT),L(CT))=CT
C
C
C
      S1=ND/2
      DO 960 I=1,S1
         DO 950 J=1,S1
            T1=FSTPTH(I+S1,J+S1)
            T2=LSTPTH(I+S1,J+S1)
            FSTPTH(I,J)=T1
            LSTPTH(I,J)=T2
            FSTPTH(I,J+S1)=T1
            LSTPTH(I,J+S1)=T2
            FSTPTH(I+S1,J)=T1
            LSTPTH(I+S1,J)=T2
  950       CONTINUE
  960    CONTINUE
C
C
C
```

```
      PTHCNT=CT
      DO 980 K=1,CT
         LIM=S(K)
         DO 970 J=2,LIM
            P(K,J-1)=KONECT(P(K,J-1),P(K,J))
  970       CONTINUE
         P(K,LIM)=0
         S(K)=S(K)-1
  980    CONTINUE
C
C
C
      RETURN
      END
```

```
C     ***************************************************************
C     * THIS ROUTINE WILL GENERATE THE LEGAL PATHS FROM SOME ROOT   *
C     * (P(I,*)).  PATHS GENERATED WILL BE STORED IN P WITH CT      *
C     * POINTING TO THE LAST VALID ONE.                             *
C     ***************************************************************
C
C
C
      SUBROUTINE EXPAND(K,CT)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
      LST=0
      END=10
      DO 100 I=1,9
         IF (P(K,11-I).NE.0) GOTO 110
         LST=P(K,10-I)
         END=10-I
  100    CONTINUE
  110 IF (END.EQ.1) STOP
      IF (END.EQ.10) RETURN
C
C
      DO 200 I=1,10
         CND = I+10
         IF (KONECT(LST,CND).LE.0) GOTO 200
         DO 125 J=1,END
            IF (CND.EQ.P(K,J)) GOTO 200
  125       CONTINUE
         CT = CT+1
         F(CT) = F(K)
         L(CT) = CND
         S(CT) = END+1
         DO 150 J=1,END
            P(CT,J) = P(K,J)
  150       CONTINUE
         P(CT,END+1) = CND
  200    CONTINUE
C
      RETURN
      END
```

```
C     ***************************************************************
C     * THIS ROUTINE WILL EXCHANGE THE CONTENTS OF THE ITH AND JTH *
C     * ELEMENTS OF THE THREE ARRAYS WHICH DEFINES/DESCRIBE EACH   *
C     * POTENTIAL PATH.                                            *
C     ***************************************************************
C
C
C
      SUBROUTINE SWAP (I,J)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
            TMP = F(I)
            F(I) = F(J)
            F(J) = TMP
C
            TMP  = L(I)
            L(I) = L(J)
            L(J) = TMP
C
            TMP  = S(I)
            S(I) = S(J)
            S(J) = TMP
C
            DO 425 K=1,10
               TMP = P(I,K)
               P(I,K) = P(J,K)
               P(J,K) = TMP
  425          CONTINUE
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     *                                                          *
C     *     SUBROUTINE SIMINT - OBTAINS INITIAL VALUES FOR SIMULA  *
C     *                                                          *
C     ************************************************************
C
C
C
C
      SUBROUTINE SIMINT
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
      IF (RESTRT.EQ.1) RETURN
C
C     ************************************************************
C     * INITIALIZE/MODIFY THE PARAMETERS WHICH DRIVE THE SIMULATION*
C     ************************************************************
C
      DO 10 I=1,17
         PARAM(I)=ORPRMS(I)
   10    CONTINUE
C
C
```

```
      NODES=PARAM(1)
      DO 30 I=1,NODES
         DO 20 J=1,4
            SEEDTB(I,J)=ORSDTB(I,J)
   20       CONTINUE
   30    CONTINUE
C
C
      NCHNLS=PARAM(2)
      DO 80 I=1,NCHNLS
         NLINES(I)=PARM3(I)
   80    CONTINUE
C
C
      WCFACT=WCFACT+WCINCR
      WPFACT=WPFACT+WPINCR
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     *                                                          *
C     * SUBROUTINE SIMULA DRIVES THE NETWORK SIMULATION ROUTINES.  *
C     * THIS IS THE DRIVER-IT BUILDS USER DEFINED TABLES,          *
C     * INITIALIZES ACTIVITY AT EACH NODE, AND EMPLOYS A TIGHT     *
C     * DO LOOP, CALLING THE EVENT MODULE UNTIL THE RUN TIME       *
C     * SPECIFICATION IS EXCEEDED.  AT THIS POINT THE STATISTICS   *
C     * SUBROUTINE IS CALLED, FOLLOWED BY PROGRAM TERMINATION.     *
C     *                                                          *
C     ************************************************************
C
C
C
      SUBROUTINE SIMULA(IPASS,STABLE,OVRLOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
      IF (RESTRT.EQ.1) GOTO 20
      WRITE(6,3999) IPASS,WCFACT,WPFACT
      ALLDST=PARAM(1)
      NDEST=ALLDST/2
      PARAM(17)=0
```

```
      EVTX(5)=0
      CALL BDATA
      CLASS=2
C
C     ***************************************************************
C     * CREATE AN INITIAL EVENT TABLE ENTRY FOR EACH NODE.          *
C     ***************************************************************
C
      DO 10 I=1,ALLDST
         IF (I.GT.NDEST) CLASS=1
         CALL NEWMSG(I,CLASS)
         CALL NUEVNT(I,CLASS)
   10    CONTINUE
C
C     ***************************************************************
C     *  START THE SIMULATION                                       *
C     ***************************************************************
C
C
      BOUND=500
C
   20 CALL EVENT
      IF (EVTX(5).LE.BOUND) GOTO 20
C
      CALL SSTDMP
      BOUND=BOUND+500
      SSMBND=(BOUND-500)/10000
      SSMBND=SSMBND*10000
      IF (SSMBND.NE.(BOUND-500)) GOTO 20
C
      SOURCE=1
      CALL SSMARK(SOURCE)
      IF (BOUND.LT.50500) GOTO 20
C
C     ***************************************************************
C     * SEE IF THE NETWORK HAS STABILIZED.                          *
C     ***************************************************************
C
      STABLE=1
      CNFINT=.01
      CALL SSANAL (SSFLAG,CNFINT)
      IF (SSFLAG.EQ.1) GOTO 40
      CNFINT=.025
      CALL SSANAL(SSFLAG,CNFINT)
      IF (SSFLAG.EQ.1) GOTO 40
      CNFINT=.05
      CALL SSANAL(SSFLAG,CNFINT)
      IF (SSFLAG.EQ.1) GOTO 40
      CNFINT=.1
      CALL SSANAL(SSFLAG,CNFINT)
      IF (SSFLAG.EQ.1) GOTO 40
      STABLE=0
```

```
C
C     ****************************************************************
C     * PRINT STATISTICS                                             *
C     ****************************************************************
C
   40 PARAM(10)=PARAM(9)
      PARAM(9)=PARAM(17)
      CALL STATX
      CALL STATS
C
C     ****************************************************************
C     * SEE IF THE NETWORK HAS BECOME OVERLOADED.                    *
C     ****************************************************************
C
      IF (XS(SSC,2).GT.1.) OVRLOD=1.
      IF (XS(SSC,3).GT..1) OVRLOD=1
C
C     ****************************************************************
C     * STOP THE SIMULATION.                                         *
C     ****************************************************************
C
  999 RETURN
C
C
C
 3999 FORMAT (///,'SIMULA PASS=',I5,'  WORK FACTORS=',2F6.3,///)
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE WILL ANALYZE THE DATA IN THE COMMON ARRAY SS  *
C     * TO SEE IF THE SIMULATION IS IN A STEADY STATE CONDITION.   *
C     ************************************************************
C
C
C
      SUBROUTINE SSANAL(SSFLAG,CNFINT)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
      IF (SSC.LE.1) RETURN
C
C
C
      DO 100 I=1,2
         SSFLAG=1
         XLAST=XS(SSC,I)
         XDELT=XLAST*CNFINT
         LIM=SSC-1
         M=SSC
         DO 10 J=1,LIM
            K=SSC-J
            X=XS(K,I)-XLAST
            IF (X.LT.0) X=(-1)*X
            IF (X.GT.XDELT) GOTO 20
            M=K
   10       CONTINUE
   20    M=(M-1)*1.25
         IF (M.GT.SSC) SSFLAG=0
         IF ((SSFLAG.EQ.0).AND.(I.EQ.2)) WRITE(6,2001) CNFINT
         IF ((SSFLAG.EQ.0).AND.(I.EQ.1)) WRITE(6,2002) CNFINT
         IF ((SSFLAG.EQ.1).AND.(I.EQ.2)) WRITE(6,2003) CNFINT
         IF ((SSFLAG.EQ.1).AND.(I.EQ.1)) WRITE(6,2004) CNFINT
  100    CONTINUE
C
C
C
 2001 FORMAT (///,6X,'PACKET DELAY IS UNSTABLE AT CNFINT =',F6.3)
```

```
 2002 FORMAT (///,6X,'UTILIZATION IS UNSTABLE AT CNFINT =',F6.3)
 2003 FORMAT (///,6X,'PACKET DELAY IS STABLE AT CNFINT =',F6.3)
 2004 FORMAT (///,6X,'UTILIZATION IS STABLE AT CNFINT =',F6.3)
C
C
C
      RETURN
      END
```

```
C     ****************************************************************
C     * THIS ROUTINE WILL PRINT THE DETAILED STEADY STATE DATA.      *
C     ****************************************************************
C
C
C
      SUBROUTINE SSTDMP
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
      IF (SNAPON.EQ.1) LSTSTP=1
      IF ((SNAPON.EQ.0).AND.(LSTSTP.EQ.1)) WRITE(6,2025) LSTSTP,SSC
      IF ((SNAPON.EQ.0).AND.(LSTSTP.EQ.1)) WRITE(6,2030)
      IF (SNAPON.EQ.1) WRITE(6,2025) LSTSTP,SSC
      IF (SNAPON.EQ.1) WRITE(6,2030)
      DO 200 I=LSTSTP,SSC
         WRITE(6,2040) I,(XS(I,J),J=1,8)
  200    CONTINUE
      LSTSTP=SSC+1
C
C
C
 2025 FORMAT ('1',10X,'STEADY STATE VALUES FROM',I5,' TO',I5)
 2030 FORMAT (///,6X,'STEADY STATE TABLE')
 2040 FORMAT (' ',5X,I10,5F10.3,2F20.1,F10.3)
C
C
C
      RETURN
      END
```

```
C     ***************************************************************
C     * SUBROUTINE BDATA WILL INITIALIZE NECESSARY ARRAYS TO ZERO   *
C     * SO THAT THEY DO NOT HAVE TO BE READ IN.                     *
C     ***************************************************************
C
C
C
C
      SUBROUTINE BDATA
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      DO 10 I=1,52
         QCNT(I)=0
         BTLNCK(I)=0
         IF (I.GT.20) GOTO 6
    6    DO 8 J=1,5
            EVTBL(I,J)=0
    8       CONTINUE
         DO 9 J=1,52
```

```
            LINKTB(I,J)=0
    9       CONTINUE
   10    CONTINUE
C
C
C
      SNAPON=0
      LSTBTL=0
      LSTSTP=1
      RTFREE=1
      RTSTRT=1
      RSFLAG=0
      SSC=0
      DO 14 J=1,501
         DO 13 I=1,8
            XS(J,I)=0.0
   13       CONTINUE
   14    CONTINUE
C
C
C
      LSTEVT=0
      PAKAVG=0.0
      UAVG=0.0
      PAKTHR=0.0
      ZDAVG=0.0
      ZBLOCK=0.0
      DO 20 I=1,26
         APCKTS(I)=0
         TDEL(I)=0.0
         ZDBLK(I,1)=0.0
         ZDBLK(I,2)=0.0
         ACKPAC(I)=0
         DO 15 J=1,26
            DSTLOD(I,J)=0
            DSTCNT(I,J)=0
            CUMLOD(I,J)=0
            CUMCNT(I,J)=0
   15       CONTINUE
         DO 16 J=1,3
            CALLS(I,J)=0
            CSARV(I,J)=0
            NODLOD(I,J)=0
   16       CONTINUE
         DO 17 J=1,13
            CUMTM(I,J)=0
   17       CONTINUE
         DO 18 J=1,50
            CALQ1(I,J)=0
            CALQ1(I,J)=0
   18       CONTINUE
         IF (I.GT.10) GOTO 21
```

```
      DO 19 J=1,1800
         QUEUE1(I,J)=0
19       CONTINUE
20    CONTINUE
21 DO 30 I=1,1170
      DO 30 J=1,11
         CHANTB(I,J)=0
30    CONTINUE
   DO 40 I=1,6
      EVTX(I)=0
40    CONTINUE
   DO 50 I=1,160
      ROUT(I)=0
      ALTCH(I)=0
      THRUTL(I,1)=0.0
      THRUTL(I,2)=0.0
50    CONTINUE
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * SUBROUTINE SSREAD WILL INITIALIZE THE SYSTEM VAIRABLES SO  *
C     * THAT THEY REPRESENT A STEADY STATE CONDITION.              *
C     ************************************************************
C
C
C
      SUBROUTINE SSREAD
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      CALL BDATA
C
```

```
C
C
      READ(5,1500) (PARAM(I),I=1,6 )
      READ(5,1500) (PARAM(I),I=7,12)
      READ(5,1500) (PARAM(I),I=13,17)
      READ(5,1500) (ORPRMS(I),I=1,6)
      READ(5,1500) (ORPRMS(I),I=7,12)
      READ(5,1500) (ORPRMS(I),I=13,17)
      NODES=PARAM(1)
      CHANS=PARAM(2)
      SLOTS=PARAM(3)
C
C
C
      DO 200 I=1,SLOTS
         READ(5,1500) (CHANTB(I,J),J=1,6)
         READ(5,1500) (CHANTB(I,J),J=7,11)
  200    CONTINUE
C
      DO 300 I=1,CHANS
         READ(5,1000) SORCHL(I),NODCHL(I),BTLNCK(I),ACKPAC(I)
         READ(5,1000) PARM3(I),JARM3(I),NLINES(I),ALTCH(I)
  300    CONTINUE
C
      DO 400 I=1,NODES
         READ(5,1500) (EVTBL(I,J),J=1,5)
         READ(5,1500) PKLINK(I),CSLINK(I)
         READ(5,1500) (SEEDTB(I,J),J=1,4),RANDSD(I)
         READ(5,2500) (LINKTB(I,J),J=1,NODES)
         READ(5,2500) (DESTAB(I,J),J=1,NODES)
         READ(5,2500) (DSTALT(I,J),J=1,NODES)
         READ(5,2500) (KONECT(I,J),J=1,NODES)
         READ(5,1500) QCNT(I),(ORSDTB(I,J),J=1,4)
  400    CONTINUE
C
      PACSIT=NODES/2
      DO 500 I=1,PACSIT
         READ(5,1000) (DSTCNT(I,J),J=1,PACSIT)
         READ(5,1000) (DSTLOD(I,J),J=1,PACSIT)
         READ(5,1000) (CUMLOD(I,J),J=1,PACSIT)
         READ(5,1000) (CUMCNT(I,J),J=1,PACSIT)
         READ(5,1500) (CALLS(I,J),J=1,3),(CSARV(I,K),K=1,3)
         READ(5,1005) (NODLOD(I,J),J=1,3),APCKTS(I),TDEL(I)
         READ(5,1000) (FSTPTH(I,J),J=1,PACSIT)
         READ(5,1000) (LSTPTH(I,J),J=1,PACSIT)
         READ(5,1500) (CUMTM(I,J),J=1,6)
         READ(5,1500) (CUMTM(I,J),J=7,13)
  500    CONTINUE
C
      ITOP=PARAM(13)
      DO 600 J=1,ITOP
         READ(5,1000) (QUEUE1(I,J),I=1,PACSIT)
```

```
         READ(5,1000) (QUEUE2(I,J),I=1,PACSIT)
  600    CONTINUE
C
       ITOP=400
       DO 700 J=1,ITOP
          READ(5,1000) (CALQ1(I,J),I=1,PACSIT)
          READ(5,1000) (CALQ1(I,J),I=1,PACSIT)
  700    CONTINUE
C
       READ(5,1000) PTHCNT
       DO 900 I=1,PTHCNT
          READ(5,2000) (P(I,J),J=1,10),S(I),F(I),L(I),SELCNT(I)
  900    CONTINUE
C
       DO 910 I=1,501
          READ(5,1100) (XS(I,J),J=1,8)
  910    CONTINUE
C
C
       READ(5,1000)(EVTX(I),I=1,6),SWITCH,STOFWD,GLOBAL,STPPRI
       READ(5,1010) PRINCR,WCFACT,WCINCR,WPINCR,WPFACT,LSTEVT,SSC
       READ(5,1000) LSTBTL,NMTRYS,POWER,BOUND,RTSTRT,RTFREE,LSTSTP
       READ(6,1000) RSFLAG
C
C
C
       SWITCH=NUSWCH
       POWER=NUPOWR
       STOFWD=NUSTFD
       GLOBAL=NUGLOB
       STPPRI=NUSTPP
       WCFACT=WCFCNU
       WPFACT=WPFCNU
C
C
C
       WRITE(6,500) SWITCH,STOFWD,GLOBAL,STPPRI,POWER
       WRITE(6,501) WCFACT,WPFACT
C
C      ************************************************************
C      * IF SWITCH = 6 THEN CREATE THE PRIMARY ONLY STRATEGY BY     *
C      * MAKING THE TWO ROUTING TABLES (DESTAB AND DSTALT) THE SAME.*
C      ************************************************************
C
       IF (SWITCH.NE.6) GOTO 990
       SWITCH=2
       DO 960 I=1,NODES
          DO 950 J=1,NODES
             DSTALT(I,J)=DESTAB(I,J)
  950        CONTINUE
  960    CONTINUE
  990 CONTINUE
```

```
C
C
C
 1000 FORMAT(5I14,/,5I14)
 1005 FORMAT(4I10,F20.3)
 1010 FORMAT(I10,4F10.3,2I10)
 1100 FORMAT(4F15.3,/,4F15.3)
 1500 FORMAT(8I10)
 2000 FORMAT(13I3,I4)
 2500 FORMAT(20I3)
 4000 FORMAT('1',20X,'THE NEW CONTROL SETTINGS ARE:',/,
     *           25X,'SWITCH =',I4,/,
     *           25X,'STOFWD =',I4,/,
     *           25X,'GLOBAL =',I4,/,
     *           25X,'STPPRI =',I4,/,
     *           25X,'POWER  =',I4)
 4001 FORMAT(' ',25X,'WCFACT =',F6.3,/,
     *           25X,'WPFACT =',F6.3)
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * SUBROUTINE SSMARK WILL INITIALIZE THE SYSTEM VAIRABLES SO  *
C     * THAT THEY REPRESENT A STEADY STATE CONDITION.              *
C     ************************************************************
C
C
C
      SUBROUTINE SSMARK(SOURCE)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      IF (SOURCE.EQ.0) GOTO 90
      PASS=1
```

```
      GOTO 190
C
C     ****************************************************************
C     *  WE HAVE REACHED STEADY STATE.  INITIALIZE COUNTERS AFTER  *
C     *  DUMPING THE CURRENT STATE.                                  *
C     ****************************************************************
C
   90 WRITE(6,2010) PARAM(9),EVTX(5)
      PASS=0
      GOTO 190
C
C
C
  100 CONTINUE
C
      SAVE09=PARAM(9)
      SAVE10=PARAM(10)
      SAVE17=PARAM(17)
C
      PARAM(10)=PARAM(9)
      PARAM(9)=PARAM(17)
      CALL STATX
      CALL STATS
C
      PARAM(9)=SAVE09
      PARAM(10)=SAVE10
      PARAM(17)=SAVE17
C
      NDEST=PARAM(1)/2
      ALLDST=PARAM(1)
      ITOP=PARAM(3)
      DO 130 I=1,ITOP
         CHANTB(I,6)=0
         CHANTB(I,9)=0
         CHANTB(I,10)=0
         CHANTB(I,11)=0
  130    CONTINUE
      NCHAN=PARAM(2)
      DO 135 I=1,NCHAN
         ACKPAC(I)=0
  135    CONTINUE
      DO 140 I=1,NDEST
         DO 140 J=1,13
            CUMTM(I,J)=0
  140       CONTINUE
      DO 150 I=1,NDEST
         DO 150 J=1,2
            CALLS(I,J)=0
  150       CONTINUE
      DO 165 I=1,52
         BTLNCK(I)=0
  165    CONTINUE
```

```
      DO 168 I=1,800
         SELCNT(I)=0
  168    CONTINUE
      PARAM(17)=PARAM(9)
C
C
C
  190 IF (SNAPON.EQ.0) GOTO 995
      WRITE(6,3500)
      WRITE(6,1500) (PARAM(I),I=1,6 )
      WRITE(6,1500) (PARAM(I),I=7,12)
      WRITE(6,1500) (PARAM(I),I=13,17)
      WRITE(6,1500) (ORPRMS(I),I=1,6)
      WRITE(6,1500) (ORPRMS(I),I=7,12)
      WRITE(6,1500) (ORPRMS(I),I=13,17)
      NODES=PARAM(1)
      CHANS=PARAM(2)
      SLOTS=PARAM(3)
C
C
C
      DO 200 I=1,SLOTS
         WRITE(6,1500) (CHANTB(I,J),J=1,6)
         WRITE(6,1500) (CHANTB(I,J),J=7,11)
  200    CONTINUE
C
      DO 300 I=1,CHANS
         WRITE(6,1000) SORCHL(I),NODCHL(I),BTLNCK(I),ACKPAC(I)
         WRITE(6,1000) PARM3(I),JARM3(I),NLINES(I),ALTCH(I)
  300    CONTINUE
C
      DO 400 I=1,NODES
         WRITE(6,1500) (EVTBL(I,J),J=1,5)
         WRITE(6,1500) PKLINK(I),CSLINK(I)
         WRITE(6,1500) (SEEDTB(I,J),J=1,4),RANDSD(I)
         WRITE(6,2500) (LINKTB(I,J),J=1,NODES)
         WRITE(6,2500) (DESTAB(I,J),J=1,NODES)
         WRITE(6,2500) (DSTALT(I,J),J=1,NODES)
         WRITE(6,2500) (KONECT(I,J),J=1,NODES)
         WRITE(6,1500) QCNT(I),(ORSDTB(I,J),J=1,4)
  400    CONTINUE
C
      PACSIT=NODES/2
      DO 500 I=1,PACSIT
         WRITE(6,1000) (DSTCNT(I,J),J=1,PACSIT)
         WRITE(6,1000) (DSTLOD(I,J),J=1,PACSIT)
         WRITE(6,1000) (CUMLOD(I,J),J=1,PACSIT)
         WRITE(6,1000) (CUMCNT(I,J),J=1,PACSIT)
         WRITE(6,1500) (CALLS(I,J),J=1,3),(CSARV(I,K),K=1,3)
         WRITE(6,1005) (NODLOD(I,J),J=1,3),APCKTS(I),TDEL(I)
         WRITE(6,1000) (FSTPTH(I,J),J=1,PACSIT)
         WRITE(6,1000) (LSTPTH(I,J),J=1,PACSIT)
```

```
         WRITE(6,1500) (CUMTM(I,J),J=1,6)
         WRITE(6,1500) (CUMTM(I,J),J=7,13)
  500    CONTINUE
C
      ITOP=PARAM(13)
      DO 600 J=1,ITOP
         WRITE(6,1000) (QUEUE1(I,J),I=1,PACSIT)
         WRITE(6,1000) (QUEUE2(I,J),I=1,PACSIT)
  600    CONTINUE
C
      ITOP=400
      DO 700 J=1,ITOP
         WRITE(6,1000) (CALQ1(I,J),I=1,PACSIT)
         WRITE(6,1000) (CALQ1(I,J),I=1,PACSIT)
  700    CONTINUE
C
      WRITE(6,1000) PTHCNT
      DO 900 I=1,PTHCNT
         WRITE(6,2000) (P(I,J),J=1,10),S(I),F(I),L(I),SELCNT(I)
  900    CONTINUE
C
      DO 950 I=1,501
         WRITE(6,1100) (XS(I,J),J=1,8)
  950    CONTINUE
C
C
      WRITE(6,1000)(EVTX(I),I=1,6),SWITCH,STOFWD,GLOBAL,STPPRI
      WRITE(6,1010) PRINCR,WCFACT,WCINCR,WPINCR,WPFACT,LSTEVT,SSC
      WRITE(6,1000) LSTBTL,NMTRYS,POWER,BOUND,RTSTRT,RTFREE,LSTSTP
      WRITE(6,1000) RSFLAG
      WRITE(6,3600)
C
C
  995 PASS=PASS+1
      IF (PASS.EQ.1) GOTO 100
C
C
 1000 FORMAT(' ',5I14,/,' ',5I14)
 1005 FORMAT(' ',4I10,F20.3)
 1010 FORMAT(' ',I10,4F10.3,2I10)
 1100 FORMAT(' ',4F15.3,/,' ',4F15.3)
 1500 FORMAT(' ',8I10)
 2000 FORMAT(' ',13I3,I4)
 2010 FORMAT (///,'  TRANSIENTS COMPLETED AT TIME = ',I10,
     *            ' WITH EVTX(5) =',I10)
 2500 FORMAT(' ',20I3)
 3500 FORMAT(' SSMARK - START')
 3600 FORMAT(' SSMARK - END')
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * SUBROUTINE NEWMSG GENERATES VOICE AND PACKET ARRIVALS.   *
C     * INFORMATION RELATING TO EACH ARRIVAL RESULTS IN          *
C     * A QUEUE ENTRY BEING BUILT.  IF THE CURRENT LOAD EXCEEDS  *
C     * A USER SPECIFIED STEADY-STATE LOAD, CHANNEL TABLE        *
C     * STATISTICAL GATHERING ENTRIES ARE ZEROED OUT.            *
C     ************************************************************
C
C
C
      SUBROUTINE NEWMSG(NODE,CLASS)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      REAL*4 ALOG
      REAL*4 RN
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
```

```
C
      DATA FLAG/1/
C
C
C
C
C     ****************************************************************
C     * KNODE IS USED WITH CIRCUIT SWITCH NODES TO ALLOW             *
C     * PROPER SUBSCRIPTING TO OCCUR OF THE CIRCUIT SWITCH TABLES.   *
C     ****************************************************************
C
      IF ((CLASS.EQ.1).AND.(PARAM(7).EQ.0)) GOTO 170
      IF ((CLASS.EQ.2).AND.(PARAM(8).EQ.0)) GOTO 170
      IF (RESTRT.EQ.1) FLAG=0
      IF (CLASS.EQ.1) KNODE=CSLINK(NODE)
      IY=0
      RN=0.0
      STIME=PARAM(9)
C
C     ****************************************************************
C     * RANDOMLY GENERATE A DESTINATION NODE UNTIL ONE DIFFERENT     *
C     * FROM THE SOURCE IS FOUND.                                    *
C     ****************************************************************
C
   10 IX=SEEDTB(NODE,3)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,3)=IY
      DEST=PARAM(1)*RN+1
      IF (DEST.EQ.NODE) GOTO 10
C
C     ****************************************************************
C     *  BRANCH TO THE PROPER HANDLER BASED ON TRANSACTION CLASS.    *
C     ****************************************************************
C
      IF (CLASS.EQ.2) GOTO 20
C
C     ****************************************************************
C     *  CIRCUIT HANDLER - CLASS = 1.                                *
C     ****************************************************************
C
      IF (DEST.LE.(PARAM(1)/2)) GOTO 10
      CSARV(KNODE,3)=DEST
      IX=SEEDTB(NODE,1)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,1)=IY
C
C     ****************************************************************
C     *  GENERATE ARRIVAL TIME                                       *
C     ****************************************************************
C
      ARV=(60000*(-1.0/(PARAM(7)*WCFACT)*ALOG(RN)))+PARAM(9)
      IX=SEEDTB(NODE,2)
```

```
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,2)=IY
      DEP=-1000.0*PARAM(14)*ALOG(RN)+ARV
      CSARV(KNODE,1)=ARV
      CSARV(KNODE,2)=DEP
  100 QCNT(NODE)=QCNT(NODE)+1
  170 RETURN
C
C     ****************************************************************
C     *  PACKET HANDLER - CLASS = 2.                                 *
C     ****************************************************************
C
   20 IF (DEST.GT.(PARAM(1)/2)) GOTO 10
      X=0.0
      XLAMDA=PARAM(8)*WPFACT
      CALL POISSN(XLAMDA,X,NODE)
      NMSGS=X
      IX=SEEDTB(NODE,1)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,1)=IY
      KEY=0
      LIMIT=QCNT(NODE)
      ITOP=PARAM(13)-5
      DO 40 I=1,ITOP,6
         IF (QUEUE1(NODE,I).EQ.0) GOTO 40
         IF (QUEUE1(NODE,I).EQ.100) GOTO 46
         IF (QUEUE1(NODE,I).EQ.999999999) GOTO 46
         IF (QUEUE1(NODE,I+1).LE.KEY) GOTO 40
         KEY=QUEUE1(NODE,I+1)
   46    LIMIT=LIMIT-1
         IF (LIMIT.EQ.0) GOTO 50
   40    CONTINUE
   50 IF (KEY.GT.STIME) STIME=KEY
C
C     ****************************************************************
C     *  GENERATE ARRIVAL TIME AND NUMBER OF PACKETS.                *
C     ****************************************************************
C
      ARV=(1000*(-1.0*ALOG(RN)))+STIME
      XPROB=PARAM(16)/100.0
      LENGTH=0
      NUM=0
      IF (NMSGS.EQ.0) NMSGS=1
      DO 25 K=1,NMSGS
         CALL GEOM(XPROB,NUM,NODE)
         LENGTH=LENGTH+NUM
   25    CONTINUE
      IF (LENGTH.EQ.0)LENGTH=1
C
C
C
C
```

```
C     ************************************************************
C     * HAVING GENERATED THE TRANSACTION DATA, COMPOSE THE QUEUE  *
C     * ENTRY(S) AS APPROPRIATE.  THE DATA STORED IN THE QUEUES   *
C     * WILL BE AS FOLLOWS.                                       *
C     *                                                           *
C     *    -- QUEUE1(1) ==> PRIORITY INDICATOR.                   *
C     *    -- QUEUE1(2) ==> TRANSACTION ARRIVAL TIME.             *
C     *    -- QUEUE1(3) ==> TRANSACTION DEPARTURE TIME.           *
C     *    -- QUEUE1(4) ==> SIZE (IN PACKETS) OF THE TRANSACTION  *
C     *    -- QUEUE1(5) ==> INTERMEDIATE DESTINATION.             *
C     *    -- QUEUE1(6) ==> NUMBER OF MESSAGES.                   *
C     *                                                           *
C     *    -- QUEUE2(1) ==> SOURCE NODE OF TRANSACTION.           *
C     *    -- QUEUE2(2) ==> FINAL DESTINATION.                    *
C     *    -- QUEUE2(3) ==> PERC. OF MESSAGE ALLOCATED TO TRANSACT.*
C     *    -- QUEUE2(4) ==> TOTAL DELAY FOR TRANSACTION.
C     *    -- QUEUE2(5) ==> ROUTE POINTER                         *
C     *    -- QUEUE2(6) ==> DEPARTURE TIME + ROUTING DELAY.       *
C     ************************************************************
C
      ITOP=PARAM(13)
      SIZE=LENGTH
      PASS=1
      IF (STOFWD.EQ.1) GOTO 26
      DEP=ARV+LENGTH*PARAM(5)
      NMSGS=NMSGS*10000
      QCNT(NODE)=QCNT(NODE)+1
      GOTO 27
C
C
C
   26 DEP=ARV+PARAM(5)
      NMSGS=1.0*NMSGS/LENGTH*10000.0+.5
      QCNT(NODE)=QCNT(NODE)+LENGTH
C
C     ************************************************************
C     * IF IN STORE AND FORWARD MODE GENERATE A QUEUE ENTRY FOR   *
C     * EACH PACKET SO THAT EACH CAN BE HANDLED SEPERATELY.       *
C     * IF NOT, GENERATE ONE ENTRY FOR THE ENTIRE MESSAGE.        *
C     ************************************************************
C
   27 DO 29 K=1,PASS
         DO 28 I=1,ITOP,6
            IF (QUEUE1(NODE,I).NE.0) GOTO 28
C
            QUEUE1(NODE,I)=1
            QUEUE1(NODE,I+1)=ARV
            QUEUE1(NODE,I+2)=DEP
            QUEUE1(NODE,I+3)=SIZE
            QUEUE1(NODE,I+4)=DEST
            QUEUE1(NODE,I+5)=NMSGS
C
```

```
            QUEUE2(NODE,I)=NODE
            QUEUE2(NODE,I+1)=DEST
            QUEUE2(NODE,I+2)=0
            QUEUE2(NODE,I+3)=0
            QUEUE2(NODE,I+4)=0
            QUEUE2(NODE,I+5)=DEP
C
            ARV=DEP
            DEP=DEP+PARAM(5)
            GOTO 29
   28       CONTINUE
         CALL ERRMSG(20)
   29    CONTINUE
      IF (STOFWD.EQ.0) RETURN
      IF (SIZE.EQ.1) RETURN
      SIZE=1
      PASS=LENGTH-1
      IF (PASS.LE.0) RETURN
      GOTO 27
C
C
C
 9999 STOP
      END
```

```
C     ***************************************************************
C     * SUBROUTINE POISSN WILL GENERATE A RANDOM POISSON VALUE      *
C     * WHERE XLAMDA IS THE MEAN AND SEEDTB(NODE,2) IS THE CURRENT *
C     * SEED VALUE.                                                 *
C     ***************************************************************
C
C
C
      SUBROUTINE POISSN(XLAMDA,X,NODE)
C
C
C
      IMPLICIT INTEGER (A-S)
      REAL*4 RN
      REAL*4 EXP
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
C
      X=0.0
      T=EXP(-XLAMDA)
      T1=1.0
      IY=0
    4 IX=SEEDTB(NODE,2)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,2)=IY
      T1=T1*RN
      IF (T1-T)9,7,7
    7 X=X+1.0
      GOTO 4
    9 RETURN
      END
```

```
C     ************************************************************
C     * THIS ROUTINE GENERATES THE NUMBER OF PACKETS              *
C     * FOR A DATA TRANSACTION.                                   *
C     ************************************************************
C
C
C
      SUBROUTINE GEOM(XPROB,NUM,NODE)
C
C
C
      IMPLICIT INTEGER (A-S)
      REAL*4 ALOG10
      REAL*4 RN
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
C
      NUM=0
      IY=0
   10 IX=SEEDTB(NODE,4)
      CALL RANDOM(IX,IY,RN)
      SEEDTB(NODE,4)=IY
      IF (RN.LE.XPROB) GOTO 20
      NUM=NUM+1
      GOTO 10
   20 RETURN
      END
```

```
C     ****************************************************************
C     * THIS IS THE RANDOM NUMBER GENERATOR.                         *
C     ****************************************************************
C
C
C
C
      SUBROUTINE RANDOM(IX,IY,RN)
C
C
C
      IMPLICIT INTEGER (A-Q)
C
C
C
      IY=IX*65539
      IF (IY)3,4,4
    3 IY=IY+2147483647+1
    4 RN=IY
      RN=RN*.4656613E-9
      IX=IY
      RETURN
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS RESPONSIBLE FOR SELECTING THE NEXT       *
C     * ACTIVITY AT A NODE.  ONCE AN EVENT IS SELECTED           *
C     * (ARRIVAL OR DEPARTURE), INFORMATION PERTAINING TO IT     *
C     * IS PLACED IN THE EVENT TABLE ENTRY FOR THAT NODE.        *
C     ************************************************************
C
C
C
      SUBROUTINE NUEVNT(NODE,CLASS)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
C
      IDELAY=0
      IYESNO=0
      LIMIT=QCNT(NODE)
C
C     ************************************************************
C     *  BRANCH TO THE PROPER HANDLER BASED ON TRANSACTION CLASS.  *
C     ************************************************************
C
      IF (CLASS.EQ.2) GOTO 110
C
C     ************************************************************
C     *  CIRCUIT HANDLER - CLASS = 1.                            *
C     ************************************************************
C
```

```
      KNODE=CSLINK(NODE)
C
C     ****************************************************************
C     * SET DEP TO THE MAXIMUM POSSIBLE VALUE AND THE SEARCH THE     *
C     * CALQ TABLES FOR THE SMALLEST ENTRY (SOONEST ACTIVITY).       *
C     ****************************************************************
C
      DEP=999999999
      DO 20 I=1,50,4
         IF (CALQ1(KNODE,I).EQ.0) GOTO 20
         IF (CALQ1(KNODE,I).GE.DEP) GOTO 15
         DEP=CALQ1(KNODE,I)
         INDEX=I
   15    LIMIT=LIMIT-1
         IF (LIMIT.EQ.0) GOTO 5
   20    CONTINUE
    5 IF (DEP.EQ.999999999) GOTO 25
      IF (CSARV(KNODE,1).GE.DEP) GOTO 30
C
C     ****************************************************************
C     *  NOW PLACE THE CIRCUIT SWITCH INFORMATION IN EVENT TABLE.    *
C     ****************************************************************
C
   25 EVTBL(NODE,1)=CSARV(KNODE,1)
      EVTBL(NODE,2)=3
      EVTBL(NODE,3)=CSARV(KNODE,2)
      EVTBL(NODE,4)=CSARV(KNODE,3)
      GOTO 100
C
C
C
   30 EVTBL(NODE,1)=DEP
      EVTBL(NODE,2)=4
      EVTBL(NODE,4)=CALQ1(KNODE,(INDEX+1))
      EVTBL(NODE,5)=INDEX
      GOTO 100
C
C     ****************************************************************
C     *  PACKET HANDLER - CLASS = 2.                                 *
C     ****************************************************************
C
  110 IF (PARAM(8).EQ.0) GOTO 100
      KEY=999999999
      ITOP=PARAM(13)-5
      DO 40 I=1,ITOP,6
C
C        ************************************************************
C        * IF QUEUE1(1) EQUAL ZERO, THEN SKIP IT.                   *
C        * IF QUEUE1(1) EQUAL 100, SKIP IT BECAUSE IT IS THE        *
C        * RETURN PATH FOR A CONNECTION.                            *
C        ************************************************************
C
```

```
      IF (QUEUE1(NODE,I).EQ.0) GOTO 40
      IF (QUEUE1(NODE,I).EQ.100) GOTO 40
      IF (QUEUE1(NODE,I+2).GE.KEY) GOTO 41
      IF (QUEUE1(NODE,I).NE.999999999) GOTO 41
      KEY=QUEUE1(NODE,I+2)
      FLAG=2
      INDEX=I
41    LIMIT=LIMIT-1
      IF (LIMIT.EQ.0) GOTO 45
40    CONTINUE
C
C     ***************************************************************
C     * THIS LOGIC CHECKS TRAFFIC THAT HAS BEEN ON QUEUE FOR        *
C     * SOME TIME TO SEE IF IT CAN BE SENT YET.                     *
C     ***************************************************************
C
45  PRIORY=0
    KK=6
    DO 55 KKK=1,5
      K=KK-KKK
      LIMIT=QCNT(NODE)
      IF (PRIORY.NE.0) GOTO 75
      ITOP=PARAM(13)-5
      DO 50 I=1,ITOP,6
        IF (QUEUE1(NODE,I).EQ.0) GOTO 50
        IF (QUEUE1(NODE,I).EQ.999999999) GOTO 46
        IF (QUEUE1(NODE,I).EQ.100) GOTO 46
        IF (QUEUE1(NODE,I+1).GE.KEY) GOTO 46
        IF (QUEUE1(NODE,I).NE.K) GOTO 46
        DEST=QUEUE1(NODE,I+4)
C
C       ***************************************************
C       * HAVING FOUND THE OLDEST TRAFFIC, CHECK THE PATH BY   *
C       * CALLING ROUTE.  IF EITHER A 0 OR 3 IS RETURNED VIA   *
C       * IYESNO THEN THE PATH IS LEGAL AND AVAILABLE.   NOTE  *
C       * THAT ROUTE IS USED TO CHECK FOR THESE CHARACTERISTICS *
C       * EVEN THOUGH THE AVAILABILITY COULD CHANGE BY THE TIME *
C       * THE EVENT IS SELECTED BY NUEVNT.                     *
C       ***************************************************
C
44      PASS=1
        AVLTST=1
        IF (STPPRI.EQ.1) GOTO 80
        IF (STOFWD.EQ.0) GOTO 82
        CALL SGLSTP(NODE,DEST,ACHAN,ANODE,AAVL,CLASS,ALSTND)
        DEST=ANODE
82      CALL ROUTE(NODE,DEST,IYESNO,IDELAY,CLASS,PASS)
        IF (IYESNO.EQ.1) GOTO 46
        IF (IYESNO.EQ.2) GOTO 46
80      KEY=QUEUE1(NODE,I+1)
        FLAG=1
        INDEX=I
```

```
            PRIORY=K
   46       LIMIT=LIMIT-1
            IF (LIMIT.LE.0) GOTO 55
   50       CONTINUE
   55    CONTINUE
      IF (KEY.NE.999999999) GOTO 75
      IF (KKK.LT.5) GOTO 75
      LIMIT=QCNT(NODE)
      FLAG=1
      ITOP=PARAM(13)-5
      KEY=QUEUE1(NODE,2)
C
C     ****************************************************************
C     *  SCAN THE QUEUE TO FIND THE SOONEST DEPARTURE.               *
C     ****************************************************************
C
      DO 35 I=1,ITOP,6
         IF (QUEUE1(NODE,I).EQ.0) GOTO 35
         IF (QUEUE1(NODE,I).LT.6) GOTO 36
   35    CONTINUE
   36 KEY=QUEUE1(NODE,I+1)
      INDEX=I
C
C     ****************************************************************
C     *  SCAN THE QUEUE TO FIND THE SOONEST ARRIVAL.                 *
C     ****************************************************************
C
      DO 60 I=1,ITOP,6
         IF (QUEUE1(NODE,I).EQ.0) GOTO 60
         IF (QUEUE1(NODE,I).EQ.100) GOTO 65
         IF (QUEUE1(NODE,I).EQ.999999999) GOTO 65
         IF (QUEUE1(NODE,I+1).GT.KEY) GOTO 85
   86    IF (QUEUE1(NODE,I).LT.5) GOTO 65
         QUEUE1(NODE,I)=1
   65    LIMIT=LIMIT-1
         IF (LIMIT.EQ.0) GOTO 75
         GOTO 60
   85    KEY=QUEUE1(NODE,I+1)
         INDEX=I
         IF (QUEUE1(NODE,I+1).LE.PARAM(9)) GOTO 86
         GOTO 75
   60    CONTINUE
C
C     ****************************************************************
C     * SELECT FROM ARRIVAL OR DEPARTURE, AND PLACE INFORMATION      *
C     * RELATING TO IT ON EVENT TABLE.                               *
C     ****************************************************************
C
   75 EVTBL(NODE,1)=QUEUE1(NODE,(INDEX+1))
      IF (FLAG.EQ.2)EVTBL(NODE,1)=QUEUE1(NODE,(INDEX+2))
      EVTBL(NODE,2)=FLAG+4
      IF (STOFWD.EQ.0) GOTO 76
```

```
      SIZE=QUEUE1(NODE,INDEX+3)
      IF (SIZE.EQ.1) EVTBL(NODE,2)=FLAG-4
      QUEUE1(NODE,INDEX+3)=1
   76 EVTBL(NODE,3)=QUEUE1(NODE,(INDEX+3))
      EVTBL(NODE,4)=QUEUE1(NODE,(INDEX+4))
      EVTBL(NODE,5)=INDEX
C
C
C
  100 RETURN
C
C
C
      END
```

```
C     ****************************************************************
C     * THIS ROUTINE SCANS THE EVENT TABLE ENTRIES LOOKING FOR       *
C     * THE NEXT SYSTEM EVENT.  IT THEN BRANCHES TO THE              *
C     * APPROPRIATE ROUTINE TO SERVICE THE EVENT.                    *
C     ****************************************************************
C
C
C
      SUBROUTINE EVENT
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
C
C
C
C     ****************************************************************
C     * IF THE NEXT MULTIPLE OF PRINCR HAS BEEN REACHED BY           *
C     * EVTX(5) COLLECT THE INTERMEDIATE STATISTICS.                 *
C     ****************************************************************
C
      N=EVTX(5)/PRINCR
      N=N*PRINCR
```

```
      IF (N.NE.EVTX(5)) GOTO 4
      IF (N.EQ.LSTEVT) GOTO 4
      CALL STATK
      LSTEVT=EVTX(5)
C
C
C
    4 IF (RSFLAG.EQ.1) GOTO 5
      IF (EVTX(5).LT.40000) GOTO 5
      CALL SSTDMP
      SOURCE=0
      CALL SSMARK(SOURCE)
      RSFLAG=1
C
C     ***************************************************************
C     *  SCAN THE EVENT TABLE TO FIND THE SOONEST ACTIVITY.         *
C     ***************************************************************
C
    5 NOROUT=0
      BEST=999999999
      ALLDST=PARAM(1)
      DO 10 I=1,ALLDST
         IF (EVTBL(I,1).EQ.0) GOTO 10
         IF (EVTBL(I,1).GE.BEST) GOTO 10
         BEST=EVTBL(I,1)
         NODE=I
   10    CONTINUE
C
C     ***************************************************************
C     *  NOW PROCESS THE EVENT OCCURRENCE TYPE BASED ON THE SECOND  *
C     * ELEMENT OF THE EVENT TABLE ENTRY. THIS ELEMENT RANGES FROM  *
C     * ONE TO FOUR WITH INTERPRETATIONS AS FOLLOWS.                *
C     *                  (1) ==> PACKET ARRIVAL                     *
C     *                  (2) ==> PACKET DEPARTURE                   *
C     *                  (3) ==> CIRCUIT ARRIVAL                    *
C     *                  (4) ==> CIRCUIT DEPARTURE                  *
C     ***************************************************************
C
      KEY=EVTBL(NODE,2)
      OLDKEY=KEY
      EVTX(KEY)=EVTX(KEY)+1
      IF (KEY.LE.4) GOTO 60
      KEY=KEY-4
      EVTX(KEY)=EVTX(KEY)+1
   60 CONTINUE
      GOTO (100,200,300,50),KEY
C
C
C
  100 CLASS=2
      CALL ARRIVE(NODE,CLASS)
  150 IF (OLDKEY.LE.4) GOTO 155
```

```
  153 CALL NEWMSG(NODE,CLASS)
  155 CALL NUEVNT(NODE,CLASS)
  160 RETURN
C
C
C
  200 CLASS=2
      CALL DEPART(NODE,CLASS)
      GOTO 155
C
C
C
  300 CLASS=1
      CALL ARRIVE(NODE,CLASS)
      GOTO 153
C
C
C
  400 CLASS=1
      CALL DEPART(NODE,CLASS)
      GOTO 155
C
C
C
      END
```

```
C     ****************************************************************
C     * THIS SUBROUTINE WILL SELECT THE PROPER ROUTINE FOR ROUTING   *
C     * A TRANSACTION.                                               *
C     ****************************************************************
C
C
C
      SUBROUTINE ROUTE(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-S)
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
C
      IF (SWITCH.EQ.1) CALL RTUTIL(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
      IF (SWITCH.EQ.2) CALL RTPRBK(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
      IF (SWITCH.EQ.3) CALL RTRAND(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
      IF (SWITCH.EQ.4) CALL RTQCNT(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
      IF (SWITCH.EQ.5) CALL RTLIMT(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      RETURN
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.  *
C     * IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  THE     *
C     * STRATEGY TO BE FOLLOWED IS RANDOM ROUTING THEREFORE THIS   *
C     * ROUTINE WIL RANDOMLY CHOOSE A ROUTE FROM THE ARRAY P.      *
C     * IYESNO SIGNIFIES WHETHER A GOOD ROUTE WAS FOUND.           *
C     ************************************************************
C
C
C
      SUBROUTINE RTUTIL(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      REAL SCORE(40),SCR,SGLSCR,RN
      DIMENSION NTRACE(160),TCHNLS(160),PSTRYS(20)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C
C     ************************************************************
C     * INITIALIZE THE TRACE VARIABLES.                          *
C     ************************************************************
```

```
C
      IROUT=PARAM(2)
   10 DO 84 I=1,IROUT
         TCHNLS(I)=0
   84    CONTINUE
      TCNT=1
      IF (PASS.EQ.2) GOTO 80
      RLGTH=0
      FLGTH=0
      DO 85 I=1,IROUT
         ROUT(I)=0
   85    CONTINUE
      DO 86 I=1,26
         FTRACE(I)=0
         RTRACE(I)=0
   86    CONTINUE
   80 NODE=LNODE
      DEST=KDEST
      IDELAY=0
      RATIO=1
      IF (CLASS.EQ.2) RATIO=PARAM(4)
C
C     ****************************************************************
C     * SCORE EACH FEASIBLE PATH                                     *
C     ****************************************************************
C
      T1=FSTPTH(LNODE,KDEST)
      T2=LSTPTH(LNODE,KDEST)
      DO 505 I=T1,T2
         SGLSCR=0
         DO 220 J=1,10
            CHAN=P(I,J)
            IF (CHAN.EQ.0) GOTO 300
            P3=PARM3(CHAN)
            NL=NLINES(CHAN)
            RQ=RATIO+ROUT(CHAN)
            SCR=1.0
            IF (NL.LT.RQ) SCR=600
            SCR=SCR+(1.0*(P3-NL+RQ)/P3)
            SGLSCR=SGLSCR+(SCR**POWER)
  220       CONTINUE
  300    SCORE(I-T1+1)=SGLSCR
  505    CONTINUE
C
C
C
      T3=T2-T1+1
      XT=SCORE(1)
      PTR=1
      DO 600 I=1,T3
         IF (SCORE(I).GE.XT) GOTO 600
         XT=SCORE(I)
```

```
          PTR=I
  600     CONTINUE
C
C
C
      PTR=PTR+T1-1
      TCNT=0
      DO 700 I=1,10
         IF(P(PTR,I).EQ.0) GOTO 800
         TCNT=TCNT+1
         TCHNLS(TCNT)=P(PTR,I)
  700    CONTINUE
      WRITE(6,2065)
C
C
C
  800 CONTINUE
C
C     ****************************************************************
C     * WALK THE PATH SELECTED TO SEE IF IT IS AVAILABLE.  IF SO,    *
C     * TRANSFER THE PATH TO TCHNLS.  IF NOT, BRANCH TO ALTERNATE    *
C     * PATH GENERATOR.                                              *
C     ****************************************************************
C
      LIMIT=TCNT
      IYESNO=0
      CHNCNT=LINKTB(PKLINK(LNODE),DEST)
      IF (CHNCNT.GT.0) IYESNO=3
      DO 900 I=1,LIMIT
         IDELAY=IDELAY+PARAM(6)
         ROUT(TCHNLS(I))=RATIO
         IF (RATIO.LE.NLINES(TCHNLS(I))) GOTO 900
         IF (AVLTST.EQ.0) BTLNCK(TCHNLS(I))=BTLNCK(TCHNLS(I))+1
         IYESNO=1
         GOTO 200
  900    CONTINUE
  200 IF (PASS.EQ.2) GOTO 500
C
C
C
      DO 400 I=1,TCNT
         FTRACE(I)=TCHNLS(I)
  400    CONTINUE
      FLGTH=TCNT
      RETURN
C
C
C
  500 DO 510 I=1,TCNT
         RTRACE(I)=TCHNLS(I)
  510    CONTINUE
      RLGTH=TCNT
```

```
      RETURN
C
C
C
 2065 FORMAT (' TRANSFER ERROR')
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.  *
C     * IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  NTRACE  *
C     * IS A TABLE USED TO INSURE NO LOOPS OCCUR IN THE ROUTING    *
C     * PROCESS.  IYESNO SIGNIFIES WHETHER OR NOT A GOOD           *
C     * ROUTE WAS FOUND.                                           *
C     ************************************************************
C
C
C
      SUBROUTINE RTPRBK(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
      DIMENSION NTRACE(160),TCHNLS(160)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      IROUT=PARAM(2)
      DO 84 I=1,IROUT
         TCHNLS(I)=0
   84    CONTINUE
      TCNT=1
      IF (PASS.EQ.2) GOTO 80
      RLGTH=0
      FLGTH=0
      DO 85 I=1,IROUT
         ALTCH(I)=0
```

```
          ROUT(I)=0
   85     CONTINUE
          DO 86 I=1,26
          FTRACE(I)=0
          RTRACE(I)=0
   86     CONTINUE
   80     DO 90 I=1,IROUT
          NTRACE(I)=0
90        CONTINUE
          NODE=LNODE
          DEST=KDEST
          INODE=NODE
          NCOUNT=1
          PRBTL=0
          ITYPE=0
          IDELAY=0
          RATIO=1
          ALT=0
          IF (CLASS.EQ.2) RATIO=PARAM(4)
          NTRACE(NCOUNT)=INODE
          IYESNO=0
C
C     ****************************************************************
C     *  BRANCH TO THE PROPER HANDLER BASED ON TRANSACTION CLASS.  *
C     ****************************************************************
C
          IF (CLASS.EQ.1) GOTO 10
          INODE=DESTAB(NODE,DEST)
          IDELAY=0
          NCOUNT=NCOUNT+1
          NTRACE(NCOUNT)=INODE
C
C     ****************************************************************
C     *  SEE IF A PATH ALREADY EXISTS FOR THIS TRANSACTION.        *
C     ****************************************************************
C
          CHNCNT=LINKTB(INODE,DEST)
          IF (CHNCNT.LE.0) GOTO 10
          ITYPE=1
   10     IF (INODE.EQ.DEST) GOTO 60
          ICHAN=DESTAB(INODE,DEST)
C
C     ****************************************************************
C     *  SEE IF THERE ARE SLOTS AVAILABLE ON THE SELECTED CHANNEL. *
C     *  IF NOT, BRANCH TO THE ALTERNATE CHANNEL SELECTOR.  IF SO, *
C     *  THEN RESERVE THEM BY DECREMENTING THE SLOT COUNT.         *
C     ****************************************************************
C
   25     IF (RATIO.GT.NLINES(ICHAN)) GOTO 20
          ROUT(ICHAN)=RATIO+ROUT(ICHAN)
          IF ((NLINES(ICHAN)-ROUT(ICHAN)).LT.0) GOTO 50
          ALT=0
```

```
      INODE=NODCHL(ICHAN)
      DO 30 K=1,NCOUNT
         IF (INODE.EQ.NTRACE(K)) GOTO 50
   30    CONTINUE
      NCOUNT=NCOUNT+1
      NTRACE(NCOUNT)=INODE
      TCHNLS (TCNT)=ICHAN
      TCNT=TCNT+1
C
C     ****************************************************************
C     *  ADD THE DELAY TIME FOR SIGNALLING THROUGH THIS NODE.        *
C     ****************************************************************
C
      IDELAY=PARAM(6)+IDELAY
      IF (DESTAB(INODE,DEST).EQ.0) GOTO 60
      GOTO 10
C
C     ****************************************************************
C     *  ALTERNATE CHANNEL SELECTOR.                                 *
C     ****************************************************************
C
   20 IF (ALT.EQ.1) GOTO 50
      PRBTL=ICHAN
      ALT=1
      IF (ALTCH(ICHAN).EQ.1) GOTO 50
      ALTCH(ICHAN)=1
      ICHAN=DSTALT(INODE,DEST)
      GOTO 25
C
C
C
   50 IYESNO=1
      IF (PRBTL.NE.0.AND.AVLTST.EQ.0) BTLNCK(PRBTL)=BTLNCK(PRBTL)+1
   60 TCNT=TCNT-1
      IF (ITYPE.EQ.IYESNO) GOTO 70
      IF (ITYPE.EQ.0) IYESNO=2
      IF (ITYPE.EQ.1) IYESNO=3
   70 IF (PASS.EQ.2) GOTO 500
      DO 400 I=1,TCNT
         FTRACE(I)=TCHNLS(I)
  400    CONTINUE
      FLGTH=TCNT
      RETURN
C
C
C
  500 DO 510 I=1,TCNT
         RTRACE(I)=TCHNLS(I)
  510    CONTINUE
      RLGTH=TCNT
C
      RETURN
```

```
END
```

```
C     ************************************************************
C     * THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.  *
C     * IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  THE     *
C     * STRATEGY TO BE FOLLOWED IS RANDOM ROUTING THEREFORE THIS   *
C     * ROUTINE WIL RANDOMLY CHOOSE A ROUTE FROM THE ARRAY P.      *
C     * IYESNO SIGNIFIES WHETHER A GOOD ROUTE WAS FOUND.           *
C     ************************************************************
C
C
C
      SUBROUTINE RTRAND(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      DIMENSION NTRACE(160),TCHNLS(160),PSTRYS(20)
C
C
C
      REAL RN
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C
```

```
C     ****************************************************************
C     * INITIALIZE THE TRACE VARIABLES.                              *
C     ****************************************************************
C
      IROUT=PARAM(2)
      TRYS=NMTRYS
      PASSES=0
   10 DO 84 I=1,IROUT
         TCHNLS(I)=0
   84    CONTINUE
      TCNT=1
      IF (PASS.EQ.2) GOTO 80
      RLGTH=0
      FLGTH=0
      DO 86 I=1, 26
         FTRACE(I)=0
         RTRACE(I)=0
   86    CONTINUE
   80 NODE=LNODE
      DEST=KDEST
      IDELAY=0
      RATIO=1
      IF (CLASS.EQ.2) RATIO=PARAM(4)
C
C     ****************************************************************
C     * RANDOMLY SELECT A PATH TO BE FOLLOWED.                       *
C     ****************************************************************
C
   30 IX=RANDSD(NODE)
      CALL RANDOM(IX,IY,RN)
      RANDSD(NODE)=IY
      NUMPTH=LSTPTH(NODE,DEST)-FSTPTH(NODE,DEST)+1
      NUMPTH=(RN*NUMPTH)+FSTPTH(NODE,DEST)
C
C
C
      JNODE=LNODE
      JDEST=KDEST
      IF (CLASS.EQ.2) JNODE=PKLINK(JNODE)
      IF (CLASS.EQ.2) JDEST=PKLINK(JDEST)
      IF (F(NUMPTH).NE.JNODE) GOTO 30
      IF (L(NUMPTH).NE.JDEST) GOTO 30
C
C
C
      IF (PASSES.EQ.0) GOTO 89
      DO 88 I=1, PASSES
         IF (NUMPTH.EQ.PSTRYS(I)) GOTO 30
   88 CONTINUE
   89 PASSES=PASSES+1
      PSTRYS(PASSES)=NUMPTH
C
```

```
C     ************************************************************
C     * WALK THE PATH SELECTED TO SEE IF IT IS AVAILABLE.  IF SO,  *
C     * TRANSFER THE PATH TO TCHNLS.  IF NOT, BRANCH TO ALTERNATE  *
C     * PATH GENERATOR.                                            *
C     ************************************************************
C
      ITOP=PARAM(1)/2
      TCNT=0
      DO 90 I=1, ITOP
         IF (P(NUMPTH,I).EQ.0) GOTO 95
         IDELAY=IDELAY+PARAM(6)
         IF (RATIO.GE.NLINES(P(NUMPTH,I))) GOTO 100
         TCHNLS(I)=P(NUMPTH,I)
         TCNT=TCNT+1
   90    CONTINUE
   95 IYESNO=0
      CHNCNT=LINKTB(PKLINK(LNODE),DEST)
      IF (CHNCNT.GT.0) IYESNO=3
      GOTO 200
C
C     ************************************************************
C     * PATH IS NOT AVAILABLE SO CHECK TO SEE IF AN ALTERNATE IS   *
C     * TO BE TRIED.                                               *
C     ************************************************************
C
  100 TRYS=TRYS-1
      IF (PASSES.EQ.1) PRBTL=P(NUMPTH,I)
      IF (TRYS.GT.0) GOTO 10
      IYESNO=1
      IF (PRBTL.EQ.0) RETURN
      IF (AVLTST.EQ.1) RETURN
      BTLNCK(PRBTL)=BTLNCK(PRBTL)+1
      RETURN
C
C     ************************************************************
C     *  SINCE IYESNO IS 0 THE PATH EXISTS.
C     ************************************************************
C
  200 IF (PASS.EQ.2) GOTO 500
      DO 400 I=1, TCNT
         FTRACE(I)=TCHNLS(I)
  400    CONTINUE
      FLGTH=TCNT
      RETURN
C
  500 DO 510 I= 1, TCNT
         RTRACE(I)=TCHNLS(I)
  510    CONTINUE
      RLGTH=TCNT
C
      RETURN
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.  *
C     * IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  THE     *
C     * STRATEGY TO BE FOLLOWED IS RANDOM ROUTING THEREFORE THIS   *
C     * ROUTINE WIL RANDOMLY CHOOSE A ROUTE FROM THE ARRAY P.      *
C     * IYESNO SIGNIFIES WHETHER A GOOD ROUTE WAS FOUND.           *
C     ************************************************************
C
C
C
      SUBROUTINE RTQCNT(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      REAL SCORE(40),SCR,SGLSCR,RN
      DIMENSION NTRACE(160),TCHNLS(160),PSTRYS(20)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C
C     ************************************************************
C     * INITIALIZE THE TRACE VARIABLES.                          *
C     ************************************************************
```

```
C
   10 DO 84 I=1,160
         TCHNLS(I)=0
   84    CONTINUE
      TCNT=1
      IF (PASS.EQ.2) GOTO 80
      RLGTH=0
      FLGTH=0
      DO 85 I=1,160
         ROUT(I)=0
   85    CONTINUE
      DO 86 I=1,26
         FTRACE(I)=0
         RTRACE(I)=0
   86    CONTINUE
   80 NODE=LNODE
      DEST=KDEST
      IDELAY=0
      RATIO=1
      IF (CLASS.EQ.2) RATIO=PARAM(4)
C
C     ****************************************************************
C     * SCORE EACH FEASIBLE PATH                                     *
C     ****************************************************************
C
      T1=FSTPTH(LNODE,KDEST)
      T2=LSTPTH(LNODE,KDEST)
      DO 505 I=T1,T2
         SGLSCR=0
         DO 220 J=1,10
            CHAN=P(I,J)
            IF (CHAN.EQ.0) GOTO 300
            SOURCE=SORCHL(CHAN)
            SCR=1.0*QCNT(SOURCE)/PARM3(CHAN)
            SGLSCR=SGLSCR+(SCR**POWER)
  220       CONTINUE
  300    SCORE(I-T1+1)=SGLSCR
  505    CONTINUE
C
C
C
      T3=T2-T1+1
      XT=SCORE(1)
      PTR=1
      DO 600 I=1,T3
         IF (SCORE(I).GE.XT) GOTO 600
         XT=SCORE(I)
         PTR=I
  600    CONTINUE
C
C
C
```

```
      PTR=PTR+T1-1
      TCNT=1
      DO 700 I=1,10
         IF(P(PTR,I).EQ.0) GOTO 800
         TCHNLS(TCNT)=P(PTR,I)
         TCNT=TCNT+1
  700    CONTINUE
      WRITE(6,2065)
C
C
C
  800 CONTINUE
      TCNT=TCNT-1
C
C     ****************************************************************
C     * WALK THE PATH SELECTED TO SEE IF IT IS AVAILABLE.  IF SO,    *
C     * TRANSFER THE PATH TO TCHNLS.  IF NOT, BRANCH TO ALTERNATE    *
C     * PATH GENERATOR.                                              *
C     ****************************************************************
C
      LIMIT=TCNT
      IYESNO=0
      CHNCNT=LINKTB(PKLINK(LNODE),DEST)
      IF (CHNCNT.GT.0) IYESNO=3
      DO 900 I=1,LIMIT
         IDELAY=IDELAY+PARAM(6)
         ROUT(TCHNLS(I))=RATIO
         IF (RATIO.LT.NLINES(TCHNLS(I))) GOTO 900
         IF (AVLTST.EQ.0) BTLNCK(TCHNLS(I))=BTLNCK(TCHNLS(I))+1
         IYESNO=1
         GOTO 200
  900    CONTINUE
C
C
C
  200 IF (PASS.EQ.2) GOTO 500
      DO 400 I=1,TCNT
         FTRACE(I)=TCHNLS(I)
  400    CONTINUE
      FLGTH=TCNT
      RETURN
C
C
C
  500 DO 510 I=1,TCNT
         RTRACE(I)=TCHNLS(I)
  510    CONTINUE
      RLGTH=TCNT
C
C
 2065 FORMAT (' TRANSFER ERROR')
C
```

```
END
```

```
C     ************************************************************
C     * THIS ROUTINE IS USED TO FIND A ROUTE THROUGH THE NETWORK.  *
C     * IT IS CALLED TWICE FOR EACH CIRCUIT SWITCH ROUTE.  THE     *
C     * STRATEGY TO BE FOLLOWED IS RANDOM ROUTING THEREFORE THIS   *
C     * ROUTINE WIL RANDOMLY CHOOSE A ROUTE FROM THE ARRAY P.      *
C     * IYESNO SIGNIFIES WHETHER A GOOD ROUTE WAS FOUND.           *
C     ************************************************************
C
C
C
      SUBROUTINE RTLIMT(LNODE,KDEST,IYESNO,IDELAY,CLASS,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      REAL SCORE(40),SCR,SGLSCR,RN
      DIMENSION NTRACE(160),TCHNLS(160),PSTRYS(20)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C
C     ************************************************************
C     * INITIALIZE THE TRACE VARIABLES.                          *
C     ************************************************************
```

```
C
   10 DO 84 I=1,160
         TCHNLS(I)=0
   84    CONTINUE
      TCNT=1
      IF (PASS.EQ.2) GOTO 80
      RLGTH=0
      FLGTH=0
      DO 85 I=1,160
         ROUT(I)=0
   85    CONTINUE
      DO 86 I=1,26
         FTRACE(I)=0
         RTRACE(I)=0
   86    CONTINUE
   80 NODE=LNODE
      DEST=KDEST
      IDELAY=0
      RATIO=1
      IYESNO=0
      CHNCNT=LINKTB(PKLINK(LNODE),DEST)
      IF (CHNCNT.GT.0) IYESNO=3
      IF (CLASS.EQ.2) RATIO=PARAM(4)
C
C     ****************************************************************
C     * SCORE EACH FEASIBLE PATH                                     *
C     ****************************************************************
C
      T1=FSTPTH(LNODE,KDEST)
      T2=LSTPTH(LNODE,KDEST)
      XFACT=.15
      DO 520 K=1,3
         XFACT=XFACT-.05
         DO 505 I=T1,T2
            DO 220 J=1, 10
               CHAN=P(I,J)
               IF (CHAN.EQ.0) GOTO 300
               UTIL=1.0*NLINES(CHAN)/PARM3(CHAN)
               IF (UTIL.LE.XFACT) GOTO 505
  220          CONTINUE
  300       GOTO 525
  505       CONTINUE
  520    CONTINUE
      IYESNO=1
      I=T1
C
C
C
  525 PTR=I
      TCNT=1
      DO 700 I=1,10
         IF(P(PTR,I).EQ.0) GOTO 800
```

```
         TCHNLS(TCNT)=P(PTR,I)
         TCNT=TCNT+1
  700    CONTINUE
       WRITE(6,2065)
C
C
C
  800 CONTINUE
       TCNT=TCNT-1
C
C      ****************************************************************
C      * WALK THE PATH SELECTED TO SEE IF IT IS AVAILABLE.  IF SO,    *
C      * TRANSFER THE PATH TO TCHNLS.  IF NOT, BRANCH TO ALTERNATE    *
C      * PATH GENERATOR.                                              *
C      ****************************************************************
C
       LIMIT=TCNT
       IYESNO=0
       DO 900 I=1,LIMIT
          IDELAY=IDELAY+PARAM(6)
          ROUT(TCHNLS(I))=RATIO
          IF (RATIO.LE.NLINES(TCHNLS(I))) GOTO 900
          IF (AVLTST.EQ.0) BTLNCK(TCHNLS(I))=BTLNCK(TCHNLS(I))+1
          IYESNO=1
          GOTO 200
  900     CONTINUE
  200  IF (PASS.EQ.2) GOTO 500
C
C
C
       DO 400 I=1,TCNT
          FTRACE(I)=TCHNLS(I)
  400     CONTINUE
       FLGTH=TCNT
       RETURN
C
C
C
  500 DO 510 I=1,TCNT
          RTRACE(I)=TCHNLS(I)
  510     CONTINUE
       RLGTH=TCNT
       RETURN
C
C
C
 2065 FORMAT (' TRANSFER ERROR')
C
C
C
       END
```

```
C     ************************************************************
C     *  THIS ROUTINE WILL IMPLEMENT THE STORE AND FORWARD ROUTING *
C     *  STRATEGY BY EITHER CALLING ROUTE AND USING ONLY A PART OF *
C     *  THE INFORMATION RETURNED OR BY CALLING THE APPROPRIATE    *
C     *  SINGLE STEP ROUTINE.                                      *
C     ************************************************************
C
C
C
      SUBROUTINE SGLSTP(NODE,DEST,NXCHAN,NXNODE,AVAIL,CLASS,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      INTEGER ONESTP(10)
C
C
C
      REAL CNFINT,UTLSTP(10)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
```

```
     A              FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      IF (STPPRI.EQ.1) GOTO 800
  100 IF (GLOBAL.EQ.0) GOTO 600
      IYESNO=0
      IDELAY=0
      PASS=1
      AVLTST=1
      CALL ROUTE(NODE,DEST,IYESNO,IDELAY,CLASS,PASS)
      NXCHAN=FTRACE(1)
      NXNODE=NODCHL(NXCHAN)
      AVAIL=1
      IF (IYESNO.EQ.1) AVAIL=0
      IF (IYESNO.EQ.2) AVAIL=0
      RETURN
C
C
C
  600 IF (SWITCH.EQ.1) CALL SSUTIL(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
      IF (SWITCH.EQ.2) CALL SSPRBK(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
      IF (SWITCH.EQ.3) CALL SSRAND(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
      IF (SWITCH.EQ.4) CALL SSQCNT(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
      IF (SWITCH.EQ.5) CALL SSLIMT(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
      RETURN
C
C
C
  800 FSTNOD=NODE
      LSTNOD=DEST
      INTNOD=DEST
      AVLTST=1
      IYESNO=0
      IDELAY=0
      PASS=1
      CALL ROUTE(FSTNOD,INTNOD,IYESNO,IDELAY,CLASS,PASS)
      IF (IYESNO.EQ.0) GOTO 840
      IF (IYESNO.EQ.3) GOTO 840
      LIMT=FLGTH-1
      DO 810 I=1,LIMT
         IF (NLINES(FTRACE(I)).LT.PARAM(4)) GOTO 812
  810    CONTINUE
      GOTO 840
C
C
C
  812 INTNOD=SORCHL(FTRACE(I))
C
C
C
  840 NXCHAN=FTRACE(1)
```

```
      NXNODE=INTNOD
      AVAIL=1
      IF (NODE.EQ.NXNODE) AVAIL=0
      RETURN
C
C
C
      END
```

```
C     ************************************************************
C     *  THIS ROUTINE APPLIES THE UTILIZATION (WEIGHTED) ROUTING   *
C     *  STRATEGY TO FINDING THE NEXT STEP IN THE PATH FROM NODE   *
C     *  TO DEST.                                                  *
C     ************************************************************
C
C
C
      SUBROUTINE SSUTIL(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      INTEGER ONESTP(10)
C
C
C
      REAL CNFINT,UTLSTP(10)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
```

```
      NODES=PARAM(1)/2
      CSNODE=PKLINK(NODE)
      CNT=0
      DO 100 I=1,NODES
         IF (KONECT(CSNODE,I).LE.0) GOTO 100
         IF (I.EQ.LSTNOD) GOTO 100
         CNT=CNT+1
         ONESTP(CNT)=I
  100    CONTINUE
      IF (CNT.GT.0) GOTO 200
  150 WRITE(6,1000)
      SSFLAG=0
      CNFINT=.01
      CALL SSANAL(SSFLAG,CNFINT)
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL SSTDMP
      CALL STATX
      CALL STATS
      STOP
C
C
C
  200 DO 250 I=1,CNT
         CHAN=KONECT(CSNODE,ONESTP(I))
         P3=PARM3(CHAN)
         NL=NLINES(CHAN)
         UTLSTP(I)=1.0*(P3-NL-PARAM(4))/P3
  250    CONTINUE
      UTLBST=100
      BSTPTR=0
      DO 275 I=1,CNT
         IF (UTLSTP(I).GE.UTLBST) GOTO 275
         UTLBST=UTLSTP(I)
         BSTPTR=I
  275    CONTINUE
      IF (BSTPTR.EQ.0) GOTO 150
C
C
C
      NXNODE=ONESTP(BSTPTR)
      AVAIL=1
      NXCHAN=KONECT(NODE,NXNODE)
      IF (PARAM(4).GT.NLINES(NXCHAN)) AVAIL=0
      RETURN
C
C
 1000 FORMAT (///,'   DEAD END - ROUTING ERROR ')
C
C
      END
```

```
C     ************************************************************
C     *  THIS ROUTINE WILL DETERMINE THE NEXT STEP TO BE TAKEN IN  *
C     *  THE ROUTE BETWEEN NODE AND DEST.                          *
C     ************************************************************
C
C
C
      SUBROUTINE SSPRBK(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
C
      CSNODE=PKLINK(NODE)
      NXCHAN=DESTAB(CSNODE,DEST)
      NXNODE=NODCHL(NXCHAN)
      LINLFT=NLINES(NXCHAN)-PARAM(4)
      NXNODE=CSLINK(NXNODE)
      IF (LINLFT.LT.0) GOTO 100
      AVAIL=1
      RETURN
C
C
C
  100 NXCHAN=DSTALT(CSNODE,DEST)
      NXNODE=NODCHL(NXCHAN)
      NXNODE=CSLINK(NXNODE)
      LINLFT=NLINES(NXCHAN)-PARAM(4)
      IF (LINLFT.LT.0) GOTO 200
```

```
      AVAIL=1
      RETURN
C
C
C
  200 AVAIL=0
      RETURN
      END
```

```
C     ************************************************************
C     *  THIS ROUTINE WILL DETERMINE THE NEXT STEP TO BE TAKEN IN  *
C     *  THE ROUTE BETWEEN NODE AND DEST.                          *
C     ************************************************************
C
C
C
      SUBROUTINE SSRAND(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      INTEGER ONESTP(10)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      TRYS=NMTRYS
      NODES=PARAM(1)/2
```

```
      CSNODE=PKLINK(NODE)
      CNT=0
      DO 100 I=1,NODES
         IF (KONECT(CSNODE,I).LE.0) GOTO 100
         CNT=CNT+1
         ONESTP(CNT)=I
  100    CONTINUE
   10 IF (CNT.GT.0) GOTO 200
      WRITE(6,1000)
      SSFLAG=0
      CNFINT=.01
      CALL SSANAL(SSFLAG,CNFINT)
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL SSTDMP
      CALL STATX
      CALL STATS
      STOP
C
C
C
  200 IX=RANDSD(NODE)
      CALL RANDOM(IX,IY,RN)
      RANDSD(NODE)=IY
      NEXDEX=(RN*CNT)+1
C
C
C
      AVAIL=1
      NXNODE=ONESTP(NEXDEX)
      NXCHAN=KONECT(NODE,NXNODE)
      IF (PARAM(4).GT.NLINES(NXCHAN)) AVAIL=0
C
C
C
      IF (AVAIL.EQ.1) RETURN
      TRYS=TRYS-1
      IF (TRYS.LE.0) RETURN
      NEXDEX=NEXDEX+1
      IF (NEXDEX.GT.CNT) GOTO 400
      DO 300 I=NEXDEX, CNT
         ONESTP(I-1)=ONESTP(I)
  300    CONTINUE
  400 CNT=CNT-1
      GOTO 10
C
C
 1000 FORMAT (///,'   DEAD END - ROUTING ERROR ')
C
C
      END
```

```
C     ************************************************************
C     *  THIS ROUTINE WILL DETERMINE THE NEXT STEP TO BE TAKEN IN  *
C     *  THE ROUTE BETWEEN NODE AND DEST.                          *
C     ************************************************************
C
C
C
      SUBROUTINE SSQCNT(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      INTEGER ONESTP(10)
C
C
C
      REAL CNFINT,UTLSTP(10)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      NODES=PARAM(1)/2
```

```
      CSNODE=PKLINK(NODE)
      CNT=0
      DO 100 I=1,NODES
         IF (KONECT(CSNODE,I).LE.0) GOTO 100
         IF (I.EQ.LSTNOD) GOTO 100
         CNT=CNT+1
         ONESTP(CNT)=I
  100    CONTINUE
      IF (CNT.GT.0) GOTO 200
  150 WRITE(6,1000)
      SSFLAG=0
      CNFINT=.01
      CALL SSANAL(SSFLAG,CNFINT)
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL SSTDMP
      CALL STATX
      CALL STATS
      STOP
C
C
C
  200 BSTPTR=0
      LOQCNT=10000
      DO 250 I=1,CNT
         CURNOD=ONESTP(I)
         IF (QCNT(CURNOD).GT.LOQCNT) GOTO 250
         LOQCNT=QCNT(CURNOD)
         BSTPTR=I
  250    CONTINUE
      IF (BSTPTR.EQ.0) GOTO 150
C
C
C
      AVAIL=1
      NXNODE=ONESTP(BSTPTR)
      NXCHAN=KONECT(CSNODE,NXNODE)
      IF (PARAM(4).GT.NLINES(NXCHAN)) AVAIL=0
      RETURN
C
C
C
 1000 FORMAT (///,'   DEAD END - ROUTING ERROR ')
C
C
C
      END
```

```
C     ************************************************************
C     *  THIS ROUTINE WILL DETERMINE THE NEXT STEP TO BE TAKEN IN  *
C     *  THE ROUTE BETWEEN NODE AND DEST.                          *
C     ************************************************************
C
C
C
      SUBROUTINE SSLIMT(NODE,DEST,NXCHAN,NXNODE,AVAIL,LSTNOD)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      INTEGER ONESTP(10)
C
C
C
      REAL CNFINT,UTLSTP(10)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      NODES=PARAM(1)/2
```

```
      CSNODE=PKLINK(NODE)
      CNT=0
      DO 100 I=1,NODES
         IF (KONECT(CSNODE,I).LE.0) GOTO 100
         IF (I.EQ.LSTNOD) GOTO 100
         CNT=CNT+1
         ONESTP(CNT)=I
  100    CONTINUE
      IF (CNT.GT.0) GOTO 200
  150 WRITE(6,1000)
      SSFLAG=0
      CNFINT=.01
      CALL SSANAL(SSFLAG,CNFINT)
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL SSTDMP
      CALL STATX
      CALL STATS
      STOP
C
C
C
  200 DO 250 I=1,CNT
         CHAN=KONECT(CSNODE,ONESTP(I))
         P3=PARM3(CHAN)
         NL=NLINES(CHAN)
         UTLSTP(I)=1.0*(P3-NL-PARAM(4))/P3
  250    CONTINUE
      XFACT=.15
      DO 290 K=1,3
         XFACT=XFACT-.05
         UTLBST=100
         BSTPTR=0
         DO 275 I=1,CNT
            IF (UTLSTP(I).LE.XFACT) GOTO 275
            IF (UTLSTP(I).GE.UTLBST) GOTO 275
            UTLBST=UTLSTP(I)
            BSTPTR=I
  275       CONTINUE
         IF (BSTPTR.NE.0) GOTO 295
  290    CONTINUE
      GOTO 150
C
C
C
  295 NXNODE=ONESTP(BSTPTR)
      AVAIL=1
      NXCHAN=KONECT(NODE,NXNODE)
      IF (PARAM(4).GT.NLINES(NXCHAN)) AVAIL=0
      RETURN
C
C
```

```
C
 1000 FORMAT (///,'   DEAD END - ROUTING ERROR ')
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE WILL UPDATE THE ROUTE TABLE REFLECT THE PATH
C     * TAKEN BY A PACKET OR CIRCUIT.
C     ************************************************************
C
C
C
C
      SUBROUTINE UPROUT(NODE,DEST,FQADDR,RQADDR,CLASS)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA5/FTRACE(26),RTRACE(26),FLGTH,RLGTH,SNAPON
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C     ************************************************************
C     * FIND THE PATH IN ARRAY P WHICH MATCHES THE SELECTED ROUTE  *
C     ************************************************************
C
      IBOT=FSTPTH(NODE,DEST)
      ITOP=LSTPTH(NODE,DEST)
      NUMNOD=PARAM(1)/2
      KNODE=NODE-NUMNOD
```

```
      KDEST=DEST-NUMNOD
C
      DO 100 PTHPTR=IBOT,ITOP
         SIZE=S(PTHPTR)
         IF (FLGTH.NE.SIZE) GOTO 100
         DO 50 COLUMN=1,SIZE
            IF (P(PTHPTR,COLUMN).NE.FTRACE(COLUMN)) GOTO 100
   50       CONTINUE
         GOTO 200
  100    CONTINUE
      CALL ERRMSG(15)
C
C
C
  200 IF (CLASS.EQ.2) QUEUE2(NODE,FQADDR+4)=PTHPTR
      IF (CLASS.EQ.1) CALQ2(KNODE,FQADDR)=PTHPTR
      SELCNT(PTHPTR)=SELCNT(PTHPTR)+1
      IF (CLASS.EQ.2) RETURN
C
C
C
      IBOT=FSTPTH(DEST,NODE)
      ITOP=LSTPTH(DEST,NODE)
C
      DO 300 PTHPTR=IBOT,ITOP
         SIZE=S(PTHPTR)
         IF (RLGTH.NE.SIZE) GOTO 300
         DO 250 COLUMN=1,SIZE
            IF (P(PTHPTR,COLUMN).NE.RTRACE(COLUMN)) GOTO 300
  250       CONTINUE
         CALQ2(KDEST,RQADDR)=PTHPTR
         SELCNT(PTHPTR)=SELCNT(PTHPTR)+1
         RETURN
  300    CONTINUE
      CALL ERRMSG(16)
      STOP
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS THE DRIVER FOR DATA/VOICE TRANSACTION      *
C     * TERMINATIONS.  IT FINDS THE CHANNEL ENTRY TO START THE     *
C     * REMOVAL PROCESS.  IT THEN CALLS REMOVE TO ACTUALLY PURGE   *
C     * TABLE ENTRIES.  TRAFFIC LOAD STATISTICS ARE THEN           *
C     * UPDATED BY THIS MODULE.                                    *
C     ************************************************************
C
C
C
      SUBROUTINE DEPART(LNODE,CLASS)
C
C
C
      IMPLICIT INTEGER (A-T)
      DIMENSION INDEX(2)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      NODE=LNODE
      DEST=EVTBL(NODE,4)
   70 QADDR=0
      CHPTR=0
      PASS=1
C
C     ************************************************************
C     * GO FIND STARTING POINT FOR REMOVAL OF THIS TRANSACTION.    *
C     ************************************************************
C
      CALL GETQ(NODE,DEST,CLASS,QADDR,CHPTR,PASS)
```

```
      IF (QADDR.EQ.0) GOTO 200
      INDEX(1)=QADDR
      IF (CLASS.EQ.2) GOTO 71
      PASS=2
      CALL GETQ(DEST,NODE,CLASS,QADDR,CHPTR,PASS)
      IF (QADDR.EQ.0) GOTO 200
      INDEX(2)=QADDR
C
C     ***************************************************************
C     * REMOVE FORWARD HALF OF CONNECTION AND THEN REVERSE HALF     *
C     ***************************************************************
C
   71 PASS=1
      CALL REMOVE(NODE,DEST,CLASS,PASS,INDEX(1))
      IF (CLASS.EQ.2) GOTO 72
      PASS=2
      CALL REMOVE(DEST,NODE,CLASS,PASS,INDEX(2))
C
C     ***************************************************************
C     * BRANCH TO PROPER HANDLER BASED ON CLASS OF TRANSACTION.     *
C     ***************************************************************
C
      IF (CLASS.EQ.1) GOTO 20
C
C     ***************************************************************
C     * IF IN STORE AND FORWARD MODE FOR PACKETS, CREATE A QUEUE    *
C     * ENTRY AT THE INTERMEDIATE DESTINATION FOR FORWARDING THE    *
C     * PACKETS AND UPDATE STATISTICS ACCORDINGLY.                  *
C     ***************************************************************
C
   72 IF (STOFWD.EQ.0) GOTO 25
C
C     ***************************************************************
C     * FIND A FREE QUEUE ENTRY FOR THE STEP.                       *
C     ***************************************************************
C
      ITOP=PARAM(13)
      DO 90 I=1,ITOP,6
         INDEX(2)=I
         IF (QUEUE1(DEST,I).EQ.0) GOTO 95
   90    CONTINUE
      CALL ERRMSG(13)
C
C
C
   95 I1=INDEX(1)
      I2=INDEX(2)
      NOTTHR=0
      LAST=QUEUE2(NODE,I1+1)
      IF (DEST.EQ.LAST) GOTO 25
      NOTTHR=1
      QCNT(DEST)=QCNT(DEST)+1
```

```
      LAPSE=QUEUE2(NODE,I1+5)-QUEUE1(NODE,I1+1)
C
      QUEUE1(DEST,I2)=1
      QUEUE1(DEST,I2+1)=QUEUE2(NODE,I1+5)
      QUEUE1(DEST,I2+2)=QUEUE2(NODE,I1+5)+LAPSE
      QUEUE1(DEST,I2+3)=QUEUE1(NODE,I1+3)
      QUEUE1(DEST,I2+4)=0
      QUEUE1(DEST,I2+5)=QUEUE2(NODE,I1+5)
C
      QUEUE2(DEST,I2)=QUEUE2(NODE,I1)
      QUEUE2(DEST,I2+1)=QUEUE2(NODE,I1+1)
      QUEUE2(DEST,I2+2)=QUEUE2(NODE,I1+2)
      QUEUE2(DEST,I2+3)=QUEUE2(NODE,I1+3)
      QUEUE2(DEST,I2+4)=QUEUE2(NODE,I1+4)
      QUEUE2(DEST,I2+5)=0
C
      NODLOD(DEST,1)=NODLOD(DEST,1)+QUEUE1(NODE,I1+3)
      NODLOD(DEST,2)=NODLOD(DEST,2)+QUEUE1(NODE,I1+3)
      NODLOD(DEST,3)=NODLOD(DEST,3)+QUEUE1(NODE,I1+5)
C
      FINDST=QUEUE2(NODE,I1+1)
      LSTNOD=NODE
      CALL SGLSTP(DEST,FINDST,ICHAN,INDEST,AVAIL,CLASS,LSTNOD)
      QUEUE1(DEST,I2+4)=INDEST
C
      DSTCNT(DEST,INDEST)=DSTCNT(DEST,INDEST)+QUEUE1(NODE,I1+5)
      DSTLOD(DEST,INDEST)=DSTLOD(DEST,INDEST)+QUEUE1(NODE,I1+3)
      CUMLOD(DEST,INDEST)=CUMLOD(DEST,INDEST)+QUEUE1(NODE,I1+3)
      CUMCNT(DEST,INDEST)=CUMCNT(DEST,INDEST)+QUEUE1(NODE,I1+5)
C
C
C
      CALL NUEVNT(DEST,CLASS)
C
C     ************************************************************
C     *  UPDATE THE PACKET NODE STATISTICS AND THEN PURGE THE    *
C     *  DATA QUEUE OF THE FORWARD LINK.  IF STORE AND FORWARD   *
C     *  MODE IS ON THEN THIS LINK WILL BE THE LAST STEP TAKEN   *
C     *  AND THE NEXT STEP WILL REMAIN INTACT.                   *
C     ************************************************************
C
   25 QCNT(NODE)=QCNT(NODE)-1
      K=INDEX(1)
      IF (DSTLOD(NODE,DEST)-QUEUE1(NODE,(K+3)).LT.0) GOTO 29
      DSTLOD(NODE,DEST)=DSTLOD(NODE,DEST)-QUEUE1(NODE,(K+3))
      DSTCNT(NODE,DEST)=DSTCNT(NODE,DEST)-QUEUE1(NODE,(K+5))
      NODLOD(NODE,1)=NODLOD(NODE,1)-QUEUE1(NODE,(K+3))
   29 ITOP=K+5
      DO 40 M=K,ITOP
         QUEUE1(NODE,M)=0
         QUEUE2(NODE,M)=0
   40    CONTINUE
```

```
   35 PARAM(9)=EVTBL(LNODE,1)
C
C
C
   80 RETURN
C
C
C
   20 KNODE=CSLINK(NODE)
      KDEST=DEST-(PARAM(1)/2)
C
C     ************************************************************
C     *  REMOVE CIRCUIT SWITCH TRANSACTION FROM CALQ TABLES.     *
C     ************************************************************
C
      CALQ1(KNODE,(INDEX(1)))=0
      CALQ1(KNODE,(INDEX(1)))=0
      CALQ1(KDEST,(INDEX(2)))=0
      CALQ1(KDEST,(INDEX(2)))=0
      KNODE=CALQ1(KDEST,(INDEX(2)+3))-(PARAM(1)/2)
      CALLS(KNODE,3)=CALLS(KNODE,3)-1
      CALL NUEVNT(DEST,CLASS)
      GOTO 35
C
C
C
   50 K=EVTBL(LNODE,5)
      ITOP=K+5
      DO 60 M=K,ITOP
         QUEUE1(LNODE,M)=0
         QUEUE2(LNODE,M)=0
   60    CONTINUE
      CALL ERRMSG(9)
      GOTO 80
C
C
C
  200 KNODE=CSLINK(NODE)
      IF (CLASS.EQ.2) GOTO 50
      CALQ1(KNODE,EVTBL(NODE,5))=0
      CALQ1(KNODE,EVTBL(NODE,5))=0
      CALL ERRMSG(8)
      RETURN
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE RIPPLES THROUGH ALL CHANNEL ENTRIES,        *
C     * ZEROING OUT ENTRIES PERTAINING TO THIS ROUTE.            *
C     ************************************************************
C
C
C
      SUBROUTINE REMOVE(LNODE,KDEST,CLASS,PASS,QADDR)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      JNODE=PKLINK(NODE)
      LINKTB(JNODE,KDEST)=LINKTB(JNODE,KDEST)-1
C
C
C
      NODE=LNODE
      DEST=KDEST
      IF (CLASS.EQ.1) KNODE=CSLINK(NODE)
      ILIM=1
      IF (CLASS.EQ.2) ILIM=PARAM(4)
C
      INDEXQ=QADDR
      IF (CLASS.EQ.1) PTHPTR=CALQ2(KNODE,INDEXQ)
      IF (CLASS.EQ.2) PTHPTR=QUEUE2(NODE,INDEXQ+4)
      SIZE=S(PTHPTR)+1
```

```
      COLUMN=0
C
   10 SIZE=SIZE-1
      IF (SIZE.EQ.0) GOTO 50
      COLUMN=COLUMN+1
      ICHAN=P(PTHPTR,COLUMN)
C
      IF (ICHAN.EQ.0) CALL ERRMSG(14)
C
C     ***************************************************************
C     *  LOOP THROUGH THIS SEQUENCE FOR EACH CHANNEL IN THE ROUTE.  *
C     ***************************************************************
C
      JCHNL=JARM3(ICHAN)-PARM3(ICHAN)+1
      ITOP=JARM3(ICHAN)
      I=JCHNL-1
      IF (PASS.EQ.1) TIME=EVTBL(NODE,1)
      IF (PASS.EQ.2) TIME=EVTBL(DEST,1)
C
C
C
      DO 15 J=1,ILIM
         JCHNL=I+1
         DO 20 I=JCHNL,ITOP
C
C           ********************************************************
C           *  CHECK FOR MATCH BY SOURCE, DEST, AND TIME.          *
C           ********************************************************
C
            IF (CHANTB(I,7).NE.NODE) GOTO 20
            IF (CHANTB(I,1).NE.DEST) GOTO 20
            IF (CHANTB(I,3).NE.TIME) GOTO 20
C
C           ********************************************************
C           *  HAVING FOUND CHANNEL MATCH, PURGE CHANNEL ENTRIES.  *
C           ********************************************************
C
   60       IF (CLASS.EQ.1) DIF=CHANTB(I,3)-CHANTB(I,2)
            BEGIN=QUEUE1(NODE,INDEXQ+1)
            IF (CLASS.EQ.2) DIF=QUEUE2(NODE,INDEXQ+5)-BEGIN
            MAX=PARAM(9)-PARAM(17)
            CHANTB(I,6)=CHANTB(I,6)+DIF
            IF (DIF.LE.MAX) GOTO 61
            CHANTB(I,6)=CHANTB(I,6)-DIF+MAX
   61       CONTINUE
            CHANTB(I,1)=0
            CHANTB(I,2)=0
            CHANTB(I,3)=0
            CHANTB(I,4)=0
            CHANTB(I,5)=0
            CHANTB(I,7)=0
            NLINES(ICHAN)=NLINES(ICHAN)+1
```

```
            GOTO 70
   20       CONTINUE
         ERRTYP=1
         CALL ERRMSG(ERRTYP)
   70    IF (CLASS.EQ.1) CHANTB(I,11)=CHANTB(I,11)+1
         IF (CLASS.EQ.1) GOTO 15
         PTR=CHANTB(I,8)
         CHANTB(I,9)=CHANTB(I,9)+QUEUE1(NODE,(PTR+5))
         CHANTB(I,10)=CHANTB(I,10)+QUEUE1(NODE,(PTR+3))
   15    CONTINUE
   30 GOTO 10
C
C
C
   50 RETURN
      END
```

```
C     ************************************************************
C     *  THIS PROCEDURE WILL PRINT ERROR MESSAGES FOR SELECTED     *
C     *  ERROR CONDITIONS THAT COULD OCCUR IN THE SIMULATION.      *
C     ************************************************************
C
C
C
      SUBROUTINE ERRMSG (ERRTYP)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      REAL CNFINT
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
C
      IF (ERRTYP.EQ.3) WRITE(6,1003)
      IF (ERRTYP.EQ.5) WRITE(6,1005)
      IF (ERRTYP.EQ.6) WRITE(6,1006)
      IF (ERRTYP.EQ.8) WRITE(6,1008)
      IF (ERRTYP.EQ.9) WRITE(6,1009)
      IF (ERRTYP.EQ.10) WRITE(6,1010)
      IF (ERRTYP.EQ.11) WRITE(6,1011)
      IF (ERRTYP.EQ.12) WRITE(6,1012)
      IF (ERRTYP.EQ.13) WRITE(6,1013)
      IF (ERRTYP.EQ.14) WRITE(6,1014)
      IF (ERRTYP.EQ.15) WRITE(6,1015)
      IF (ERRTYP.EQ.16) WRITE(6,1016)
      IF (ERRTYP.EQ.20) WRITE(6,1020)
      IF (ERRTYP.EQ.21) WRITE(6,1021)
      SSFLAG=0
```

```
      CNFINT=.01
      CALL SSANAL(SSFLAG,CNFINT)
      PARAM(10)=PARAM(9)
      PARAM(9)=0
      CALL SSTDMP
      CALL STATX
      CALL STATS
      STOP
C
C
C
 1003 FORMAT (' COULD NOT FIND NODE IN CHANTB - GETQ')
 1005 FORMAT (' CALQ2 QUEUE IS FULL - ARRIVE')
 1006 FORMAT (' CALQ2 QUEUE IS FULL - ARRIVE')
 1008 FORMAT (' COULD NOT FIND CS ENTRY IN CHANTB - DEPART')
 1009 FORMAT (' COULD NOT FIND PS ENTRY IN CHANTB - DEPART')
 1010 FORMAT (' PASS = 2 FOR A PS ENTRY - UPDATE')
 1011 FORMAT (' PTHPTR OR COLUMN WRONG - POINTS TO 0 CHANNEL - UPDATE')
 1012 FORMAT (' NO CAPACITY TO ALLOCATE - UPDATE')
 1013 FORMAT (' QUEUE FULL - DEPART')
 1014 FORMAT (' PTHPTR OR COLUMN WRONG - POINTS TO 0 CHANNEL - REMOVE')
 1015 FORMAT (' COULD NOT FIND FORWARD PATH IN P TABLE - UPROUT')
 1016 FORMAT (' COULD NOT FIND REVERSE PATH IN P TABLE - UPROUT')
 1020 FORMAT (' QUEUE FULL - NEWMSG')
 1021 FORMAT (' CHANTB FULL - UPDATE')
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE IS RESPONSIBLE FOR ARRIVAL OF A DATA/VOICE    *
C     * TRANSACTION AT THIS NODE.  IT IS THE DRIVER-WITH           *
C     * RESPONSIBILITY FOR GETTING A ROUTE, UPDATING CHANNEL       *
C     * TABLES AND GENERATING PACKET DELAY INFORMATION.            *
C     ************************************************************
C
C
C
      SUBROUTINE ARRIVE(LNODE,CLASS)
C
C
C
      IMPLICIT INTEGER (A-S)
      DIMENSION LDELAY(12)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
C
      DATA LDELAY/101,201,301,401,501,601,701,801,901,1001,2001,5001/
C
C
C
      FDELAY=0
      RDELAY=0
      NPCKTS=0
      IDELAY=0
      FLAG=1
      NODE=LNODE
      IF (EVTBL(NODE,1).GT.PARAM(9)) PARAM(9)=EVTBL(NODE,1)
      STIME=PARAM(9)
      INDEXQ=EVTBL(NODE,5)
```

```
      DEST=EVTBL(NODE,4)
      IF (CLASS.EQ.1) GOTO 45
      FSTARV=1
      LSTARV=1
      IF (STOFWD.EQ.0) GOTO 45
      LSTNOD=0
      CALL SGLSTP(NODE,DEST,NXCHAN,NXNODE,AVAIL,CLASS,LSTNOD)
      DEST=NXNODE
      EVTBL(NODE,4)=DEST
      QUEUE1(NODE,INDEXQ+4)=DEST
      IF (NODE.NE.QUEUE2(NODE,INDEXQ)) FSTARV=0
      IF (DEST.NE.QUEUE2(NODE,INDEXQ+1)) LSTARV=0
C
C
C
   45 IYESNO=0
      PASS=1
      AVLTST=0
      CALL ROUTE(NODE,DEST,IYESNO,IDELAY,CLASS,PASS)
      FDELAY=IDELAY
C
C     ****************************************************************
C     *  BRANCH IF CS TRANSACTION                                    *
C     ****************************************************************
C
      IF (CLASS.EQ.1) GOTO 30
C
C     ****************************************************************
C     * BRANCH TO PROPER CASE HANDLER BASED ON VALUE OF IYESNO       *
C     *                 0-NO EXISTING ROUTE, PATH AVAILABLE          *
C     *                 1-EXISTING ROUTE AVAILABLE, NO PATH          *
C     *                 2-NO EXISTING ROUTE AVAILABLE, NO PATH       *
C     *                 3-EXISTING PATH AVAILABLE, PATH AVAILABLE    *
C     ****************************************************************
C
      IF (IYESNO.EQ.2) GOTO 10
      IF (IYESNO.EQ.3) GOTO 20
      IF (IYESNO.EQ.0) GOTO 20
      IF (IYESNO.EQ.1) GOTO 10
C
C     ****************************************************************
C     *  NOW ADD DELAY TO CUMULATIVE TIME TABLE.                     *
C     ****************************************************************
C
   75 QDELAY=STIME-EVTBL(NODE,1)
      QUEUE2(NODE,INDEXQ+2)=QUEUE2(NODE,INDEXQ+2)+FDELAY+QDELAY
      IF (FSTARV.NE.1) GOTO 79
      QUEUE2(NODE,INDEXQ+2)=FDELAY
   79 IF (LSTARV.NE.1) GOTO 71
      FDELAY=QUEUE2(NODE,INDEXQ+2)
      ORGSRC=QUEUE2(NODE,INDEXQ)
      APCKTS(ORGSRC)=APCKTS(ORGSRC)+NPCKTS
```

```
      TDEL(ORGSRC)=TDEL(ORGSRC)+1.*FDELAY*NPCKTS/1000.
      DO 70 I=1,12
         IF (FDELAY.GT.LDELAY(I)) GOTO 70
         CUMTM(ORGSRC,I)=CUMTM(ORGSRC,I)+NPCKTS
         GOTO 71
   70    CONTINUE
      CUMTM(ORGSRC,13)=CUMTM(ORGSRC,13)+NPCKTS
C
C
C
   71 IF ((STOFWD.EQ.0).OR.(STPPRI.EQ.1)) GOTO 72
      DEST=PKLINK(DEST)
      NODE=PKLINK(NODE)
      ICHAN=DESTAB(DEST,NODE)
      IF (NLINES(ICHAN).NE.PARM3(ICHAN)) GOTO 180
      ICHAN=DSTALT(DEST,NODE)
      IF (NLINES(ICHAN).NE.PARM3(ICHAN)) GOTO 180
      ACKPAC(ICHAN)=ACKPAC(ICHAN)+1
      GOTO 180
C
C
C
   72 IF (LINKTB(DEST,NODE).GT.0) GOTO 180
      DEST=PKLINK(DEST)
      NODE=PKLINK(NODE)
   73 ICHAN=DESTAB(DEST,NODE)
      IF (ICHAN.LE.0) GOTO 180
      IF (NLINES(ICHAN).EQ.PARM3(ICHAN)) ACKPAC(ICHAN)=ACKPAC(ICHAN)+1
      DEST=NODCHL(ICHAN)
      IF (DEST.EQ.NODE) GOTO 180
      GOTO 73
C
C
C
  180 RETURN
C
C     ************************************************************
C     *  NO PATH IS AVAILABLE (IYESNO = 1 OR 2).                 *
C     ************************************************************
C
   10 INDEXQ=EVTBL(NODE,5)
      QUEUE1(NODE,INDEXQ)=QUEUE1(NODE,INDEXQ)+1
      GOTO 180
C     ************************************************************
C     *  A HALF DUPLEX CONNECTION IS AVAILABLE FOR THE PS PATH   *
C     *  THEREFORE PROCEED TO BUILD A NEW PS PATH.               *
C     ************************************************************
C
   20 IF (EVTBL(NODE,1).GE.STIME) GOTO 25
      IDELAY=IDELAY+STIME-EVTBL(NODE,1)
      FDELAY=FDELAY+STIME-EVTBL(NODE,1)
```

```
   25 INDEXQ=EVTBL(NODE,5)
      NPCKTS=QUEUE1(NODE,(INDEXQ+3))
      NMSGS=QUEUE1(NODE,(INDEXQ+5))
C
C     ***************************************************************
C     *  UPDATE NODE COUNTERS                                       *
C     ***************************************************************
C
      DEST1=DEST
      LENGTH=NPCKTS
      DSTCNT(NODE,DEST1)=DSTCNT(NODE,DEST1)+NMSGS
      DSTLOD(NODE,DEST1)=DSTLOD(NODE,DEST1)+LENGTH
      CUMLOD(NODE,DEST1)=CUMLOD(NODE,DEST1)+LENGTH
      CUMCNT(NODE,DEST1)=CUMCNT(NODE,DEST1)+NMSGS
      NODLOD(NODE,1)=NODLOD(NODE,1)+LENGTH
      NODLOD(NODE,2)=NODLOD(NODE,2)+LENGTH
      NODLOD(NODE,3)=NODLOD(NODE,3)+NMSGS
C
C     ***************************************************************
C     *  UPDATE QUEUE ENTRIES WITH INFORMATION IN CHANNEL TABLES    *
C     ***************************************************************
C
      QUEUE1(NODE,INDEXQ)=999999999
      QUEUE1(NODE,(INDEXQ+1))=QUEUE1(NODE,(INDEXQ+1))+IDELAY
      QUEUE2(NODE,(INDEXQ+5))=QUEUE1(NODE,(INDEXQ+2))+IDELAY
C
C     ***************************************************************
C     *  GO ACTUALLY UPDATE CHANNEL TABLES.                         *
C     ***************************************************************
C
   35 CALL UPROUT(NODE,DEST,INDEXQ,RINDEX,CLASS)
      PASS=1
      CSDEX=INDEXQ
      CALL UPDATE(NODE,DEST,CLASS,IDELAY,PASS,CSDEX)
      IF (CLASS.EQ.2) GOTO 75
      PASS=2
      CSDEX=RINDEX
      CALL UPDATE(DEST,NODE,CLASS,IDELAY,PASS,CSDEX)
      GOTO 180
C
C     ***************************************************************
C     *  CIRCUIT HANDLER - CLASS = 1.                               *
C     ***************************************************************
C
   30 NDEST=PARAM(1)/2
      IF (IYESNO.EQ.1) GOTO 40
      IF (IYESNO.EQ.2) GOTO 40
      IYESNO=0
      IDELAY=0
      PASS=2
      AVLTST=0
      CALL ROUTE(DEST,NODE,IYESNO,IDELAY,CLASS,PASS)
```

```
      IF (IYESNO.EQ.1) GOTO 40
      IF (IYESNO.EQ.2) GOTO 40
      ERRTYP=3
      KNODE=NODE - NDEST
      DO 432 I=1,50,4
         IF (CALQ1(KNODE,I).EQ.0) GOTO 436
  432    CONTINUE
      CALL ERRMSG(5)
  436 INDEXQ=I
      KNODE=DEST - NDEST
      DO 442 I=1,50,4
         IF (CALQ1(KNODE,I).EQ.0) GOTO 446
  442    CONTINUE
      CALL ERRMSG(6)
  446 RINDEX=I
      KNODE=NODE - NDEST
      CALLS(KNODE,1)=CALLS(KNODE,1)+1
      CALLS(KNODE,3)=CALLS(KNODE,3)+1
      GOTO 35
C
C     ****************************************************************
C     * THE CALL TO ROUTE RETURNED EITHER A 1 OR 2, THUS THE         *
C     * CIRCUIT SWITCHED ROUTE CANNOT BE USED AND WE MUST            *
C     * INCREMENT THE COUNTER AT CALLS(KNODE,2).                     *
C     ****************************************************************
C
   40 KNODE=NODE - NDEST
      CALLS(KNODE,2)=CALLS(KNODE,2)+1
      GOTO 180
C
C
C
      END
```

```
C     ************************************************************
C     * THIS ROUTINE UPDATES CHANTB FOR THE ROUTE SELECTED.      *
C     ************************************************************
C
C
C
C
      SUBROUTINE UPDATE(LNODE,LDEST,CLASS,IDELAY,PASS,CSDEX)
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      NODE=LNODE
      DEST=LDEST
      INDEX=EVTBL(NODE,5)
      INODE=NODE
      RATIO=1
      IF (CLASS.EQ.2) RATIO=PARAM(4)
      FLAG=1
      IF (CLASS.EQ.1) KNODE=INODE-(PARAM(1)/2)
      IF (CLASS.EQ.2) INODE=PKLINK(NODE)
C
C     ************************************************************
C     *  BRANCH TO CIRCUIT HANDLER IF CLASS = 1.                 *
C     ************************************************************
C
C
      IF (CLASS.EQ.1) GOTO 70
      IF (PASS.EQ.2) CALL ERRMSG(10)
C
```

```
C
C
      PTHPTR=QUEUE2(NODE,INDEX+4)
      SIZE=S(PTHPTR)+1
      COLUMN=0
C
C
C
   10 COLUMN=COLUMN+1
      SIZE=SIZE-1
      IF (SIZE.EQ.0) GOTO 50
      ICHAN=P(PTHPTR,COLUMN)
      IF (ICHAN.EQ.0) CALL ERRMSG(11)
C
C
C
   35 NLINES(ICHAN)=NLINES(ICHAN)-RATIO
      IF (NLINES(ICHAN).LT.0) CALL ERRMSG(12)
C
C
C
      JCHNL=JARM3(ICHAN)-PARM3(ICHAN)+1
      ITOP=JARM3(ICHAN)
      I=JCHNL-1
C
C     ************************************************************
C     *  EXECUTE THIS LOOP FOR EACH SLOT NEEDED ON THIS CHANNEL.   *
C     ************************************************************
C
      DO 15 J=1,RATIO
         JCHNL=I+1
C
C        ************************************************************
C        * FIND A FREE CHANNEL ENTRY IN THE TABLE.                  *
C        ************************************************************
C
         DO 20 I=JCHNL,ITOP
            IF (CHANTB(I,4).EQ.0) GOTO 25
   20       CONTINUE
         CALL ERRMSG(21)
C
C        ************************************************************
C        * HAVING FOUND AN EMPTY TABLE ENTRY UPDATE THE ENTRIES     *
C        * BASED ON WHETHER IT IS A FORWARD OR REVERSE PATH.        *
C        ************************************************************
C
   25    IF (FLAG.EQ.0) GOTO 40
         IF (CLASS.EQ.1) GOTO 26
         LINKTB(INODE,DEST)=LINKTB(INODE,DEST)+1
         GOTO 27
   80    IF (PASS.EQ.2) CHANTB(I,3)=CSARV(KDEST,2)+IDELAY
         IF (PASS.EQ.1) CHANTB(I,3)=CSARV(KNODE,2)+IDELAY
```

```
      IF (PASS.EQ.1) CHANTB(I,2)=EVTBL(NODE,1)+IDELAY
      IF (PASS.EQ.2) CHANTB(I,2)=EVTBL(DEST,1)+IDELAY
      GOTO 90
 26   CSCHNL=JCHNL
 27   FLAG=0
 40   CHANTB(I,1)=DEST
      IF (CLASS.EQ.1) GOTO 80
      IF (PASS.EQ.1) CHANTB(I,2)=QUEUE1(NODE,INDEX+1)
      IF (PASS.EQ.2) CHANTB(I,2)=QUEUE1(DEST,INDEX+1)
      IF (PASS.EQ.1) CHANTB(I,3)=QUEUE1(NODE,(INDEX+2))
      IF (PASS.EQ.2) CHANTB(I,3)=QUEUE1(DEST,(QADDR+2))
 90   CHANTB(I,4)=RATIO
      CHANTB(I,5)=NODCHL(ICHAN)
      CHANTB(I,7)=NODE
      CHANTB(I,8)=INDEX
      IF (CLASS.EQ.1) GOTO 16
 15   CONTINUE
 16 INODE=NODCHL(ICHAN)
    GOTO 10
C
C
C
 50 RETURN
C
C     ****************************************************************
C     *  CIRCUIT HANDLER - CLASS = 1.                                *
C     ****************************************************************
C
 70 I=CSDEX
    IF (I.EQ.0) CALL ERRMSG(3)
    COLUMN=0
130 IF (PASS.EQ.2) GOTO 140
110 CALQ1(KNODE,I)=CSARV(KNODE,2)+IDELAY
    CALQ1(KNODE,I+3)=LNODE
170 CALQ1(KNODE,I+1)=LDEST
    CALQ1(KNODE,I+2)=I
    INDEX=I
    PTHPTR=CALQ2(KNODE,I)
    SIZE=S(PTHPTR)+1
    GOTO 10
140 KDEST=DEST-(PARAM(1)/2)
    CALQ1(KNODE,I)=CSARV(KDEST,2)+IDELAY
    CALQ1(KNODE,I+3)=LDEST
    GOTO 170
C
C
C
    END
```

```
C     ************************************************************
C     * SUBROUTINE GETQ IS USED WHEN IT IS NECESSARY TO TERMINATE A *
C     * ROUTE.  IT WILL FIND THE STARTING CHANNEL ADDRESS FOR THE  *
C     * REMOVAL PROCESS. TO DO SO, IT WILL SCAN ALL POSSIBLE FIRST *
C     * LINKS, I.E. ALL NODES THAT ARE DIRECTLY CONNECTED TO THE   *
C     * SOURCE NODE (LNODE).                                       *
C     ************************************************************
C
C
C
      SUBROUTINE GETQ(LNODE,LDEST,CLASS,QADDR,CHPTR,PASS)
C
C
C
      IMPLICIT INTEGER (A-T)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA7/KONECT(52,52),ORPRMS(17),ORSDTB(52,4)
C
C
C
      QADDR=0
      NODE=LNODE
      DEST=LDEST
      IF (PASS.EQ.1) TIME=EVTBL(NODE,1)
      IF (PASS.EQ.2) TIME=EVTBL(DEST,1)
      ATOP=PARAM(1)
      DO 100 K=1,ATOP
         IF (KONECT(LNODE,K).LE.0) GOTO 100
         ICHAN=KONECT(LNODE,K)
         JCHNL=JARM3(ICHAN) - PARM3(ICHAN)+1
         ITOP=JARM3(ICHAN)
         DO 10 J=JCHNL, ITOP
C
```

```
C           ********************************************************
C           *  TRY TO MATCH UP SOURCE, DEST, AND TIME.             *
C           ********************************************************
C
            IF (CHANTB(J,7).NE.NODE) GOTO 10
            IF (CHANTB(J,1).NE.DEST) GOTO 10
            IF (CHANTB(J,3).NE.TIME) GOTO 10
            QADDR=CHANTB(J,8)
            CHPTR=J
            GOTO 30
   10       CONTINUE
  100    CONTINUE
C
C     ***************************************************************
C     * THE RETURN STATEMENT AT LINE 30 CAN BE REACHED IN TWO WAYS.*υ
C     * IF IT IS REACHED BY A BRANCH, THEN A QUEUE ADDRESS WAS      *
C     * FOUND AND THE ADDRESS IS TO BE RETURNED VIA QADDR. IF IT IS*
C     * REACHED BY THE TERMINATION OF THE ABOVE DO LOOP, NO QUEUE  *
C     * ADDRESS WAS FOUND. IN THIS CASE THE CALLING PROCEDURE WILL *
C     * RECOGNIZE THAT QADDR HAS NOT BEEN CHANGED BY GETQ.         *
C     ***************************************************************
C
C
    30 RETURN
       END
```

```
C     ************************************************************
C     *                                                          *
C     * THIS ROUTINE CAPTURES THE PERFORMANCE STATISTICS IN THE   *
C     * GLOBAL TABLE YS FOR LATER ANALYSIS.                      *
C     *                                                          *
C     ************************************************************
C
C
C
      SUBROUTINE STATK
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA8/XS(501,8),SSC,PRINCR,LSTSTP,
     8             WCFACT,WCINCR,WPFACT,WPINCR,LSTEVT
C
C
C
C
C     ************************************************************
C     *  CALCULATE THE THROUGHPUT AND UTILIZATION.               *
C     ************************************************************
C
      IF (PARAM(9).EQ.PARAM(17)) RETURN
      SSC=SSC+1
      ILIM=PARAM(2)
      NSITES=PARAM(1)/2
      UTOT=0.0
      UTIL3=0.0
      DO 10 I=1,ILIM
         JCHNL=JARM3(I)-PARM3(I)+1
```

```
      ITOP=JARM3(I)
      UTIL1=0.0
      PSENT1=ACKPAC(I)
      DO 20 J=JCHNL,ITOP
         DUMMY=CHANTB(J,6)
         IF (CHANTB(J,4).EQ.0) GOTO 80
         IF (CHANTB(J,2).GE.PARAM(9)) GOTO 80
         DUMMY=DUMMY+PARAM(9)-CHANTB(J,2)
         IF (CHANTB(J,2).GE.PARAM(17)) GOTO 80
         DUMMY=DUMMY-(PARAM(17)-CHANTB(J,2))
80       PSENT=CHANTB(J,10)
         PSENT1=PSENT1+PSENT
         UTIL=1.0*DUMMY/(PARAM(9)-PARAM(17))
         UTIL1=UTIL1+UTIL
20       CONTINUE
      UTOT=UTOT+UTIL1
      UTIL3=UTIL3+(PARM3(I)-NLINES(I))
10    CONTINUE
   UAVG=UTOT/JARM3(ILIM)
   XS(SSC,1)=UAVG
   XS(SSC,5)=UTIL3/JARM3(ILIM)
C
C    ****************************************************************
C    *  CALCULATE THE PACKET NODE STATISTICS.                       *
C    ****************************************************************
C
   ITOP=PARAM(1)/2
   IF (PARAM(8).EQ.0) GOTO 31
   SUMPAK=0
   TOTDEL=0.0
   DO 30 I=1,ITOP
      SUMPAK=SUMPAK+APCKTS(I)
      TOTDEL=TOTDEL+TDEL(I)
      IF (APCKTS(I).EQ.0) GOTO 30
      ZDELAY=TDEL(I)/APCKTS(I)
30    CONTINUE
   IF (SUMPAK.EQ.0) GOTO 35
   ZDAVG=TOTDEL/SUMPAK
   GOTO 35
31 ZDELAY=0.0
   ZDAVG=0.0
35 XS(SSC,6)=SUMPAK
   XS(SSC,7)=PARAM(9)
   IF (SSC.EQ.1) GOTO 37
   C1=XS(SSC,6)-XS(SSC-1,6)
   C2=XS(SSC,7)-XS(SSC-1,7)
   XS(SSC,8)=1.0*C1/C2
37 CONTINUE
   XS(SSC,2)=ZDAVG
   CURBTL=0
   DO 36 I=1,ILIM
      CURBTL=CURBTL+BTLNCK(I)
```

```
   36    CONTINUE
         XS(SSC,4)=CURBTL-LSTBTL
         LSTBTL=CURBTL
C
C     ****************************************************************
C     *  CALCULATE THE CS NODE STATISTICS.                           *
C     ****************************************************************
C
         BIGTOT=0
         ALOST=0
         DO 40 I=1,ITOP
            ITOT=CALLS(I,1)+CALLS(I,2)
            ILOST=CALLS(I,2)
            IF (ITOT.EQ.0) GOTO 50
   50 CONTINUE
            BIGTOT=BIGTOT+ITOT
            ALOST=ALOST+ILOST
   40    CONTINUE
         IF (BIGTOT.EQ.0) GOTO 71
         ZBLOCK=1.0*ALOST/BIGTOT
         GOTO 72
   71 ZCALLS=0.0
         ZBLOCK=0.0
   72 CONTINUE
         XS(SSC,3)=ZBLOCK
         RETURN
         END
```

```
C     ************************************************************
C     * THIS ROUTINE IT IS RESPONSIBLE FOR OUTPUT GENERATION OF    *
C     * STATISTICAL INFORMATION.                                   *
C     ************************************************************
C
C
C
C
      SUBROUTINE STATS
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
      NODES=PARAM(1)
C
C     ************************************************************
C     *  WRITE THE PACKET NODE INFORMATION.                        *
C     ************************************************************
C
C
      WRITE(6,2005)
      WRITE(6,2006)
      ITOP=PARAM(1)/2
      DO 30 I=1,ITOP
         WRITE(6,2007) I,(CUMTM(I,K),K=1,13)
   30    CONTINUE
      DO 50 I=2,ITOP
         DO 40 K=1,13
            CUMTM(I,K)=CUMTM(I-1,K)+CUMTM(I,K)
```

```
 40       CONTINUE
 50    CONTINUE
       CUMTOT=0
       DO 52 I=1,13
          CUMTOT=CUMTOT+CUMTM(ITOP,I)
 52    CONTINUE
       WRITE(6,2010) (1.0*CUMTM(ITOP,I)/CUMTOT,I=1,13)
       WRITE(6,2013)
       IUP=PARAM(13)-4
       DO 60 I=1,ITOP
          DO 90 J=2,IUP,6
             IF (QUEUE1(I,J).LE.PARAM(10)) GOTO 90
             JPTR=QUEUE1(I,J+3)
             NODLOD(I,1)=NODLOD(I,1)-100000
             DSTLOD(I,JPTR)=DSTLOD(I,JPTR)-10
             DSTCNT(I,JPTR)=DSTCNT(I,JPTR)-10000
 90          CONTINUE
          WRITE(6,2014) I,((NODLOD(I,K)/10000),K=1,3)
 60    CONTINUE
       WRITE(6,2015)
       ALLDST=PARAM(1)/2
       SUMCUM=0
       SUMDST=0
       DO 70 I=1,ALLDST
          DO 70 J=1,ALLDST
             IF (I.EQ.J) GOTO 70
             WRITE(6,2016) I,J,DSTLOD(I,J),(DSTCNT(I,J)/10000),
     *                     CUMLOD(I,J),(CUMCNT(I,J)/10000)
             SUMCUM=SUMCUM+(CUMCNT(I,J)/10000)
             SUMDST=SUMDST+(DSTCNT(I,J)/10000)
 70    CONTINUE
       WRITE(6,7000)SUMDST,SUMCUM
       WRITE(6,8010)
       DO 38 I=1,ILIM
          WRITE(6,8015) I,BTLNCK(I)
 38    CONTINUE
       WRITE(6,8020)
       DO 39 I=1,784
          WRITE(6,8025) I,S(I),F(I),L(I),SELCNT(I)
 39    CONTINUE
       WRITE(6,8030)
       RETURN
C
C
C
 2005 FORMAT ('1',40X,'PACKET NODE STATISTICS')
 2006 FORMAT ('0','NODE',5X,'DELAY(SECS)',4X,'<.1',4X,'<.2',4X,
     1        '<.3',4X,'<.4',4X,'<.5',4X,'<.6',4X,'<.7',4X,'<.8',4X,
     2        '<.9',4X,'< 1',4X,'< 2',4X,'< 5',4X,'> 5')
 2007 FORMAT (' ',2X,I2,14X,I9,11I7,I9)
 2010 FORMAT (' ',21X,13F7.4)
 2013 FORMAT ('-','NODE CURRENT PACKET LOAD CUMULATIVE PACKET ',
```

```
     1        'LOAD CUMULATIVE TRANSACTIONS')
 2014 FORMAT (' ',2X,I2,10X,I10,13X,I10,14X,I10)
 2015 FORMAT ('-','NODE DEST CURRENT PACKET LOAD CURRENT ',
     1        'TRANSACTION LOAD CUMULATIVE PACKET LOAD CUMULATIVE',
     2        ' TRANSACTION LOAD')
 2016 FORMAT (' ',2X,I2,3X,I2,10X,I10,15X,I10,13X,I10,18X,I10)
 7000 FORMAT(45X,'----------',41X,'----------',/,45X,I10,41X,I10)
 8010 FORMAT (///,'    BOTTLNECKS')
 8015 FORMAT (I4,I10)
 8020 FORMAT (///,'  I   SIZE  FIRST LAST     SELCNT')
 8025 FORMAT (I4,3I6,I10)
 8030 FORMAT (/////,'    ')
C
C
C
      END
```

```
C     ************************************************************
C     *                                                          *
C     * THIS ROUTINE PRINTS OUT PERFORMANCE STATISTICS IN AN     *
C     * ABBREVIATED FORM.                                        *
C     *                                                          *
C     ************************************************************
C
C
C
C
C
      SUBROUTINE STATX
C
C
C
      IMPLICIT INTEGER (A-S)
C
C
C
      COMMON/AREA1/EVTBL(52,5),PKLINK(26),PARAM(17),CHANTB(1170,11),
     1             QUEUE1(10,1800),CALQ1(10,50),CUMTM(26,13),QCNT(52),
     1             CALLS(26,3),LINKTB(52,52),SEEDTB(52,4),NLINES(160),
     1             SORCHL(160),NODCHL(160),CUMLOD(26,26),CSARV(26,3),
     1             NODLOD(26,3),DSTLOD(26,26),DSTCNT(26,26),CSLINK(52)
C
C
      COMMON/AREA2/CUMCNT(26,26),ROUT(160),APCKTS(26),TDEL(26),SWITCH,
     2             ZDBLK(26,2),PAKAVG,UAVG,PAKTHR,ZDAVG,ZBLOCK,RSFLAG,
     2             THRUTL(160,2),PARM3(160),JARM3(160),STOFWD,EVTX(6),
     2             BOUND,GLOBAL,STPPRI,AVLTST,RESTRT,NUSWCH,NUPOWR,
     2             NUSTFD,NUGLOB,NUSTPP,WCFCNU,WPFCNU,QUEUE2(10,1800)
C
C
      COMMON/AREA3/ACKPAC(26),BTLNCK(52),LSTBTL,CALQ2(10,50)
C
C
      COMMON/AREA4/DESTAB(52,52),DSTALT(52,52),ALTCH(160)
C
C
      COMMON/AREAA/P(800,10),S(800),F(800),L(800),SELCNT(800),NMTRYS,
     A             FSTPTH(26,26),LSTPTH(26,26),RANDSD(52),POWER,PTHCNT
C
C
C
C     ************************************************************
C     *  WRITE THE FINAL PARAMETERS.                             *
C     ************************************************************
C
      WRITE(6,3000)
      WRITE(6,3011)
      WRITE(6,3012)
      WRITE(6,3015) (PARAM(I),I=1,16)
C
```

```
C     ****************************************************************
C     *  CALCULATE THE THROUGHPUT AND UTILIZATION.                   *
C     ****************************************************************
C
      PASS=1
      WRITE(6,7001)
    1 WRITE(6,7002)
      ILIM=PARAM(2)
      NSITES=PARAM(1)/2
      PAKTOT=0
      UTOT=0.0
      DO 10 I=1,ILIM
         JCHNL=JARM3(I)-PARM3(I)+1
         ITOP=JARM3(I)
         UTIL1=0.0
         PSENT1=ACKPAC(I)
         DO 20 J=JCHNL,ITOP
            DUMMY=CHANTB(J,6)
            IF (CHANTB(J,4).EQ.0) GOTO 80
            IF (CHANTB(J,2).GE.PARAM(10)) GOTO 80
            DUMMY=DUMMY+PARAM(10)-CHANTB(J,2)
   80       PSENT=CHANTB(J,10)
            PSENT1=PSENT1+PSENT
            UTIL=1.0*DUMMY/(PARAM(10)-PARAM(17))
            UTIL1=UTIL1+UTIL
   20       CONTINUE
         UTIL1=UTIL1/PARM3(I)
         NS=SORCHL(I)-NSITES
         ND=NODCHL(I)-NSITES
         WRITE(6,7003) I,PSENT1,UTIL1,NS,ND,PARM3(I)
         THRUTL(I,1)=PSENT1
         THRUTL(I,2)=UTIL1
         PAKTOT=PAKTOT+PSENT1
         UTOT=UTOT+UTIL1
   10    CONTINUE
      PAKAVG=PAKTOT/ILIM
      UAVG=UTOT/ILIM
      XSECS=(PARAM(10)-PARAM(17))/1000.
      PAKTHR=PAKAVG/XSECS+0.5
      WRITE(6,7004) PAKAVG,UAVG,PAKTHR
C
C
C
      IF (PASS.GT.1) GOTO 28
      WRITE(6,7014)
      ACKTOT=0
      DO 27 I=1,ILIM
         WRITE(6,7015) I,ACKPAC(I)
         ACKTOT=ACKTOT+ACKPAC(I)
         ACKPAC(I)=0
   27    CONTINUE
      PASS=PASS+1
```

```
      WRITE(6,7017) ACKTOT
      WRITE(6,7016)
      GOTO 1
C
C     ****************************************************************
C     *  CALCULATE THE PACKET NODE STATISTICS.                       *
C     ****************************************************************
C
   28 WRITE(6,7005)
      WRITE(6,7006)
      ITOP=PARAM(1)/2
      IF (PARAM(8).EQ.0) GOTO 31
      QTOT=0
      SUMPAK=0
      TOTDEL=0.0
      DO 30 I=1,ITOP
         QTOT=QTOT+QCNT(I)
         SUMPAK=SUMPAK+APCKTS(I)
         TOTDEL=TOTDEL+TDEL(I)
         ZDELAY=TDEL(I)/APCKTS(I)
         WRITE(6,7007) I,ZDELAY,QCNT(I)
         ZDBLK(I,1)=ZDELAY
   30    CONTINUE
      ZQAVG=1.0*QTOT/ITOP
      ZDAVG=TOTDEL/SUMPAK
      GOTO 35
   31 ZDELAY=0.0
      DO 32 I=1,ITOP
         WRITE(6,7007) I,ZDELAY,QCNT(I)
         ZDBLK(I,1)=ZDELAY
   32    CONTINUE
      ZDAVG=0.0
      ZQAVG=0.0
   35 CONTINUE
      WRITE(6,7008)ZDAVG,ZQAVG
C
C     ****************************************************************
C     *  CALCULATE THE CS NODE STATISTICS.                           *
C     ****************************************************************
C
   53 WRITE(6,7009)
      WRITE(6,7010)
      BIGTOT=0
      ALOST=0
      AKEPT=0
      DO 40 I=1,ITOP
         K=ITOP+I
         ITOT=CALLS(I,1)+CALLS(I,2)
         ILOST=CALLS(I,2)
         IKEPT=CALLS(I,3)
         UTIL=0.0
         IF (ITOT.EQ.0) GOTO 50
```

```
      UTIL=1.0*CALLS(I,2)/ITOT
   50 WRITE(6,7011) K,ITOT,ILOST,UTIL,IKEPT
      ZDBLK(I,2)=UTIL
      BIGTOT=BIGTOT+ITOT
      ALOST=ALOST+ILOST
      AKEPT=AKEPT+IKEPT
   40 CONTINUE
      IF (BIGTOT.EQ.0) GOTO 71
      ZCALLS=1.0*BIGTOT/ITOP
      ZBLOCK=1.0*ALOST/BIGTOT
      ZSYS=1.0*AKEPT/ITOP
      GOTO 72
   71 ZCALLS=0.0
      ZBLOCK=0.0
      ZSYS=0.0
   72 WRITE(6,7012)ZCALLS,ZBLOCK,ZSYS
C
C     ****************************************************************
C     *  WRITE OUT THE EVENT TYPE FREQUENCIES.                       *
C     ****************************************************************
C
   73 WRITE(6,7013) (EVTX(I),I=1,4)
      RETURN
C
C
C
 3016 FORMAT (' ',2X,I2,3X,I2,10X,I10,15X,I10,13X,I10,18X,I10)
 3000 FORMAT ('1',40X,'SYSTEM PARAMETERS')
 3011 FORMAT ('0',1X,'NODES LINKS SLOTS RATIO SLOT NODE  CS',6X,
     1        'PS  MSG START TIME END TIME PACKET  VDR RATES ',
     2        'Q SIZE   CS      PACKET PACKETS')
 3012 FORMAT (' ',25X,'TIME DELAY ARRIVAL ARRIVAL',21X,
     1        'LOADING',19X,'SERVICE SIZE   PER MSG')
 3015 FORMAT (' ',1X,3(I5,1X),I2,2X,I4,'MS ',I2,' MS',3X,I2,
     1        'MIN',3X,I2,'SEC',2X,I5,' MS',I8,'MS ',I7,1X,
     2        I5,'KBS ',I7,3X,I3,'SEC ',I4,'B',2X,I2)
 7001 FORMAT (///,5X,'PASS=1     ACKPACS COUNTED',///)
 7002 FORMAT (1H0,5X,'CHAN',3X,'THROUGHPUT',3X,'UTILIZATION',
     1         1X,'SOURCE',3X,'DEST',3X,'SLOTS')
 7003 FORMAT (1H ,5X,I3,3X,I10,4X,F8.3,3(4X,I4))
 7004 FORMAT (1H0,9X,'AVG NO OF PACKETS PER LINK=',I10/10X,
     1         'AVG LINK UTILIZATION =',F6.3/10X,
     2        'AVG LINK THROUGHPUT (PACKETS/SEC)=',I10//)
 7005 FORMAT (1H0,40X,'PACKET NODE SUMMARY'//)
 7006 FORMAT (1H ,5X,'NODE',3X,'AVG PACKET DELAY (SEC)',3X,
     1        'DATA TRANSACTIONS IN SYSTEM'/)
 7007 FORMAT (1H ,5X,I3,8X,F10.3,15X,I10)
 7008 FORMAT (1H0,9X,'AVG PACKET DELAY (SEC)=',F8.3/10X,
     1        'AVG NO OF DATA TRANSACTIONS AT A NODE=',F8.1//)
 7009 FORMAT (1H0,40X,'CS NODE SUMMARY'//)
 7010 FORMAT (1H ,'NODE',5X,'TOTAL CALLS',5X,'CALLS LOST',9X,
     1        'BLOCKING',5X,'CALLS IN SYSTEM')
```

```
 7011 FORMAT (1H ,2X,I2,11X,I5,10X,I5,12X,F5.3,10X,I5)
 7012 FORMAT (1H0,9X,'AVG NO OF CALLS PER NODE=',F8.1/10X,
     1        'FRACTION OF CALLS BLOCKED=',F9.3/10X,
     2        'AVG NO OF CALLS IN SYSTEM PER NODE=',F8.1)
 7013 FORMAT (1H0,19X,'CLASS 2 (DATA) ARRIVALS =',I10/20X,
     1        'CLASS 2 (DATA) DEPARTS  =',I10/20X,
     2        'CLASS 1 ( CS ) ARRIVALS =',I10/20X,
     3        'CLASS 1 ( CS ) DEPARTS  =',I10)
 7014 FORMAT (///,'   CHANNELS     ACKPACS')
 7015 FORMAT (' ',2I10)
 7016 FORMAT (///,'     PASS=2    ACKPACS NOT COUNTED',///)
 7017 FORMAT (/,'    TOTAL = ',I10)
C
C
C
      END
```

APPENDIX B

DESCRIPTION OF TABLES

The following diagrams present the structure of the major tables used by FLO and describe the components of each. Many of the tables used by FLO are the same as those used by Clabaugh's original simulation [22] and, as such, the descriptions which follow are based on his dissertation and that of Kiemele [65].

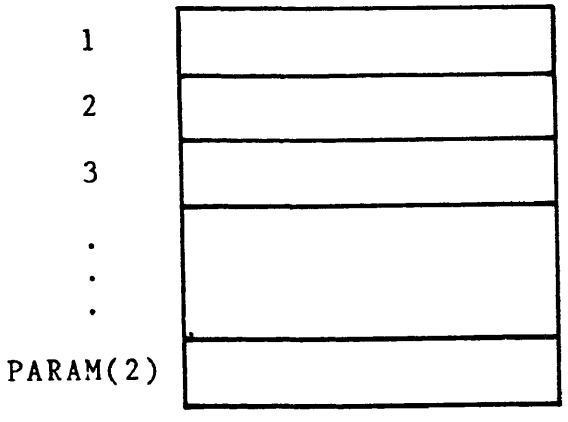
Table B-1. Parameter Table (PARAM [X])

This single dimension array contains the basic parameters used to control SIMULA execution. The values are set by FLO prior to initiating SIMULA.

1	Total number of PS and CS nodes
2	Total number of HDX channels
3	Total number of slots in the network
4	Ratio of packet to voice slots
5	Frame time duration
6	Fixed time routing delay per node
7	Circuit switch voice arrival rate
8	Packet switch transaction arrival rate
9	Start time for simulation run
10	Ending time for simulation run
11	Packet switch saturation level
12	Voice digitization rate
13	Buffer size at each packet switch
14	Average voice call service time
15	Number of bits per packet
16	Average number of packets per message
17	System error run time

Table B-2. Slot Table (PARM3 [X])

This single dimension array contains the number of slots allocated to HDX channel X, where X = 1, 2, 3, ... , PARAM(2).

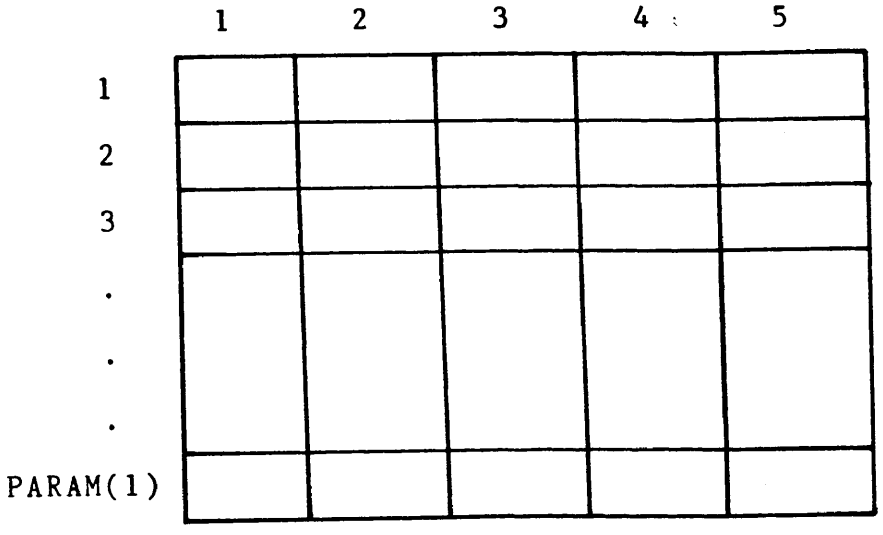


The relationship between PARM3 and PARAM(3) is:

$$PARAM(3) = \sum_{i=1}^{PARAM(2)} PARM3(i)$$

Table B-3. Event table (EVTBL[Node,Entry])

This table maintains the next event occurrence at each node. The [Node, Entry] table entries are:



Entry	Definition
1	Time (Msec)
2	Type: 1 = Class II (data) arrival 2 = Class II (data) departure 3 = Class I (voice) arrival 4 = Class I (voice) departure
3	Message length if Class II or time of departure if Class I
4	Final destination
5	Queue address (pointer into QUEUE1 and QUEUE2)

Table B4. Destination Table (DESTAB [Node,Dest])

This table gives the primary routing channel between each node pair [Node,Dest]. If the source node is a packet node, the [Node,Dest] entry contains the number of the directly connected circuit switch node instead of a channel number.

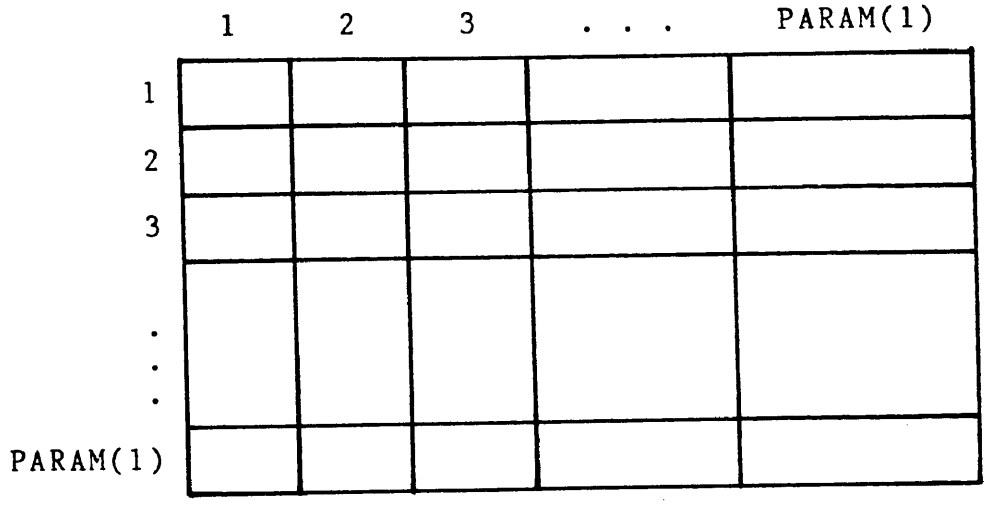


Table B5. Alternate Destination Table (DSTALT[Node,Dest])

Similar to table B4, this table provides an alternate channel between each nodal pair [Node,Dest].

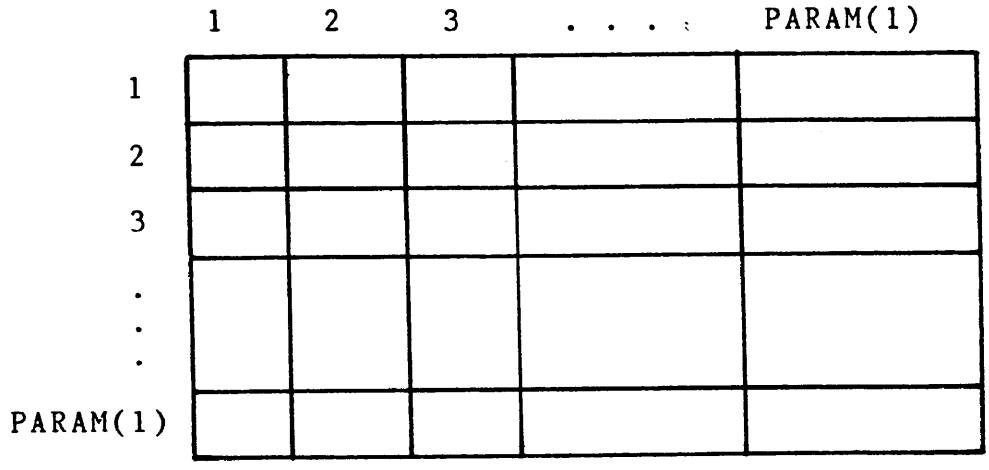
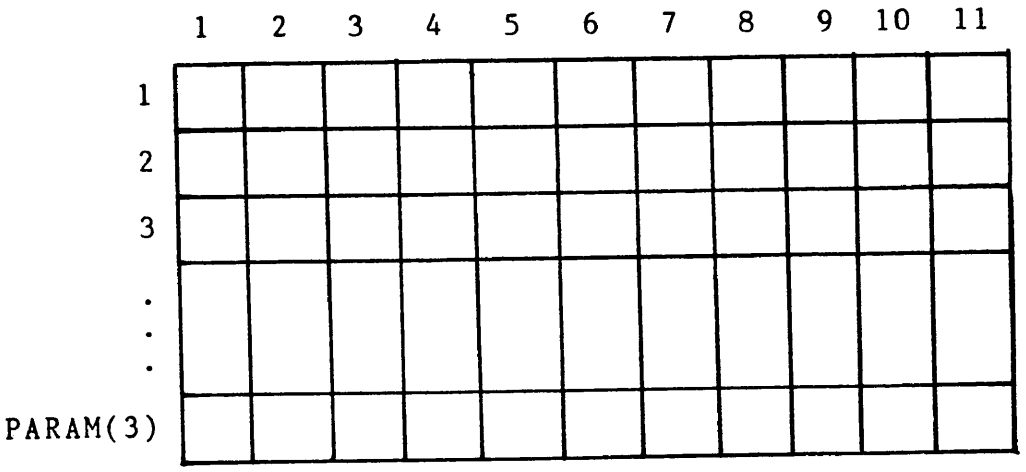


Table B6. Channel Table (CHANTB[Channel,Entry])

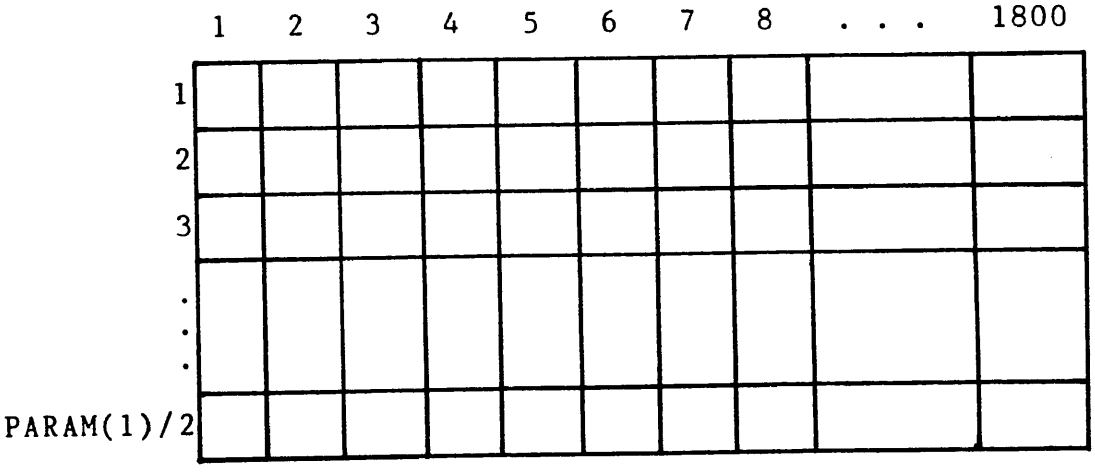
Each row of this table represents a particular slot in the network. For each slot, this table maintains the 11 attributes shown.



Entry	Definition
1	Final destination
2	Time slot is active
3	Time slot is available
4	Number of slots used for transaction
5	Intermediate destination
6	Cumulative time in use
7	Source node
8	Queue address
9	Cumulative number of transactions
10	Cumulative number of packets
11	Cumulative number of voice calls

Table B7. Queue Tables (QUEUE1[Node,Entry] and QUEUE2[Node,Entry])

This table exists for packet switched nodes only. Each row corresponds to a packet switch node and maintains a record of data transactions at that node. Each data transaction requires twelve entries, six in QUEUE1 and six in QUEUE2. By convention, the first six entries of both tables are associated with the first transaction, the second six with the second transaction and so on. Both tables are dimensioned appropriately based on the user specified arrival rate.

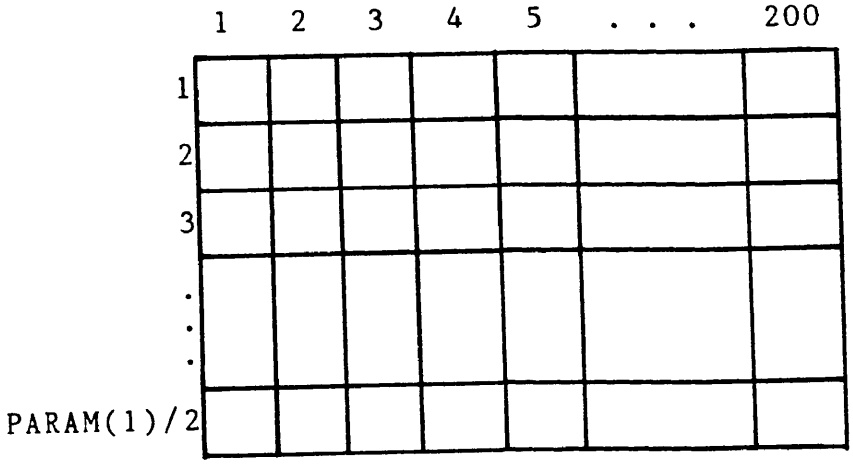


QUEUE1	Definition
1	Priority
2	Transaction arrival time
3	Transaction departure time
4	Total packet length
5	Intermediate Destination
6	Number of messages

QUEUE2	Definition
1	Source Node
2	Final Destination
3	Percentage of Message Allocated
4	Total delay
5	Route pointer
6	Departure Time + Routing Delay

Table B8. Call Queue Tables (CALQ1[Knode,Entry])

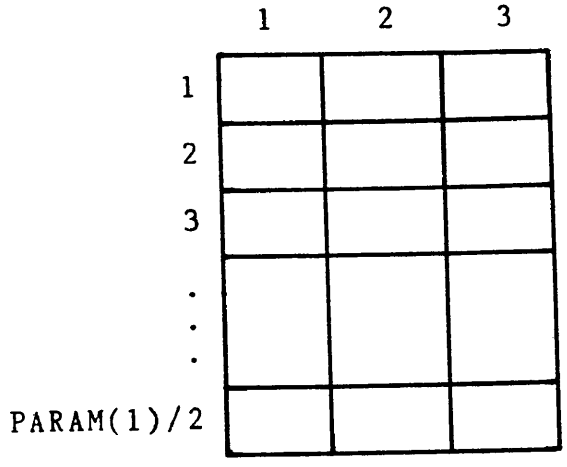
This table exists for circuit switched nodes only. Each row corresponds to a cicuit switch node and contains a record of all departing calls at that node. Each call requires four entries each of which is described below. By convention, the first four entries are associated with the first transaction, the second four with the second transaction and so on.



Entry	Definition
1	Departure Time
2	Destination
3	Channel address pointer
4	Source or Destination node

Table B9. Circuit Switch Arrival Table (CSARV[Knode,Entry])

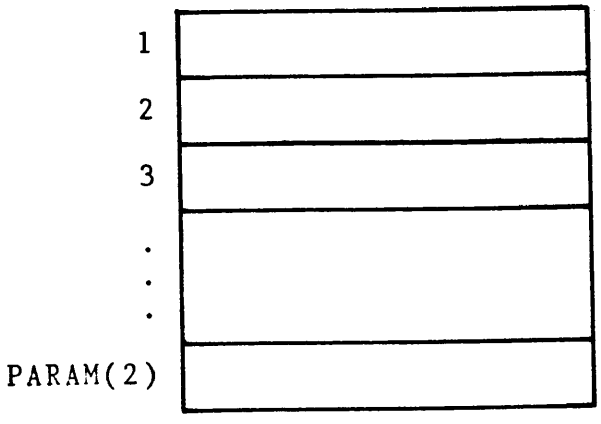
This table exists for circuit switched nodes only and contains information relating to the next voice call arrival at a node.



Entry	Definition
1	Time of Arrival
2	Time of Departure
3	Destination

Table B10. Link Availability Table (NLINES[Channel])

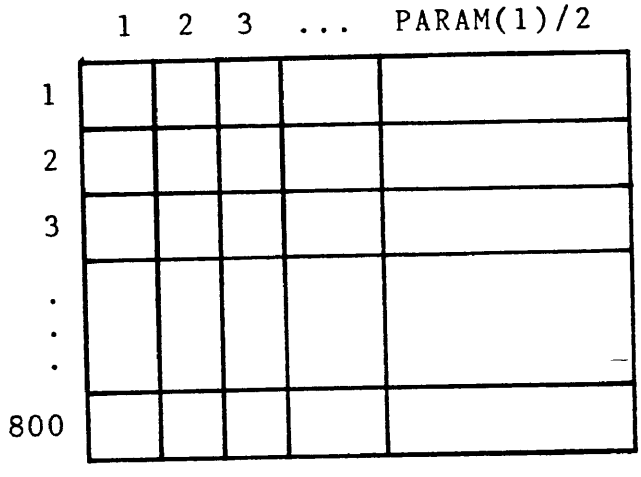
Each independent half duplex channel can be thought of as consisting of a number of slots. This table is a working table which shows the current count of the number of available slots for each channel.



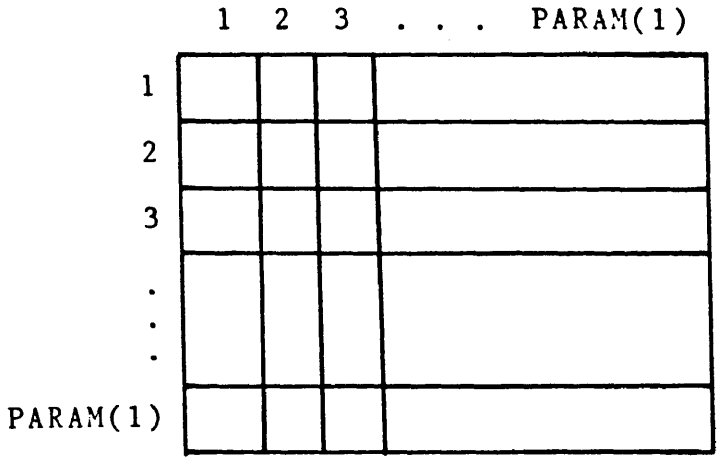
Initially, all slots are available and the table is initialized to be equal to PARM3.

Table B11. Path Tables (P[Path,Param(1)/2], FSTPTH[Node,Dest], LSTPTH[Node,Dest])

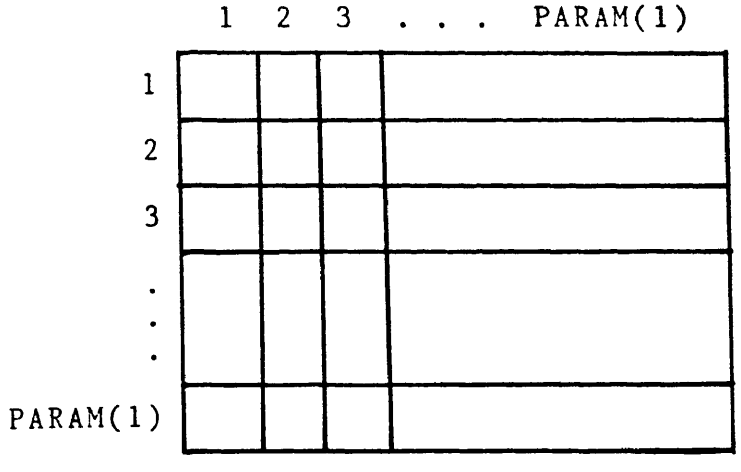
These three tables combine to present the valid paths which connect any pair of nodes. FSTPTH provides the first row in table P connecting Node to Dest, while LSTPTH provides the last row. All rows in table P between the two identified and including them are valid paths connecting these two nodes. Each row contains the link identifiers which compose the valid path. P is dimensioned to PARAM(1)/2 since this is the maximum size of a valid path. The rows connecting any pair of nodes are sorted from shortest to longest. P is sized to include enough rows to contain all valid paths.



P



FSTPTH



LSTPTH

APPENDIX C

DATA TABLES

The following tables present summary statistics for the experimental runs completed. For all runs, data was collected when the number of packets which had entered the system reached 40,000.

Workload = 3/30

Metric	Prbk	Pron	Limt	Qcnt-1	Qcnt-2	Util-1	Util-1
End Time (Secs)	4,020.764	4,012.422	4,020.239	4,012.270	3,981.289	4,020.239	4,020.239
Pkts sent(kpkt)	10,803.075	10,777.419	10,807.841	10,774.539	10,699.222	10,807.841	10,807.841
Pkt Thru-put	2,686.821	2,686.013	2,688.358	2,685.397	2,687.376	2,688.358	2,688.358
Link Thru-put	643.	640.	649.	651.	743.	644.	646.
Avg Pkt del	0.106	0.117	0.105	0.119	0.158	0.104	0.104
Blk fact	0.001	0.007	0.000	0.006	0.015	0.000	0.000
Departs/sec	9.937	9.917	9.941	9.933	9.938	9.941	9.941
Departs (%)	0.999	0.996	0.999	0.996	0.989	0.999	0.999
% del > 5 sec	0.000	0.000	0.000	0.000	0.000	0.000	0.000
% del > 2 sec	0.000	0.002	0.000	0.001	0.004	0.000	0.000
% del > 1 sec	0.001	0.007	0.000	0.007	0.018	0.000	0.000
Avg Trans/node	5.4	16.0	4.3	15.7	44.2	4.3	4.3

Workload = 4/40

Metric	Prbk	Pron	Limt	Qcnt-1	Qcnt-2	Util-1	Util-1
End Time (Secs)	3,984.461	3,975.458	3,964.963	3,966.497	3,897.955	4,008.829	4,010.057
Pkts sent(kpkt)	14,267.780	14,243.568	14,207.071	14,203.206	13,954.388	14,361.776	14,365.210
Pkt Thru-put	3,580.856	3,582.875	3,583.154	3,966.497	3,579.925	3,582.536	3,582.297
Link Thru-put	868.	851.	967.	865.	982.	907.	907.
Avg Pkt del	0.160	0.194	0.172	0.210	0.361	0.121	0.119
Blk fact	0.018	0.031	0.009	0.042	0.086	0.000	0.001
Departs/sec	9.930	9.933	9.936	9.927	9.924	9.931	9.932
Departs (%)	0.989	0.987	0.985	0.984	0.967	0.995	0.996
% del > 5 sec	0.000	0.001	0.000	0.001	0.002	0.000	0.000
% del > 2 sec	0.009	0.016	0.007	0.018	0.043	0.001	0.001
% del > 1 sec	0.025	0.040	0.025	0.046	0.104	0.005	0.004
Avg Trans/node	44.4	52.3	61.2	63.3	132.5	19.7	18.2

Workload = 5/50

Metric	Prbk	Pron	Limt	Qcnt-1	Qcnt-2	Util-1	Util-1
End Time (Secs)	3,901.527	3,923.684	3,806.923	3,905.863	3,847.765	3,832.970	3,839.137
Pkts sent(kpkt)	17,457.296	17,557.856	17,039.680	17,468.096	17,214.608	17,140.064	17,181.984
Pkt thru-put	4,474.478	4,474.840	4,475.972	4,472.276	4,473.924	4,471.745	4,475.481
Link Thru-put	1,113.	1,064.	1,344.	1,080.	1,232.	1,327.	1,319.
Avg Pkt del	0.391	0.394	0.616	0.431	0.655	0.536	0.485
Blk fact	0.074	0.097	0.101	0.120	0.180	0.043	0.042
Departs/sec	9.922	9.920	9.922	9.915	9.919	9.929	9.923
Departs (%)	0.968	0.973	0.944	0.968	0.954	0.951	0.952
% del > 5 sec	0.004	0.004	0.006	0.004	0.010	0.005	0.003
% del > 2 sec	0.056	0.060	0.098	0.067	0.114	0.081	0.071
% del > 1 sec	0.116	0.116	0.198	0.129	0.209	0.171	0.154
Avg Trans/node	129.9	108.7	223.7	128.3	184.5	195.4	191.4

Workload = 6/60

Metric	Prbk	Pron	Limt	Qcnt-1	Qcnt-2	Util-1	Util-1
End Time (Secs)	3,838.013	3,882.102	3,748.305	3,860.758	3,825.951	3,759.711	3,748.825
Pkts sent(kpkt)	20,561.936	20,799.024	20,094.512	20,714.176	20,504.608	20,160.944	20,107.712
Pkt Thru-put	5,357.443	5,357.671	5,360.950	5,365.313	5,359.349	5,362.365	5,363.899
Link Thru-put	1,347.	1,275.	1,611.	1,295.	1,483.	1,658.	1,669.
Avg Pkt del	0.799	0.711	1.131	0.775	1.073	1.467	1.375
Blk fact	0.162	0.166	0.205	0.203	0.271	0.154	0.149
Departs/sec	9.899	9.901	9.912	9.914	9.904	9.913	9.918
Departs (%)	0.950	0.961	0.929	0.957	0.947	0.932	0.930
% del > 5 sec	0.019	0.015	0.025	0.018	0.031	0.045	0.039
% del > 2 sec	0.145	0.132	0.220	0.144	0.204	0.289	0.274
% del > 1 sec	0.243	0.216	0.365	0.237	0.321	0.438	0.420
Avg Trans/node	201.5	157.5	285.7	173.3	211.6	273.9	282.9

APPENDIX D

MINIMUM AVERAGE PACKET DELAY ANALYSIS

Packet delay takes two forms, queuing delay and processing delay. The former is variable and, as noted in the main text of this document, is heavily dependent on network conditions and routing options selected. The latter is a function of the path selected for a given transaction and reflects the time necessary to process a packet at each "step" along that path. The minimum average packet delay can be determined by eliminating the first component and concentrating on the second under steady state conditions.

The simulation model assumes that Class II service requests occur at all nodes with the same arrival rate and distribution. While the arrival rate is a user-specified parameter, the distribution is assumed to be Poisson. The model further assumes that all nodes are equally likely to be the destination for an arriving Class II transaction. These assumptions imply that at steady state, the expected number of service requests from node a to node b will be k, some value which is constant for all node pairs (a,b) and is proportional to the arrival rate.

Assuming a steady state condition also allows conclusions to be reached regarding the length of Class II transactions. At steady state, the average length of these transactions, i.e. the average number of packets, will stabilize to a second constant, 1. Finally, the fact that "end-to-end" routing was selected for this experiment implies that all packets of a given message follow the same route and experience the same packet delay.

Taken together, the above arguments indicate that the number of packets traversing the experimental network between any pair of nodes, a and b, is some constant n where n=y*1. Further, each of the packets traversing a particular path experience the same delay.

Reducing these constants, it can be concluded that the average packet delay for the experimental network is equal to the average delay which can be expected for a single packet traversing the network from any node a to any node b. Further, this value is equal to the number of links the packet can expect to traverse multiplied by the minimum time to traverse a link. A user-specified parameter assumed to be 50 milliseconds for this experiment, the link traversal time includes both the signal propogation time and the time to process the packet at the originating node.

Inspection of Figure 3-3 indicates that the minimum number of links which must be traversed in travelling between any nodal pair is as indicated below.

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	2	3	3	1	4	3	1
2	1	0	1	2	3	2	1	3	2	2
3	2	1	0	2	3	1	2	2	3	1
4	2	2	2	0	3	1	1	2	2	3
5	3	3	3	3	0	2	2	1	1	4
6	3	2	1	1	2	0	2	1	3	2
7	1	1	2	1	2	2	0	3	1	2
8	4	3	2	2	1	1	3	0	2	3
9	2	2	3	2	1	3	1	2	0	3
10	1	2	1	3	4	2	2	3	3	0

Review of this table shows that there are the longest individual path is of length 4 with the following table showing the frequency with which each path length occurs.

Length	Frequency
1	26
2	36
3	24
4	4

The weighted average of these values is 2.066667, which is the number of links a packet travelling from any node a to any other node b can expect to traverse. This value can be converted to an average packet delay

by multiplying it times the assumed link traversal time of 50 milliseconds, yielding a minimum average packet delay of 0.1033333 seconds or approximately 0.103 seconds.