UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSC/EE/80D-9

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

SIMULATION AND ANALYSIS OF AUTODIN II NETWORK DESIGN

THESIS

James W. Healy
Denzel H. Henderson
GCS-80D

(14) AFIT/GCS/EE/80D-9

(6) SIMULATION AND ANALYSIS OF AUTODIN II NETWORK DESIGN.

(9) Master's thesis,

by

(10) James W. Healy
Denzel H. Henderson

(11) Dec 84

(12) 274

Thesis

Prepared in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Graduate Computer Systems
December, 1980

School of Engineering
Air Force Institute of Technology
Wright-Patterson Air Force Base,
Ohio

Preface

The need for a comprehensive computer communication network has been evolving over the past years due to the increase in the number of information systems in the Department of Defense (DOD). There is also economical considerations of establishing a single large reliable and comprehensive communications network as opposed to numerous small to medium communications networks.

The AUTODIN II computer communication network is designed to meet these needs. However, communications requirements change and, as with any large project during design and development phases, questions are raised as to the operational capabilities of the system. Many such questions may be answered through simulating the network in its operational environment. This thesis effort developed a computer simulation program of the proposed AUTODIN II network (or any variations of it) in an environment that can be changed either permanently or dynamically.

We had hoped to have operational data from an initial three node AUTODIN II network to validate the simulation, but the data was not available before this thesis effort was completed. Also, we were desirous to test hypothesis concerning other AUTODIN II configurations and ascertain the "weakest links", if any, of the present proposed network, but sufficient time was not available. Others are invited to carry out these tasks.

We would like to express our thanks to two AFIT faculty members, Dr. Thomas Hartrum and Maj. Alan Ross, for their interest, encouragement and friendship during this thesis effort. Our loving appreciation is expressed to our wives and families whose love and understanding endured through this graduate program.

Contents

List of Figures

List of Tables

Abstract

The AUTODIN II computer communications network is expected to provide for present and future Department of Defense communications requirements. The AUTODIN II requirements approach for development is to start small, meet the known validated requirements, and grow as requirements increase and the needs for service change. The background and present and future development of AUTODIN II network is discussed.

The proposed AUTODIN II network configuration is designed to provide reliable communications and minimal time delay for both interactive and timesharing and transaction-oriented systems. As development towards an operational network proceeds, questions concerning efficiency and optimality of the network are asked by both the design managers and future AUTODIN II users. To answer these questions, and other future questions concerning the operational AUTODIN II network, a simulation program at the switch control module (SCM) hierarchal level was developed.

The simulation program allows the user to:

(1) Change the configuration network connectivity and add or delete nodes, SCMs, links and lines;

(2) Generate various communication traffic levels from any source location and to any destination location;

(3) Define network parameters, such as priorities, percentages of packet sizes, transmission times and special workload identification.

(4) Perform any of the previous options dynamically to analyze the effects of network configuration failures or catastrophic events.

A User's Manual is provided as part of the thesis and may be used as a separate document. It describes the input requirements and user options, the output data files and their format, and a data analysis methodology.

Analyses are performed for two Air Force organizations, each with opposite views or concerns regarding the AUTODIN II network; one from the design managers view of the overall network performance measures and specifications, the other from the user's view of the specific netowrk performance measures for a particular type of workload and priority.

CHAPTER 1

INTRODUCTION

Background

In recent years the number of computer information systems with geographically dispersed terminals has experienced continuous growth. These information systems are transaction oriented and require online communications for interactive timesharing, remote job entry and bulk data transfer. Each information system's transaction application has usually required the establishment of its own dedicated communications network with special leased lines.

In 1972, the assistant Secretary of Defense for Communications, Command, Control and Intelligence tasked Defense Communications Agency (DCA) to make recommendations for a common Defense Communication System (DCS) to fulfill computer communications requirements for the Department of Defense (DOD). DCA was also tasked to prepare a plan to fulfill the communications requirements for the World Wide Military Command and Control System (WWMCCS).

DCA had already initiated a comprehensive study to determine the present and future data communications requirements in the military environment. From this study projections into the mid-1980's indicated that within DOD there would be approximately 2500 computers involved with data communications with 20,000 input/output terminals. The study recommended an AUTODIN II plan to be written which called for 47 worldwide data communication switches. This recommendation was later revised, and the current philosophy is to start small, meet the known validated requirements, and then grow and evolve as requirements and the needs of the DOD community increase or change. In the technology area, the DCA study concluded that packet switching would be the best approach to use in AUTODIN II. The AUTODIN II Request for Proposal was released in 1975 and the contract for development and implementation was awarded to Western Union Telegraph Company in 1976 (Ref 1).

AUTODIN II has similar capabilities and performance characteristics to the ARPANET developed by the Defense Research Projects Agency. Subscriber computers and terminals are connected to the nearest AUTODIN II packet switching node (PSN). At the initial PSN, a subscriber message is converted to one or more AUTODIN II standardized messages or packets containing the destination address, security, precedence, community of interest information and other pertinent information. Each packet is dynami-

cally routed along one of several paths to the destination PSN, where the packet is forwarded to the appropriate destination computer or terminal. The delay time between message entry and delivery is so small, that each subscriber's computer or terminal will appear as a dedicated connection.

The path for a packet to take between the source and destination node is determined by a routing algorithm developed initially by Systems Control Incorporation (Ref 2). The algorithm is based on the ARPANET routing methodology but with major changes to implement the requirements of AUTODIN II. Like ARPANET, a distributed routing algorithm, as opposed to a centralized algorithm, was selected to increase survivability of the network. The routing algorithm for AUTODIN II provides more network information to each node in order to make nodes more responsive to network changes and to prevent looping of packets. The routing algorithm is of primary importance in the consideration of the AUTODIN II network configuration and a significant portion of this thesis effort.

A significant difference exists between the AUTODIN II network and previous computer networks. Previous computer networks have been constructed around a single computer per switching node. The AUTODIN II Packet Switching Node consists of one or more computers, called Switching Control Modules (SCM). Each SCM can act independently or with other SCMs. This concept allows expansion of a PSN (by adding SCMs) to meet permanent expansion of traffic workload.

Statement of the Problem

The DCA has not developed a satisfactory inhouse analysis of the AUTODIN II network to test and verify the proposed configuration and software. Systems Control, Inc. (SCI) simulated the PSN level of the network in developing the routing algorithm (Ref 2:App C). Rockwell International Co. (RIC) developed a simulation of a single PSN with one, two and three processors (Ref 3). Capt. J. Whittenton developed a simulation of the PSN level with the proposed network configuration (Ref 4). However, the AUTODIN II algorithm must be simulated for various network configuration and tested under potential stress conditions, including projected workload changes, network failures and various traffic levels and peaks. Knowing the results of the stress conditions for various network configurations will allow for corrective action to be taken before the complete network is developed. Thus, a significant dollar savings could be realized.

Scope

The primary objectives of this thesis are:

1. To model the AUTODIN II backbone network, that is, from the time a message enters a PSN to the time it leaves a PSN.
2. To analyse the network behavior under projected operational and stress conditions.

Objective (1) will allow the user to simulate AUTODIN II,

varying numerous parameters, such as traffic levels, traffic fluctations, flow densities, workload systems and levels of priorities. For each set of parameters the user will be able to change the proposed network configuration to test another configuration. Verification of the model is through comparison with Capt. J. Whittenton's thesis results (Ref 4). The desired result of the simulation is to provide future users with sufficient flexibility to analyze the effects of changing network configurations and workloads.

The authors separately evaluated potential network problems from two opposing views: the designer and the customer. The evaluations were accomplished using the simulation model and the data available at the present time. The authors hoped to use actual data from the initial 3-node network scheduled for operation in June, 1980, but the schedules were slipped past this thesis effort.

The efforts to simulate all details of the AUTODIN II network are beyond the scope of this thesis effort. Therefore, only the backbone of the network is simulated; that is, subscriber computers, terminals and associated access components are not included in the model. The routing algorithm code was classified at the time of this thesis effort and was not available to the authors. Therefore, some differences may exist in the simulated algorithm and the actual algorithm. However, after discussions with DCA personnel, these differences should be very minor.

Approach

This thesis study was initiated through the interests of two sponsors, Capt. W. Nielson, DCA Engineering Center, and Major B. Gilbert, of AF/LEXY. Two AFIT students, the authors of this thesis, Mr. J. Healey and Mr. D. Henderson, were assigned respectively to these sponsors. Since each student required a model of the AUTODIN II network, a joint effort was established to produce a simulation program.

Early in the thesis effort, it was decided not only to attempt to produce a simulation model which could be used to satisfy the requirements of the sponsors, but also the requirements of other Air Force and contractor personnel who may be interested in analyzing the effects resulting from changes in the AUTODIN II network workload and/or configuration. A users manual of the simulation model was written for this purpose and is included as part of the thesis.

The first step in the thesis project was to research and study literature written on the AUTODIN II network. With a background into the present state of the art of the network and with the guidance of the AFIT faculty, the QGERT model was selected. Before selection of QGERT, a simplified version of the AUTODIN II configuration and routing algorithm was tested. Upon satisfactory results from this test, the authors proceeded to model the AUTODIN II network.

This simulation model differs from previous simulations in overall concept and fills a gap left by them. The Rock-

well International Co. simulation concentrated on simulating the interactions occurring within a single PSN and the simulations by Capt. J. Whittenton and by Systems Control, Inc., concentrated on interactions between PSNs within the complete network. This simulation was developed to portray interactions between SCMs within the complete network and to allow SCMs to be added or removed from a PSN.

Upon successful completion of the model, the authors simulated the proposed network and evaluated the potential problems provided by their respective sponsors.

Sequence of Presentation

This report contains eight chapters. Chapter Two describes the AUTODIN II network topology and architecture. Included in this description are some of the requirements and specifications of the AUTODIN II system. Chapter Three describes the AUTODIN II routing algorithm, it's functions and it's six modules. Chapter Four describes the simulation model of the AUTODIN II network and routing algorithm. Also included is the simulation input data in detail for future users of the model. Chapter Five contains the author's methods of analyzing the simulation model results. Chapter Six describes the results of verification of the model with Capt. J. Whittenton's simulation results. Chapters Seven and Eight discuss the results of the simulation model from two viewpoints: Chapter Seven contains the results of

the analysis by Mr. D. Henderson for Major B. Gilbert, AF/LEXY and Chapter Eight contains the results of the analysis by Mr. J. Healey for his sponsor, Capt. J. Nielson, DCA Engineering Center. Appendix 1 contains the User's manual for the AUTODIN II simulation model.

CHAPTER 2

THE AUTODIN II NETWORK

Autodin II Description

AUTODIN II is designed to provide economical and reliable data communications service for both interactive time-sharing and transaction-oriented systems requiring rapid response between terminals and computer-to-computer data transfers requiring high transmission capacity (Ref 1 : ES-2).

Subscriber terminals and computers (hosts) will be connected to the nearest PSN. Subscriber terminals are character oriented input/batch devices such as teletypewriters, CRT terminals and remote batch terminals and minicomputers that serve as communication handlers (Ref 5 :v-1-55). Subscriber hosts are larger computers capable of producing large quantities of data. The subscriber data (or messages) are separated into discrete elements called packets, each packet containing the address of its destination, security, precedence and community of interest information (Ref 1:ES-3). Community of interest information specifies which users are allowed access to certain information or messages. Packets will be dynamically routed to their destination

along one of several paths. Each packet is routed independently so that if the subscriber's data requires more than one packet, each packet may take a separate route. The appropriate route is determined by the AUTODIN II routing algorithm discussed in Chapter Three. A subscriber computer will be capable of maintaining multiple virtual connections with other terminals or computers.

There are currently 47 ADP systems identified as potential candidates to connect to AUTODIN II network. These 47 ADP systems consist of 161 host computers and 1324 connected terminals (Ref 1:1). A number of these connected terminals are concentrator devices that, as a total, connect an additional 1700 devices to the AUTODIN II network.

The primary requirements and specifications for implementing the AUTODIN II network are (Ref 1:8):

1. Initial network configuration of four nodes, network control center and system test facility with capacity for termination of 200 access lines.
2. Capability to grow modularly to eight or more nodes with approximately 1700 subscribers.
3. Node capacities:
    1. 150 to 200 full duplex line terminations;
    2. Each line can be 110 bps to 56 kbps for subscribers and 9.6 KBPS to 230 KBPS for trunks;
    3. Throughput from 500 KBPS to 2.5 MBPS.
4. Maximum packet length of approximately 5300 bits.
5. Four traffic acceptance categories (CATI, CATII,

CATIII and CATIV).

6. Nonblocking for the highest precedence traffic.

7. Availability of 99 percent for single-homed subscribers and 99.95 percent for dual-homed subscribers.

8. Minimum impact on existing subscriber equipment/ system software.

9. End-to-end undetected bit error rate of less than 1×10^{-12}.

10. Segment misdelivery rate of less than 1×10^{-11}. A subscriber's data is broken into elements (at the CCU for hosts, at the PSN for terminals) called segments. At the initial PSN, segments are converted to packets.

11. 300 MS maximum backbone delay for interactive and highest precedence packets (600 bit packets).

12. Transparent to text

13. Capable of supporting effective transmission rate of at least 75 percent of lowest access line speed on a connection.

14. Top-down, structured design of software using a high order language to its maximum extent.

15. Capable of being certified to handle all levels of classified traffic.

The initial AUTODIN II network will consist of three PSNs located at Fort Dietrick, Frederick, Md.; Tinker AFB, Oklahoma City, Okla.; and McClellan AFB, Sacramento, Cal. After thorough testing, a fourth PSN at Gentile AFS, Dayton, Ohio, will be added. The network will continue to grow to eight PSNs. As requirements increase, additional PSNs may be added to the network. The eight PSNs will be located at the present AUTODIN I CONUS sites. This was recommended

to keep overall costs down.

AUTODIN II Architecture

The AUTODIN II network and routing algorithm are the results of studies performed by DCA, Western Union Telegraph Company (primary contractor), Ford Aerospace and Communications Corporation and Systems Control, Inc. Each packet switching node will be directly connected to at least three other PSNs with backbone trunks (lines) between the PSNs operating at a maximum of 56k BPS as shown in Figure 1. All network elements will consist of common equipment, the Digital Equipment Corporation (DEC) PDP-11 family of computers. This will reduce quantities of spare parts and simplify training.

The basic elements of a PSN are: Switch Control Module (SCM), Terminal Access Controller (TAC), Line Control Module (LCM) and Patch and Test Facility (PTF). (Ref 1:ES-21) The SCM manages the basic packet switching function and interfacing of subscriber hosts and consists of one PDP-11 processor. Each PSN contains one or more SCMs depending on the quantity of throughput required. The TAC performs the function of interfacing the subscriber terminals to the network and also consists of one or more PDP 11-70 processors depending on the quantity of subscribers and the traffic volume. The LCM, attached to both the SCM and TAC units, terminates access lines. The PTF monitors and updates test

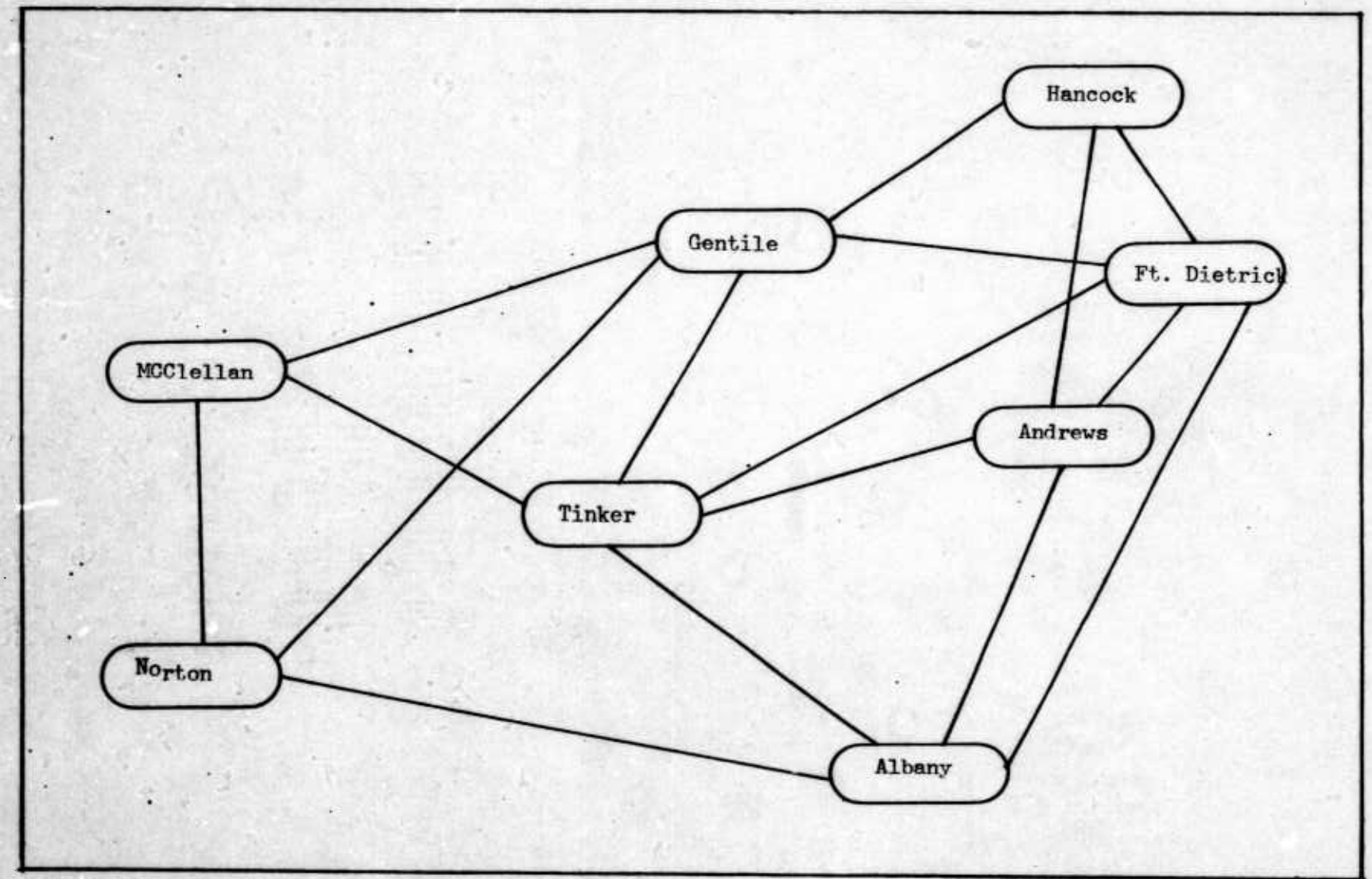


Fig 1. The AUTODIN II Network Connectivity

points to restore or maintain circuit operations. The PTF uses the present AUTODIN I equipment. Each PSN also maintains a standby subsystem (PDP 11-70) to take over in the event one of the processors fails.

One of the design objectives of AUTODIN II was to minimize the subscriber hardware and software modifications when connecting a host to the network. This objective is met through the use of one of two channel control units (CCUs) as shown in Figure 2. The single CCU (SCCU) provides one physical port between the host and the SCCU and one logical connection to the network. The multiple CCU (MCCU) provides up to 32 ports between the hosts and the MCCU and 32 simultaneous logical connections to the network.

Located outside the eight PSN network will be the Network Control Center (NCC). The NCC gathers information on network performance and usage to respond to the management and control requirements of AUTODIN II. The PSN, SCMS, Links and trunks of the proposed AUTODIN II network configuration are shown in Figure 3.

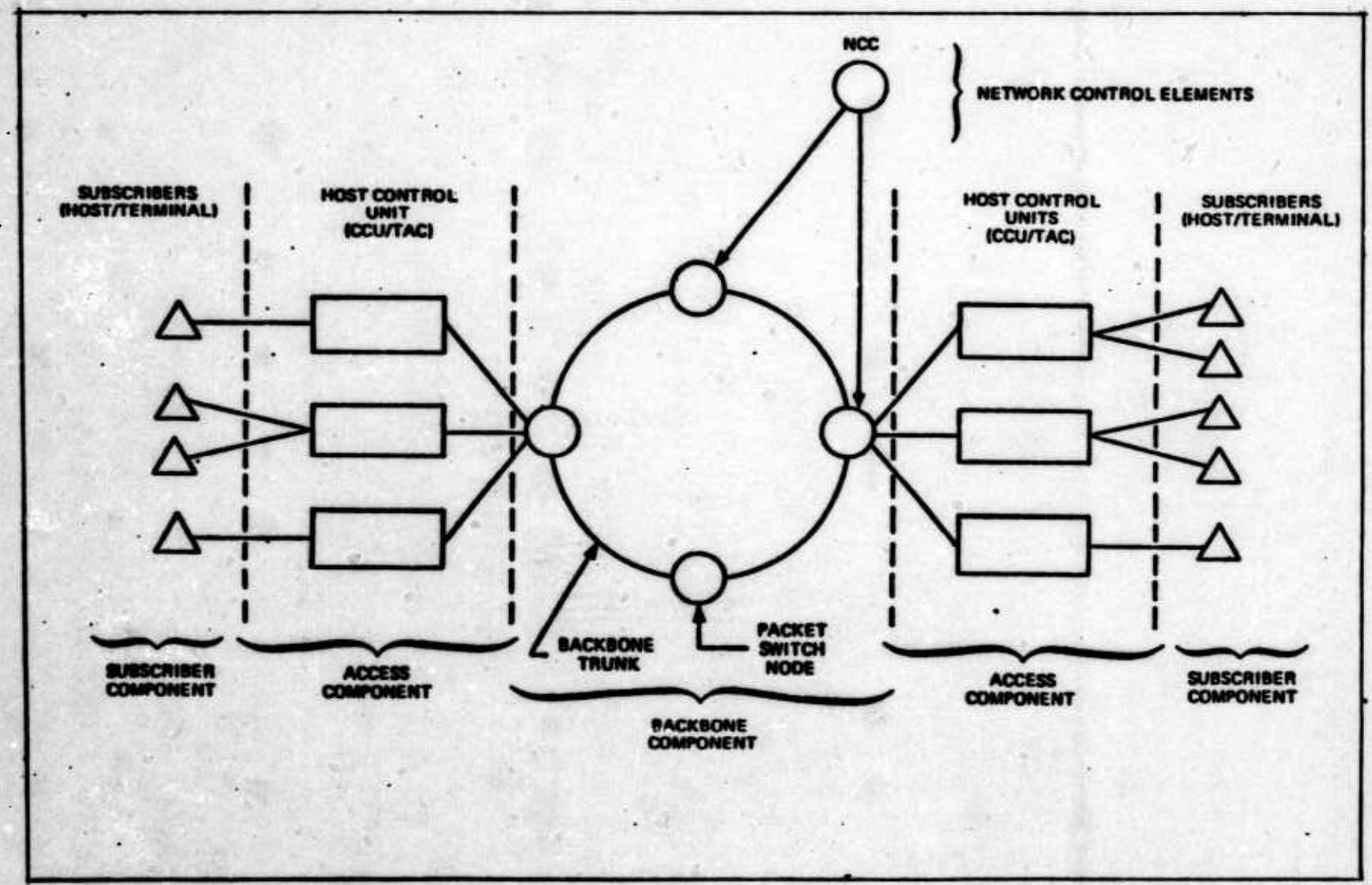


Fig 2. AUTODIN II Control (Ref 2:ES.2-2)

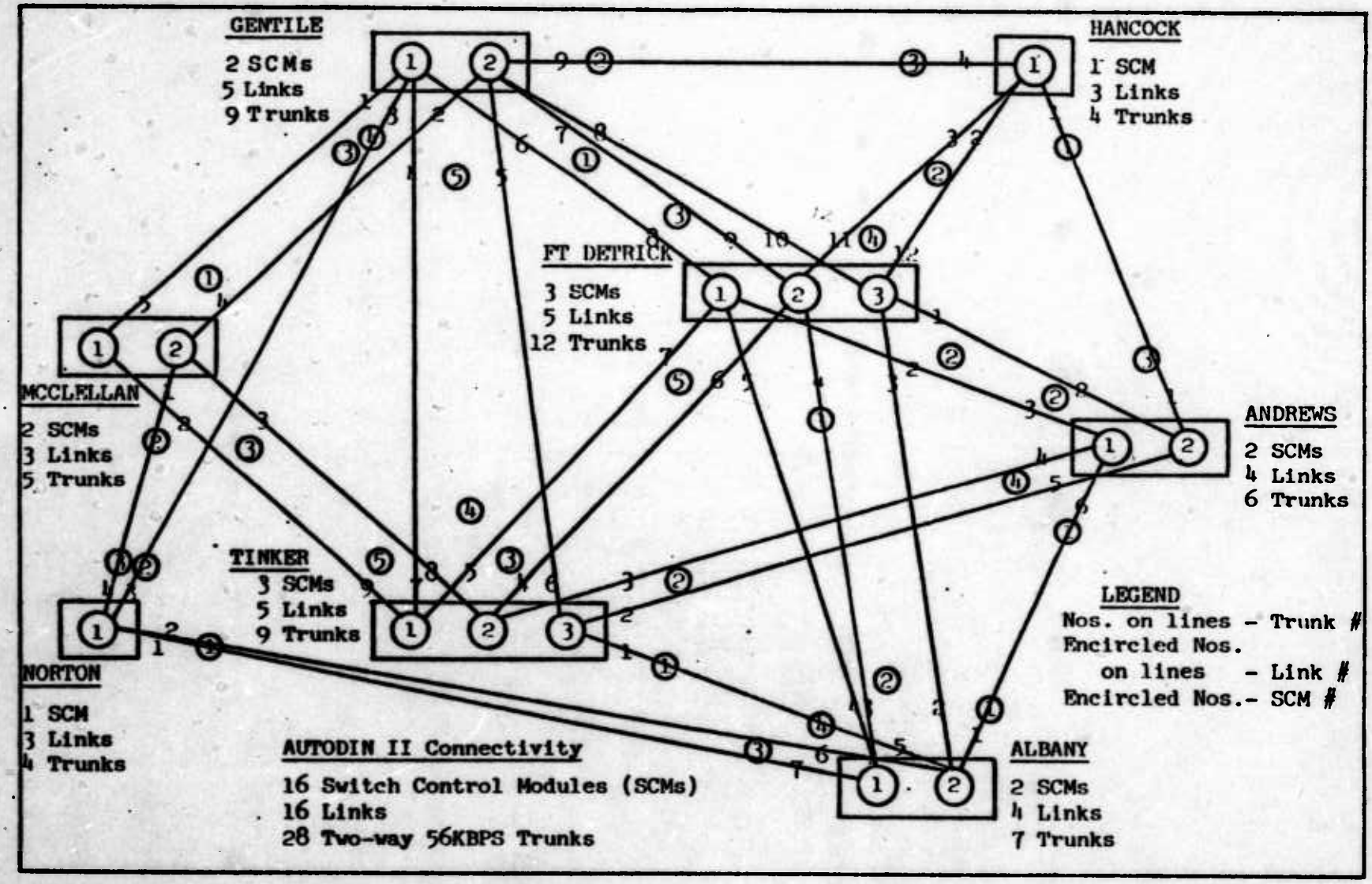


Fig 3. Network Layout of AUTODIN II

CHAPTER 3

THE AUTODIN II ROUTING ALGORITHM

Description

The AUTODIN II network employs a distributed adaptive routing algorithm. It is distributed in that decisions of the paths that packets will take are not controlled by a centrally located processor. It is adaptive because, for example, a system failure or link congestion in one area of the network will effect the decision as to what path packets will take in another part of the network. Adaptive routing is possible through the exchange of information between nodes in the network. The AUTODIN II routing algorithm is a derivative of the ARPANET routing algorithm with significant changes and improvements for application to the AUTODIN II network architecture and topology. In fact, a detailed study of the ARPANET applications and problems was made before the present AUTODIN II routing algorithm was proposed and accepted (Ref 5:V-1-105).

A routing decision is made by accessing routing tables maintained at every SCM. The routing tables contain the most current information on the network status. Thus a com-

plete path from a source SCM to a destination SCM is not determined by the source SCM; each packet is directed along the path the current SCM thinks is best. The operation of the AUTODIN II routing algorithm is summarized as follows

1. Each node maintains a copy of the network topology.

2. Each node monitors its output trunk group queue, and compares them with a predetermined congestion threshold value.

3. Whenever the connectivity of a node alters, or the congestion status of any of its trunk groups alters, or on a periodical basis, the node transmits a connectivity/congestion message to all other nodes.

4. On receipt of a connectivity/congestion message, each node updates its local copy of the network topology and updates its table of minimum-hop paths to all destinations. (Minimum-hop is the least number of hops required for a packet to take to reach its destination PSN from its present PSN.)

5. Tandem nodes (nodes between the source and destinations nodes) route packets by choosing only among the minimum-hop paths.

6. Source nodes route packets by choosing only among the minimum-hop paths and the minimum-hop + 1 paths.

7. When a tandem node or source node has two or more paths to a destination, it chooses a specific output link by minimizing the conbination of local queue delays plus bias representing the delay due to and reported congestion on the paths.

As discussed previously, each node contains one to three SCMs depending on the projected traffic volume at that node. Nodes are interconnected by from one to three logical

links with each link consisting of from one to three physical transmission lines (trunks). Each SCM processor operates with a copy of the routing algorithm and is therefore responsible for routing packets on the paths to the next PSN (SCM). A distributed routing algorithm may have problems with looping of packets between PSNs or SCMs or delayed information on network status. Looping is possible since each PSN, independently, makes its own decisions, and two PSNs could loop a packet between each other. Therefore, certain precautions have been built into the routing algorithm. Some of these precautions are

1. A field in the packet header is decremented and tested at each PSN to identify packets previously processed by the PSN.

2. No packets are returned to the node from which they have just been received.

3. All routing updates take place concurrently throughout the network and the full delay table is accumulated before the minimum delay route is chosen.

4. The exchange of routing information is planned to occur frequently enough (of the order of 0.5 seconds) so that changes are rapidly transmitted throughout the network.

It is important to note that although each SCM could appear to be considered as an independent PSN, this is actually not the case. A hierarchical approach is taken. The routing algorithm is implemented on an inter-PSN basis with a separate process for intra-node routing between SCMs. For example, if an SCM computes a path for a packet which

requires transmission on the line of another SCM within the same node, the packet is placed in the output queue of that SCM through the processor communications link (PCL). If the link contains more than one line to the next node the line with the minimum output queue is selected.

Design

An explanation of the routing algorithm tables, arrays and modules will now be presented. The routing algorithm data base is a table of the status of the network called the link distance (LD) table. The LD table is maintained by periodic internodal status packets. The LD table is a three state value defined from node i to node j;

$$LD(i,j) = \begin{cases} 1.00 & \text{if i and j are connected, but not congested.} \\ 1.01 & \text{if i and j are connected and congested.} \\ & \text{if i and j are not connected or i = j.} \end{cases}$$

An example of the LD table for the Andrews node is given in Figure 4.

If the LD table in a node is changed, an algorithm called MINHOP (to be explained later) is executed to generate, from the LD table, a routing distance (D) table. The D table for node N contains the distance to every other node in the network over each link from node N. The D table contains, for each link of node N, the distance, in number of hops, to each of the other nodes in the network. Only minimum hop

SOURCE NODE (i)		k	DESTINATION NODE (j)							
		j:	ALBANY 1	ANDREWS 2	FT. DETRICK 3	GENTILE 4	HANCOCK 5	MCCLELLAN 6	NORTON 7	TINKER 8
	ALBANY	1	1000	1	1	1000	1000	1000	1	1
	ANDREWS	2	1	1000	1	1000	1	1000	1000	1
	FT. DETRICK	3	1	1	1000	1	1	1000	1000	1
	GENTILE	4	1000	1000	1	1000	1	1	1	1
	HANCOCK	5	1000	1	1	1	1000	1000	1000	1000
	MCCLELLAN	6	1	1000	1000	1	1000	1000	1	1
	NORTON	7	1	1000	1000	1	1000	1	1000	1000
	TINKER	8	1	1	1	1	1000	1	1000	1000

Fig 4. Initial LD Table for the Andrews Node

and minimum-hop plus 1 paths have non-infinite entries. With the minimum hop and minimum-hop plus 1 constraint it can be proven that looping cannot occur in the network (Ref 2:A-2). An example of the D table for the Andrews node is given in Figure 5.

The routing algorithm is made up of 6 modules: PAKROUTE, MINHOP, RTABLE, TTABLE, UPDATE and MESGEN. The overall organization of the routing modules is shown in Figure 6. The UPDATE box includes both RTABLE and TTABLE modules. The purpose of the authentication block is to ensure that only actual nodes (not users) can send authentic network status packets. Otherwise a user, either maliciously or accidentally, could bring down the network by bringing all links down.

The reader should note from Figure 6, that the routing modules are divided into two functions: Background functions and Per Packet functions. An explanation of each of these functions and each of the routing modules follows. For a more detailed explanation, consult the references in the bibliography.

Background Functions

Background functions are accomplished on a periodic basis to update (if necessary) the routing tables and other associated information as shown in Figure 7. The routing of packets in the network is hierarchical in that first, node to node routing is determined and second, which trunk

D(i, l)

NODE(i)	i	OUTPUT LINK (l) 1	2	3	4
ALBANY	1	1	1000	1000	1000
ANDREWS	2	0	0	0	0
FT. DETRICK	3	1000	1	1000	1000
GENTILE	4	1000	1000	1000	1000
HANCOCK	5	1000	1000	1	1000
MCCLELLAN	6	1000	1000	1000	1000
NORTON	7	1000	1000	1000	1000
TINKER	8	1000	1000	1000	1

INTIAL D TABLE

D(i, l)

NODE(i)	i	OUTPUT LINK (l) 1	2	3	4
ALBANY	1	1	2	1000	2
ANDREWS	2	0	0	0	0
FT. DETRICK	3	2	1	2	2
GENTILE	4	3	2	2	2
HANCOCK	5	1000	2	1	1000
MCCLELLAN	6	3	3	3	2
NORTON	7	2	3	3	3
TINKER	8	2	2	1000	1

REVISED D TABLE

Fig 5. D Tables for Andrews

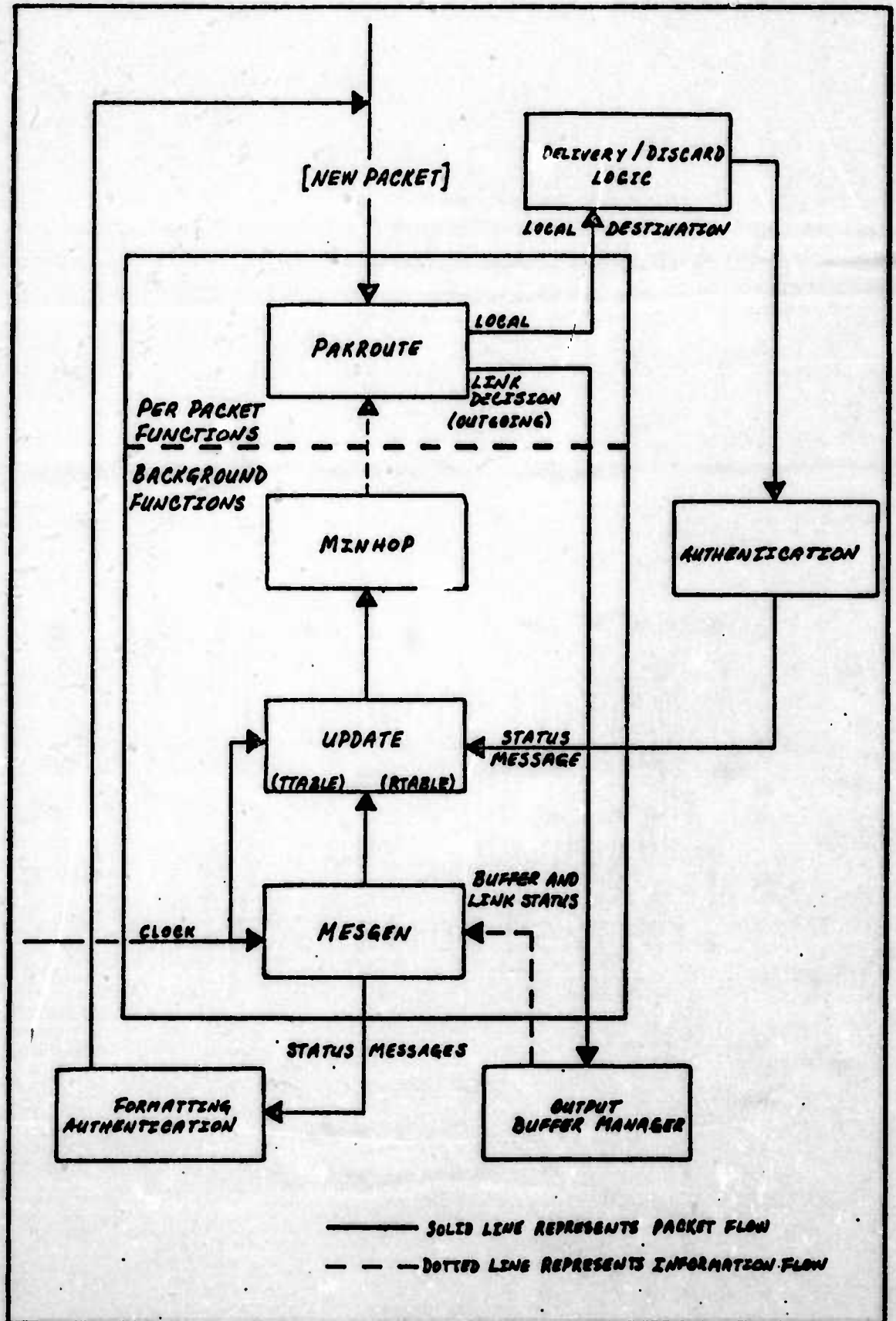


Fig 6. Overview of Routing Module (Ref 2:A-4)

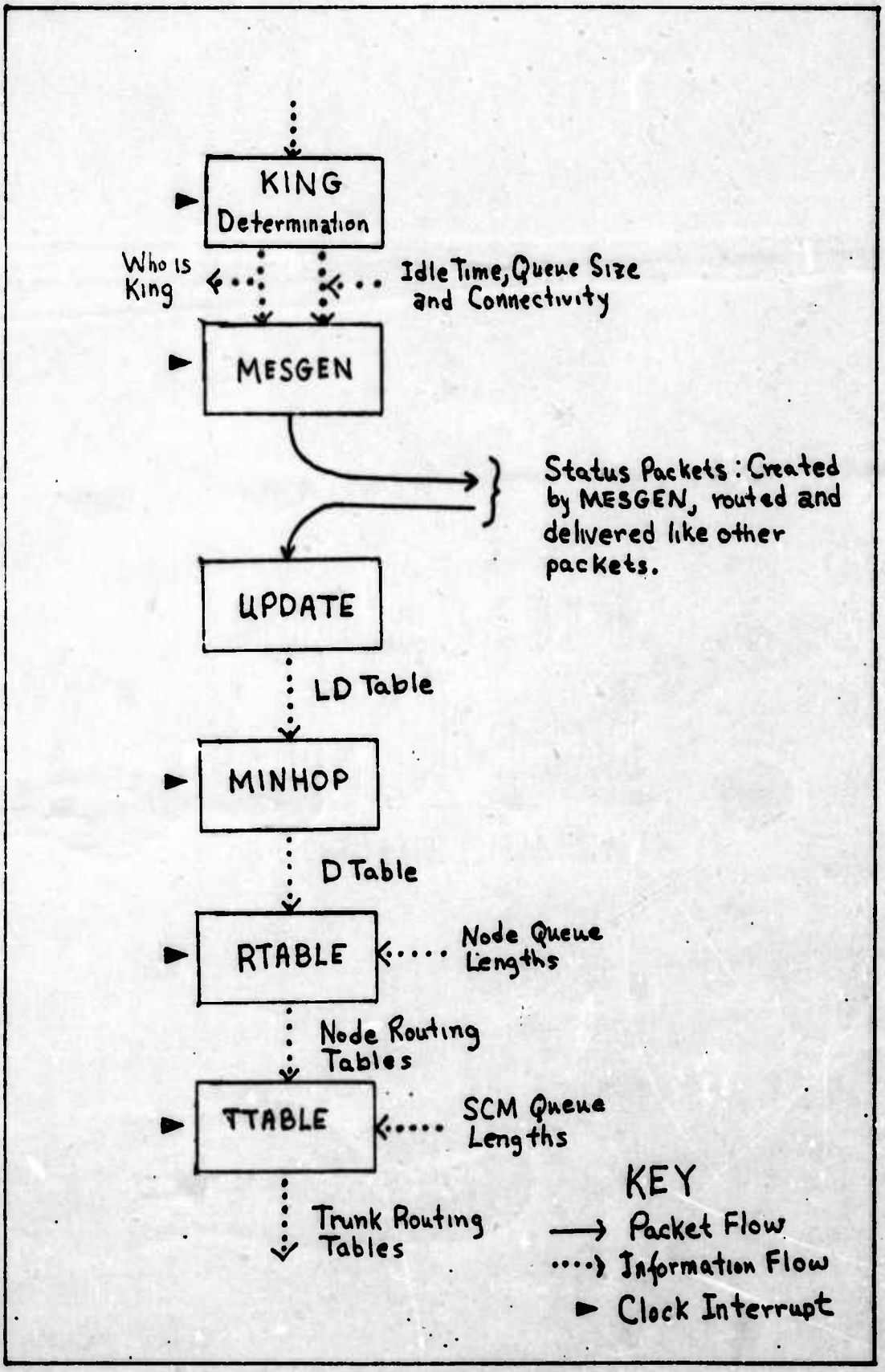


Fig 7. Schematic View of Background Tasks

of a link to use is determined. The background functions, performed by MESGEN, UPDATE, MINHOP, RTABLE and TTABLE modules, develop the node-to-node and trunk routing tables.

The background tasks need not be performed by each SCM in a node, and MESGEN (the module which generates status packets to other nodes) must only be accomplished by one SCM. Otherwise duplication of status packets could flood the network. Thus, one SCM in each node will be designated as "king" and be responsible for implementing the background tasks and notifying the other SCMs within the node of changes in the routing tables. The king will be periodically determined in a distributed fashion as the SCM having minimal load. This will distribute these background tasks to the processor with the smallest current load, Each SCM must determine the same king SCM. In the case of a failure of the king SCM, the next king will take over the background functions (Ref 7:4).

To provide the king with the necessary information, and for each SCM to determine a common king SCM, the SCMs within a node will exchange vectors containing (Ref 7:4).

1. the processor utilization during a time interval and/or other estimates of the current processor load
2. the input queue size for processing at the SCM
3. the output queue sizes for each of the trunks emanating from the SCM with minimal prosessor utilization.

Figure 8 shows the process each SCM goes through to determine the next king and when and how to exchange vectors.

MESGEN

The MESGEN module checks for changes in the connectivity and congestion of the node and, if a change exists, sends status packets containing the node's current status to all other nodes in the network. The flowchart of MESGEN is shown in Figure 9. A link is congested whenever the average number of packets over all trunk output queues in the link is greater than a threshold t1. A node is considered to be saturated whenever the total number of packets in the node input queue exceeds a threshold t2. When a node is saturated, all outgoing links are considered congested. MESGEN is executed periodically at two different time intervals, FTICK and STICK. When executed as a result of FTICK (100 milliseconds), "bad news" is checked, that is, if the congestion or saturation thresholds have been exceeded since the last FTICK or if a trunk, link or SCM have gone down since the last FTICK. When executed as a result of STICK (400 milliseconds), "good news" is checked, that is, if the congestion and saturation levels are within the thresholds since the last FTICK or STICK or if a connectivity within the node which was down is now up. Again, only if a change in the node connectivity or congestion and saturation levels is found at FTICK or STICK intervals will status packets be

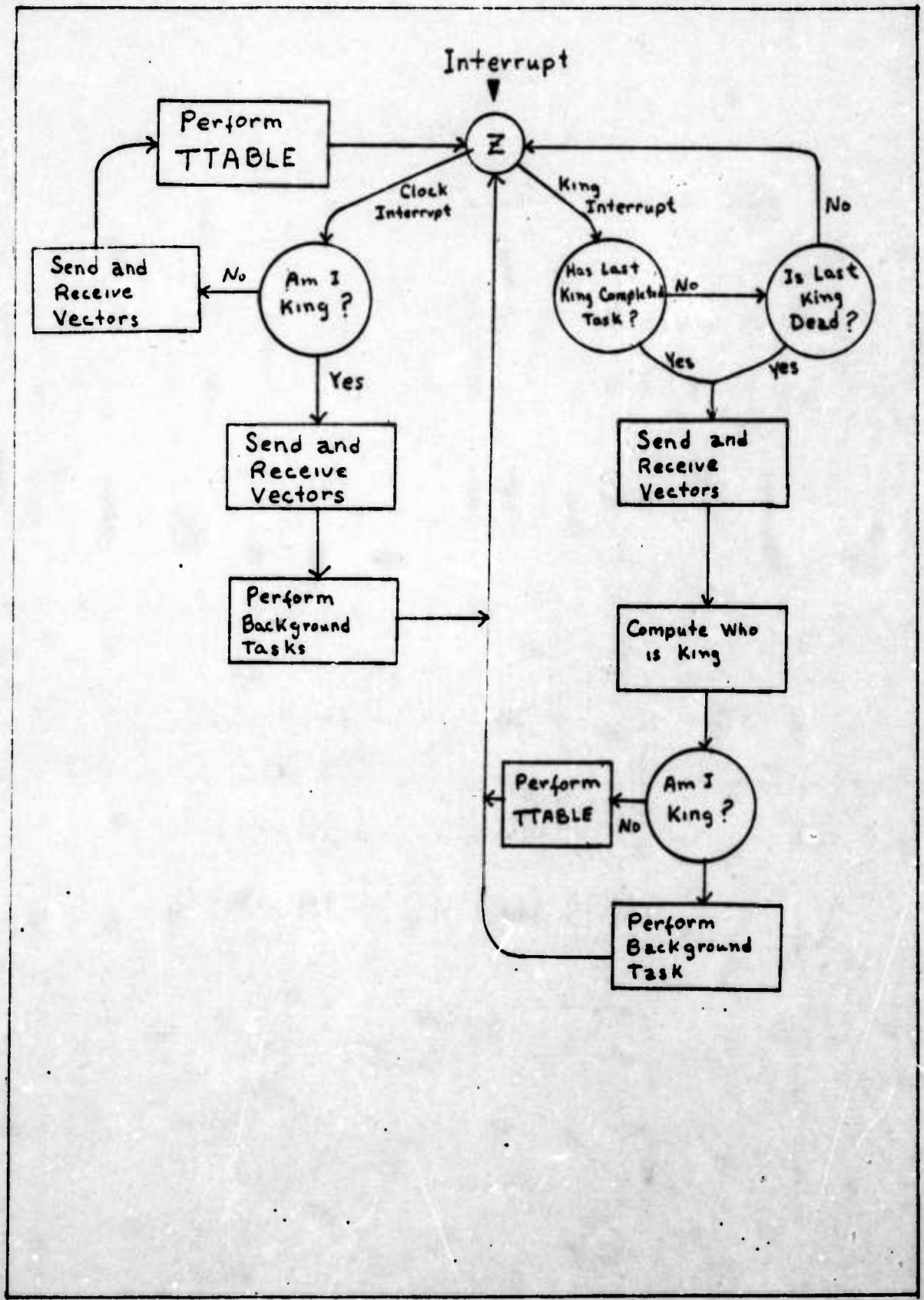


Fig 8. King Process and Background Timing

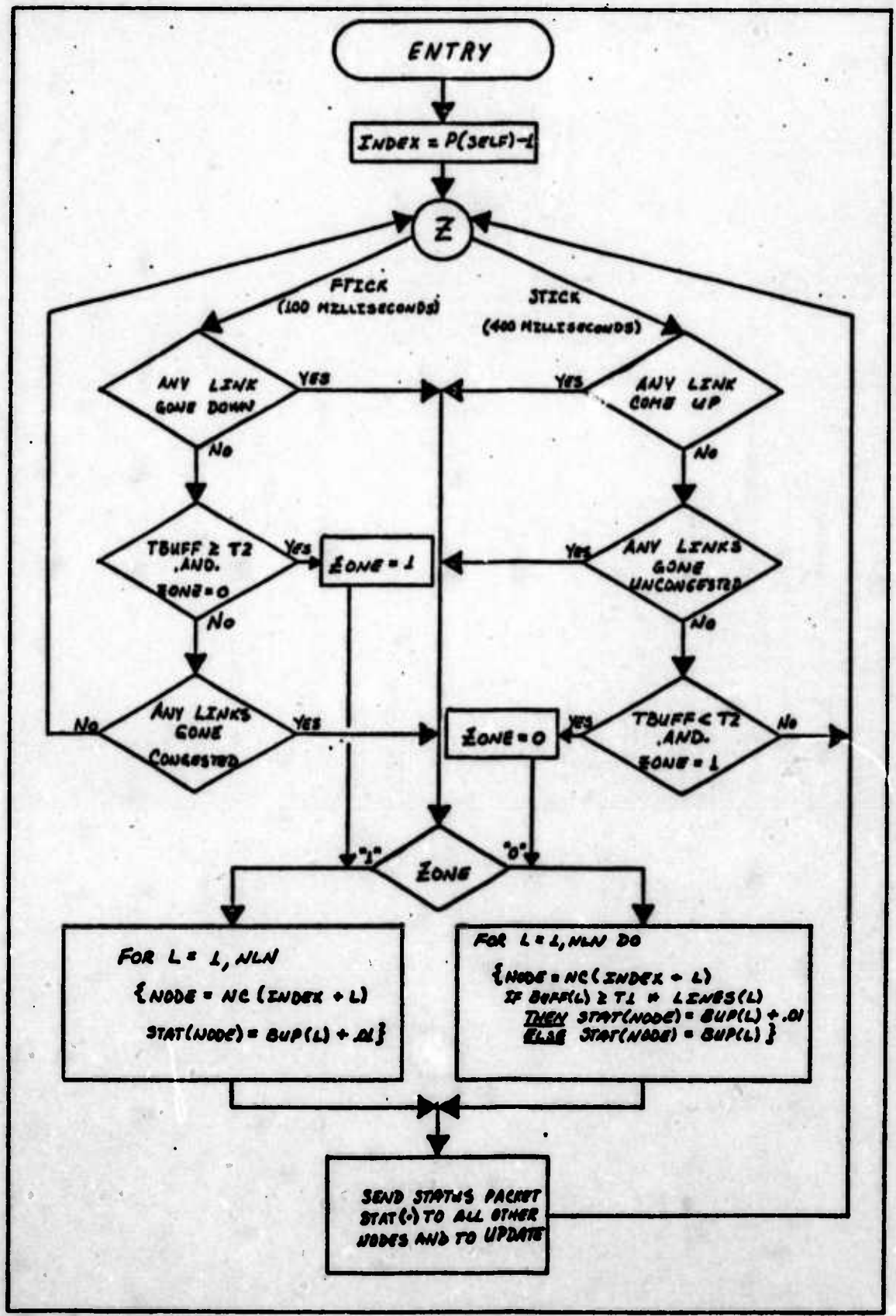

ENTRY
INDEX = P(SELF)-1
Z
FTICK
(100 MILLISECONDS)
STICK
(400 MILLISECONDS)
ANY LINK GONE DOWN
YES
No
ANY LINK COME UP
YES
No
TBUFF ≥ T2 .AND. ZONE=0
Yes
No
ZONE = 1
ANY LINKS GONE UNCONGESTED
YES
No
ANY LINKS GONE CONGESTED
No
YES
ZONE = 0
TBUFF < T2 .AND. ZONE = 1
YES
No
"1"
ZONE
"0"
FOR L = 1, NLN
{NODE = NC (INDEX + L)
STAT(NODE) = BUF(L) + .01}
FOR L = 1, NLN DO
{NODE = NC(INDEX + L)
IF BUFF(L) ≥ T1 # LINES(L)
THEN STAT(NODE) = BUF(L) + .01
ELSE STAT(NODE) = BUF(L)}
SEND STATUS PACKET STAT(·) TO ALL OTHER NODES AND TO UPDATE

Fig 9. MESGEN

generated and sent to other nodes in the network. If a change is found a copy of the status packet will be sent to the UPDATE module for local (nodal) processing.

UPDATE

The UPDATE module processes status packets received from other nodes or from other SCMs within the same node. When a status packet is received from another node, the SCM at which it arrives executes UPDATE and sends copies to the other SCMs in the same node. Each arriving status packet is checked to see if it contains a change from the existing information in the LD table. If a change does exist the LD table is updated and a flag is set to indicate a change has occurred. The MINHOP, RTABLE and TTABLE modules are then executed. Figure 10 describes the procedure of UPDATE.

RTABLE

The RTABLE module uses the information in the distance (D) table and queue length data in the node's links to maintain the two routing arrays, SROUTE and ROUTE. The SROUTE array contains the minimum hop or minimum hop plus one paths to determine the next path for a packet entering at a source node. The ROUTE array contains the minimum hop paths to determine the next path for a packet entering a tandem node. Both arrays are N x 1 arrays, where N is the number of nodes

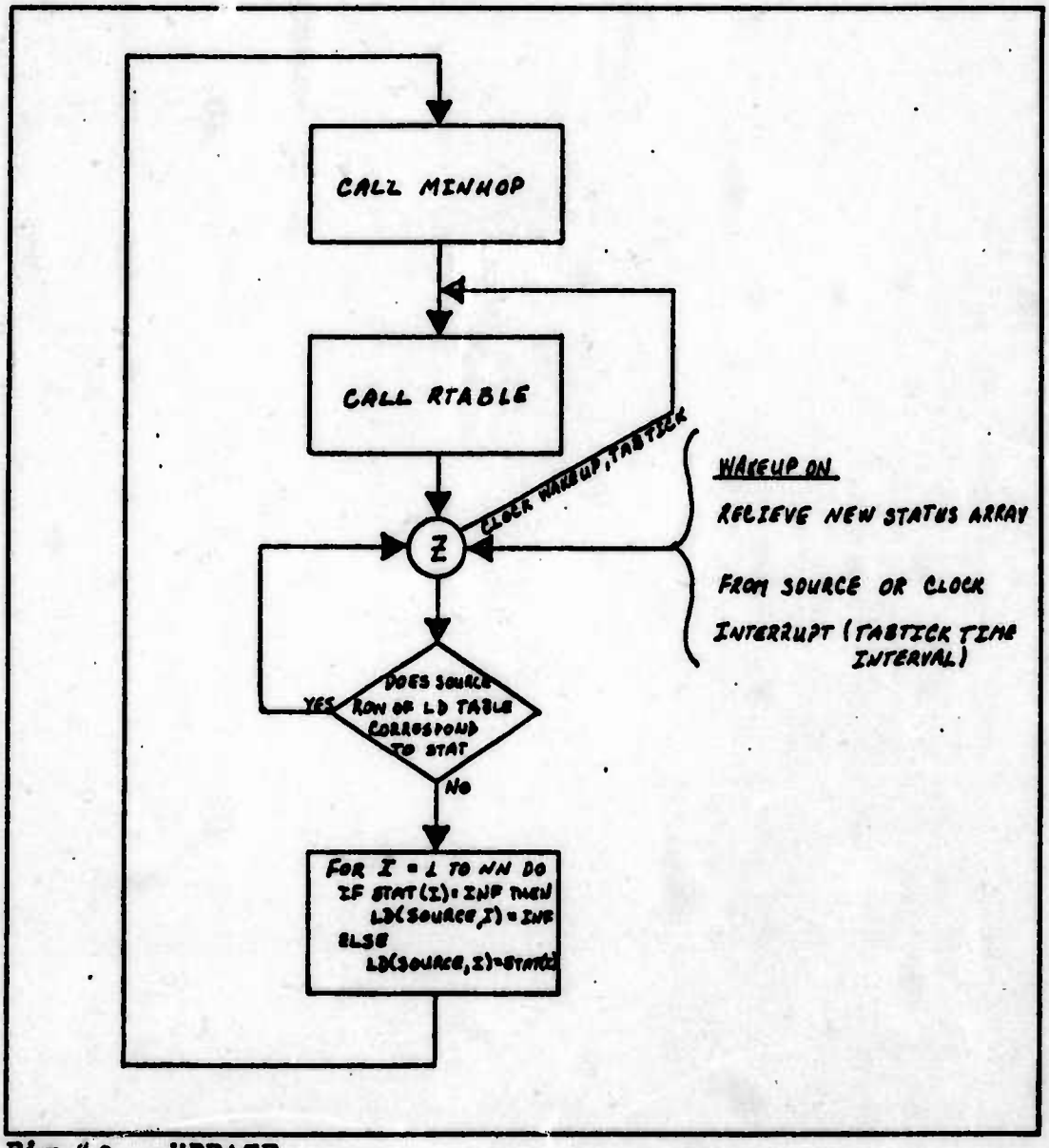


Fig 10. UPDATE

in the network. An entry in either array contains the link over which a packet will be routed next. The rationale behind having two routing arrays is based on the idea that under normal operating conditions a packet will encounter at the most only one tandem node because of the network connectivity. The source node could have multiple "best" routes to select from to get the packet to its destination, but a tandem node, in general, would have only one "best" route which would be the link directly connecting it to the destination node (Ref 4:32). The RTABLE procedure is shown in Figure 11.

TTABLE

TTABLE is the module used to build the trunk routing tables. These tables contain the trunks over which packets are to be routed, given a selected output link from SROUTE or ROUTE. The TTABLE algorithm and its associated array were not specified in the SCI documents. But from a description of the algorithm (Ref 6:38-39), a TTABLE module was developed by the authors to more fully exercise the routing algorithm. The trunk is selected based on the minimum packets in the output queues of all trunks in the selected link.

MINHOP

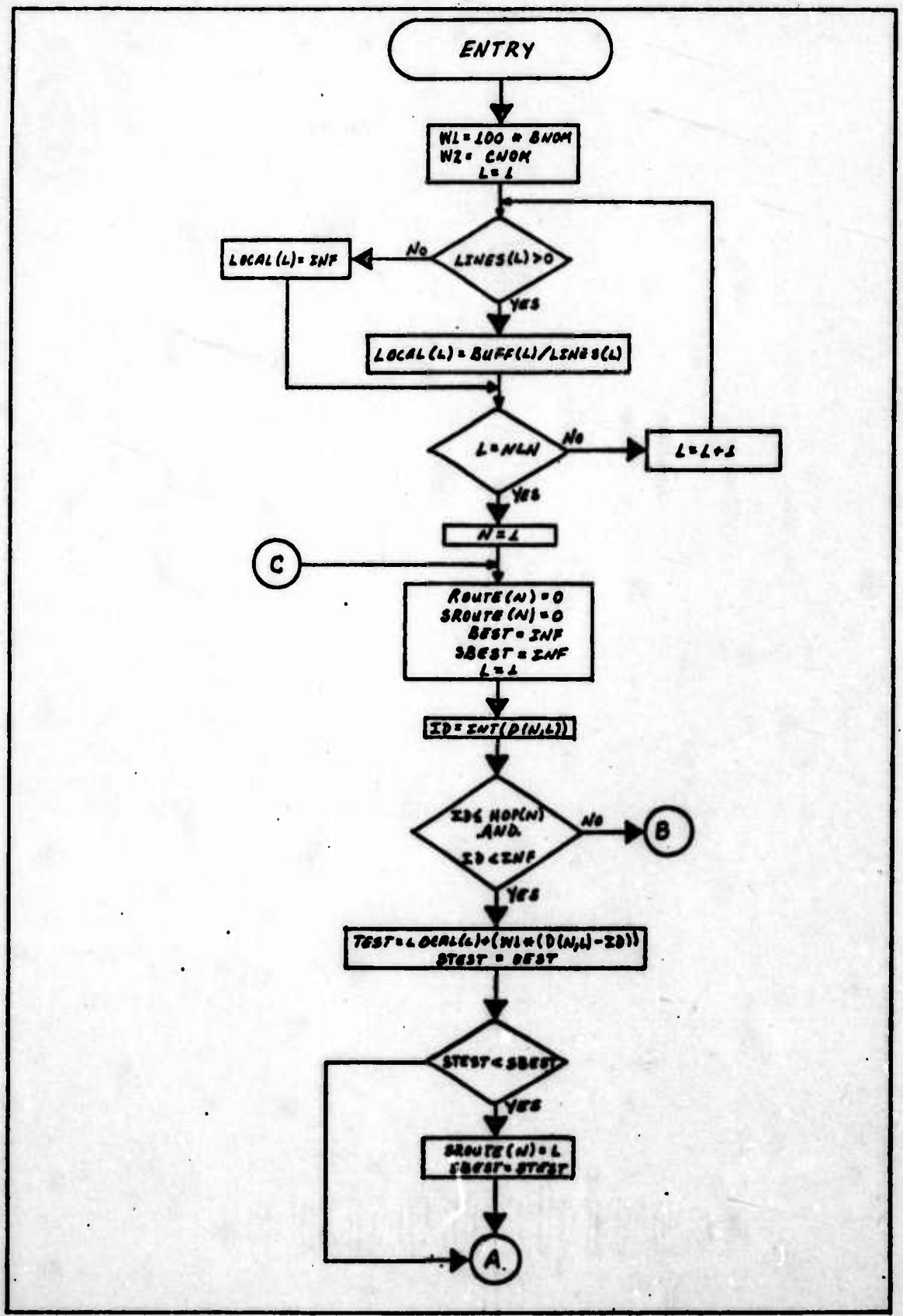


Fig 11. RTABLE

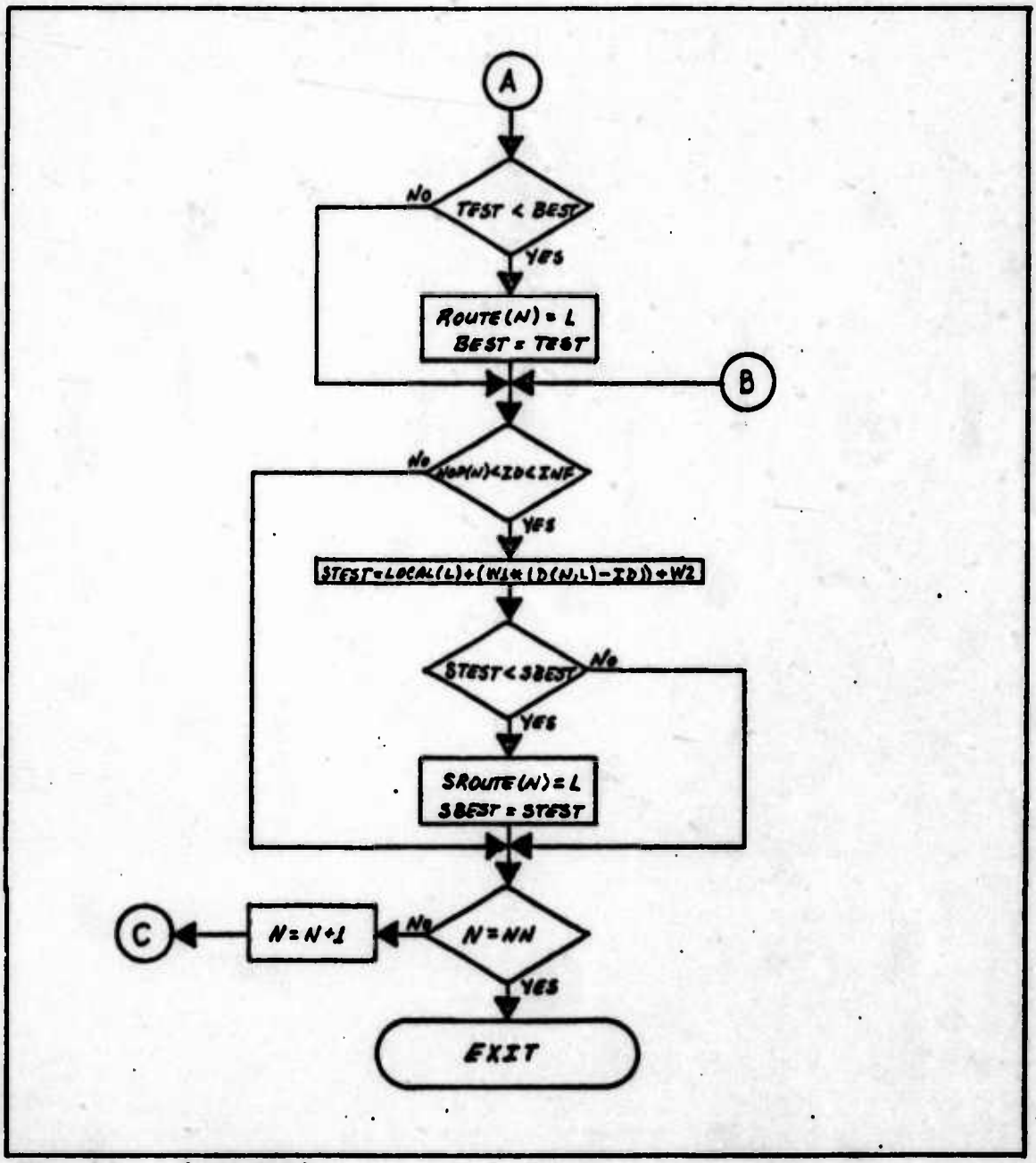

A
TEST < BEST
No
YES
ROUTE(N) = L
BEST = TEST
B
No
YES
STEST=LOCAL(L)+(W1*(D(N,L)-ID))+W2
STEST<SBEST
No
YES
SROUTE(N)=L
SBEST = STEST
C
N=N+1
No
N=NN
YES
EXIT

Fig 11. (CONT'D)

The MINHOP module is the "heart" of the routing algorithm (Ref 2:A-9) . It calculates the entries of the D table which, again, reflects the number of hops (plus a constant of .01, if congestion exists) required to reach another node on a particular link. This table contains only minimum hop and minimum hop plus one values. The entries are infinite if the hop count exceeds the minimum hop plus one. MINHOP consists of five processes: CONNECTIVITY, CONN-CHANGE, INIALIZE-D, COMPUTE-D, and INCREMENTAL-D, briefly described below.

CONNECTIVITY

CONNECTIVITY is used for initial computations of the connectivity pointer (P) array and the connectivity vector (NC) array. An example of the P and C arrays are shown in Figure 12. The P array is an (n+1) x 1 array representing the nodes in the network whose entries point to elements in the NC array. The NC array is an NL x 1 array, where NL is the number of one way links in the network. Each entry in the NC array corresponds to the destination node of each link from an initial node. The P and NC arrays are computed from the LD table to provide equivalent but more convenient informational form (Ref 2:A-9) . CONNECTIVITY is executed only when gross modifications in the network require it, that is, upon adding or permanently deleting nodes or links, or upon initialization (Ref 7:25).

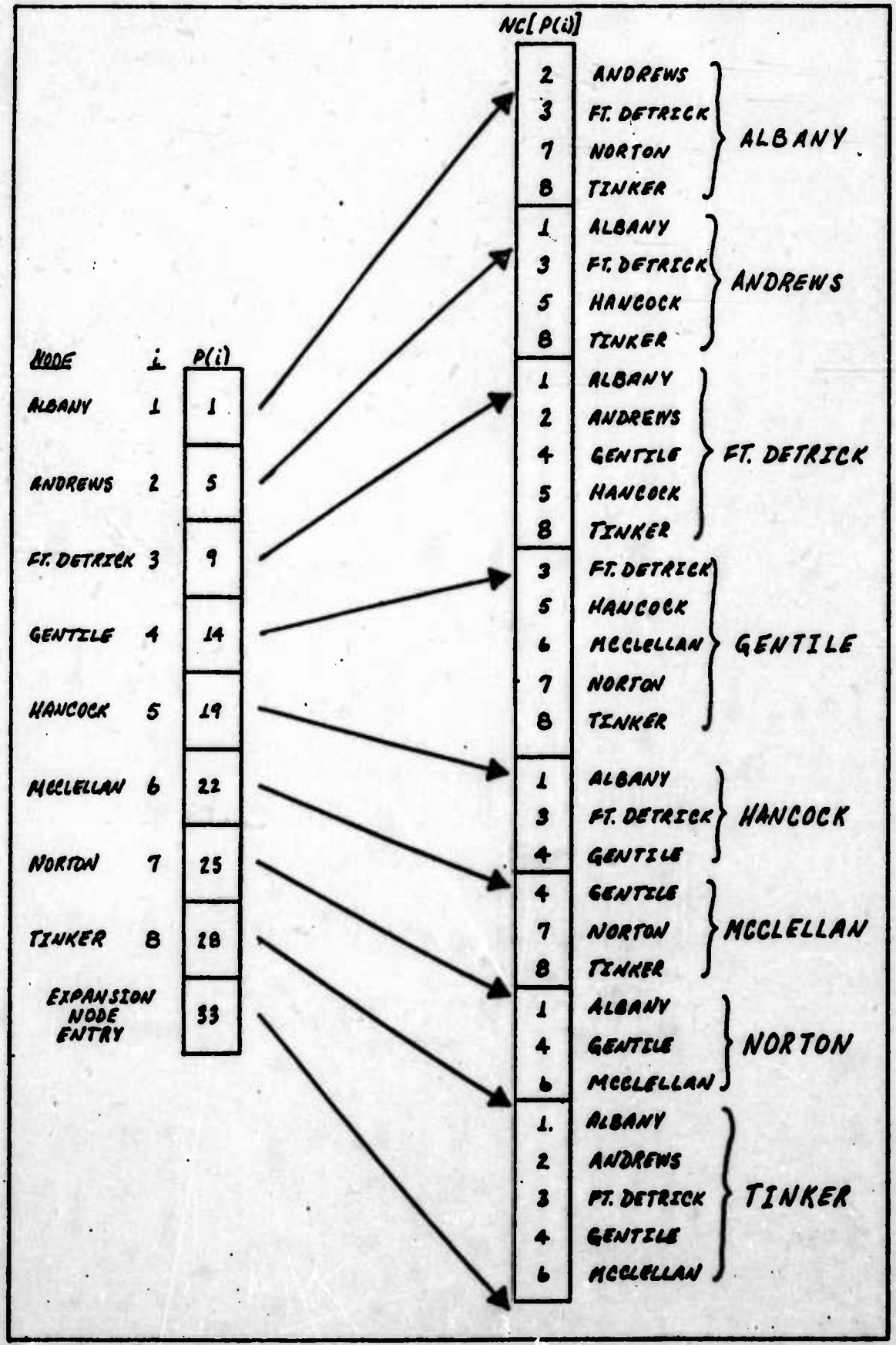


Fig12. P and NC Arrays

CONN-CHANGE

CONN-CHANGE is used to update the NC array and the corresponding entries in the LD table to reflect a link failure or repair or other minor network modifications. This procedure is faster than performing the CONNECTIVITY procedure and will be used to update the NC array most of the time.

INITIALIZE-D

INITIALIZE-D initializes the entries in the D table using entries from the LD table and the P and NC arrays. As with the P and NC arrays in the previous two processes, INITIALIZE-D and COMPUTE-D initialize and maintain two arrays, LIST and HOP, to provide network data in a convenient informational form. The LIST array is an N X 1 array where NLN entries are the nodes connected to the subject node and the other N - NLN - 1 entries are the nodes not connected. The authors could not find any use of this array in the routing modules and found the array dimensioned incorrectly in (Ref 2:A-15). The HOP array is an N X 1 array whose entries contain the minimum number of hops required for a packet to reach a particular node (if the entry is for the subject node the value is zero). An example of the LIST and HOP arrays are given in Figures 13 and 14.

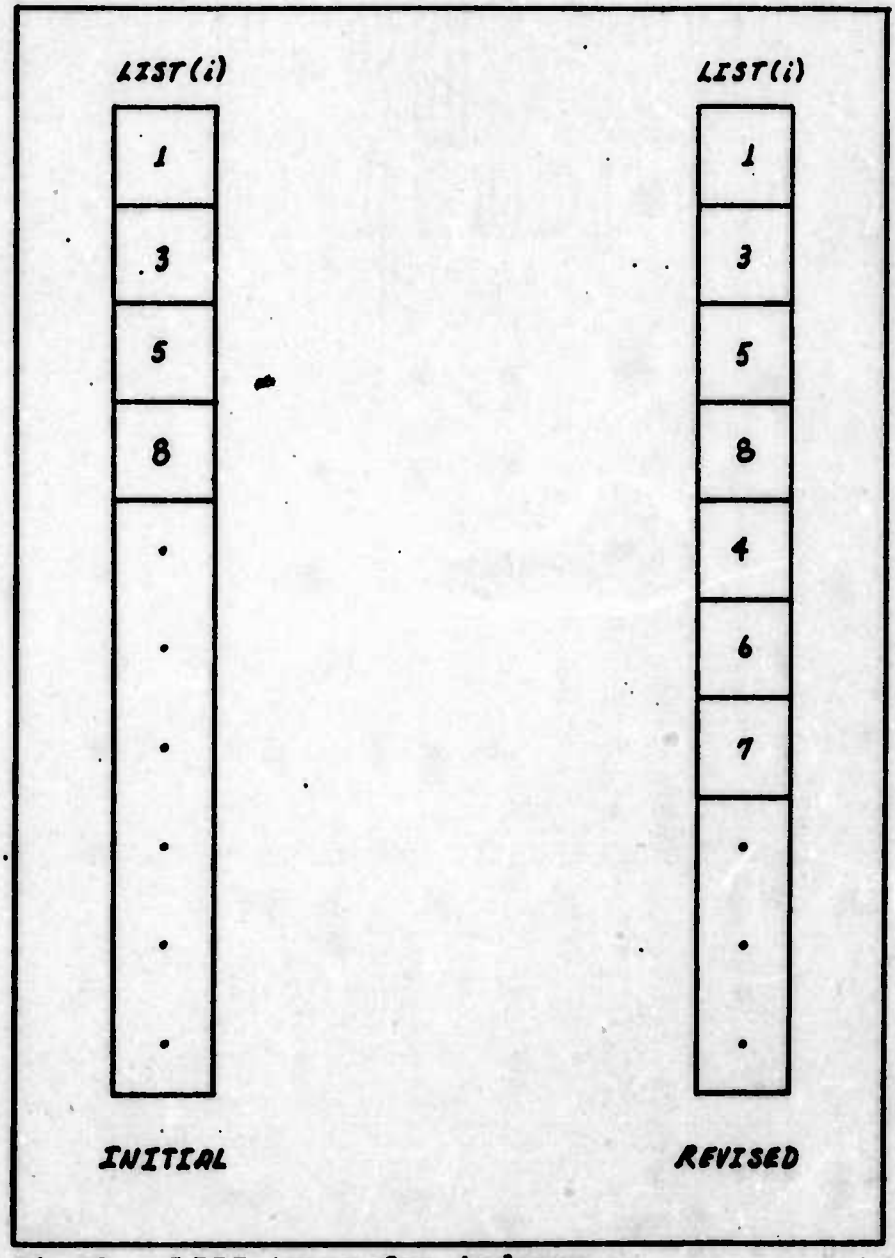


Fig13. LIST Array for Andrews

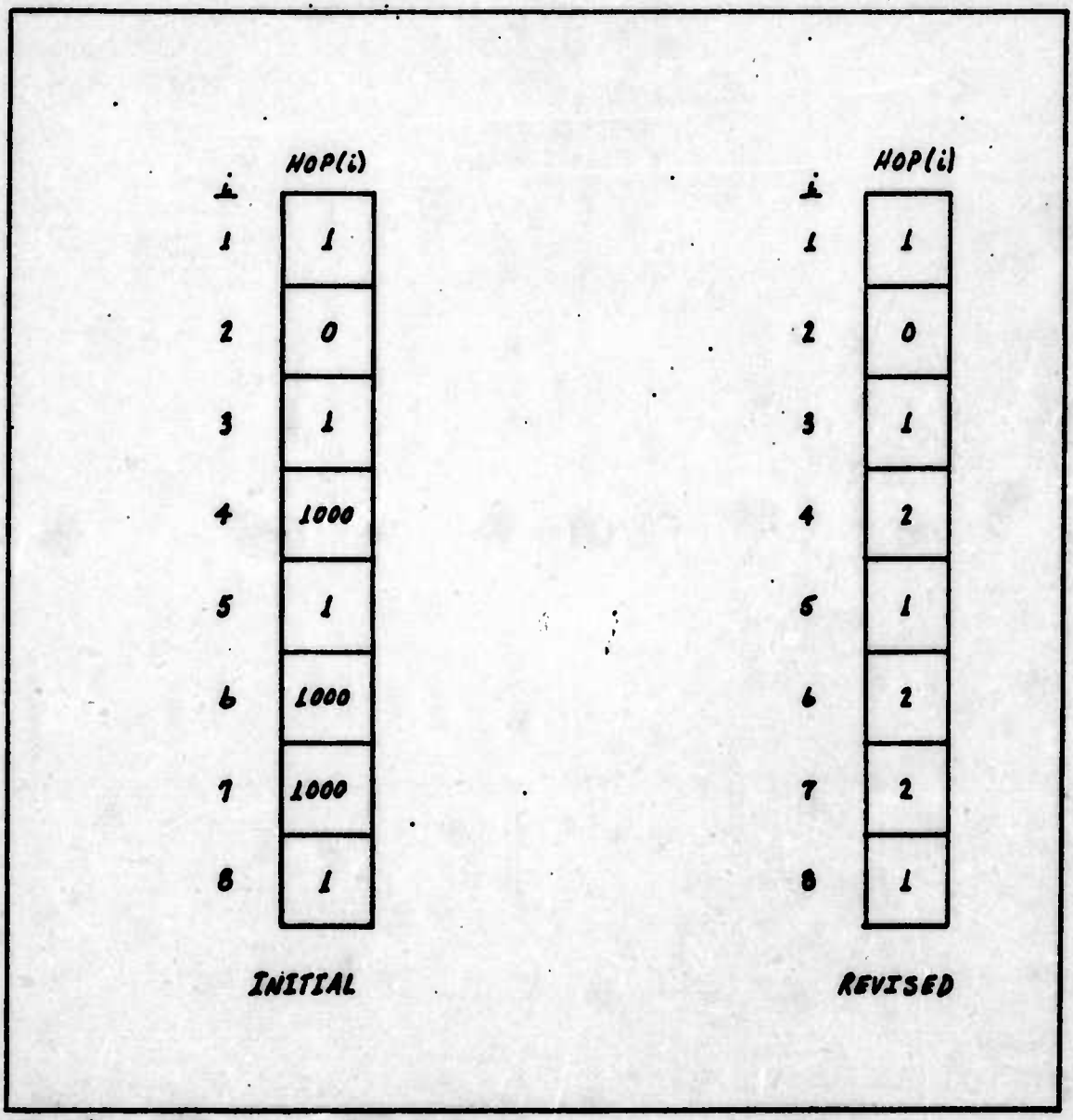


Fig14. HOP Array for Andrews

COMPUTE-D

COMPUTE-D completes the computations of the D table and the LIST and HOP arrays.

INCREMENTAL-D

INCREMENTAL-D is a procedure that is entered when only a congestion status change is reported. For a congestion status change it performs the same function as the INITIALIZE-D and COMPUTE-D processes but in much less time. Initializing the D table and LIST and HOP arrays and updating them for connectivity status changes must be performed by the INITIALIZE-D and COMPUTE-D modules. Since congestion reports are much more frequent than connectivity change reports, the INCREMENTAL-D saves a significant amount of processing time (Ref 7:33).

Per Packet Function

The per packet function determines the path a packet will take to reach its destination through a single module, PAKROUTE. PAKROUTE is executed for every packet entering the SCM. Figure 15 shows a flowchart of the per packet routing tasks. The routing tables developed by RTABLE and TTABLE are used to determine which outgoing link and trunk, if any, will be used.

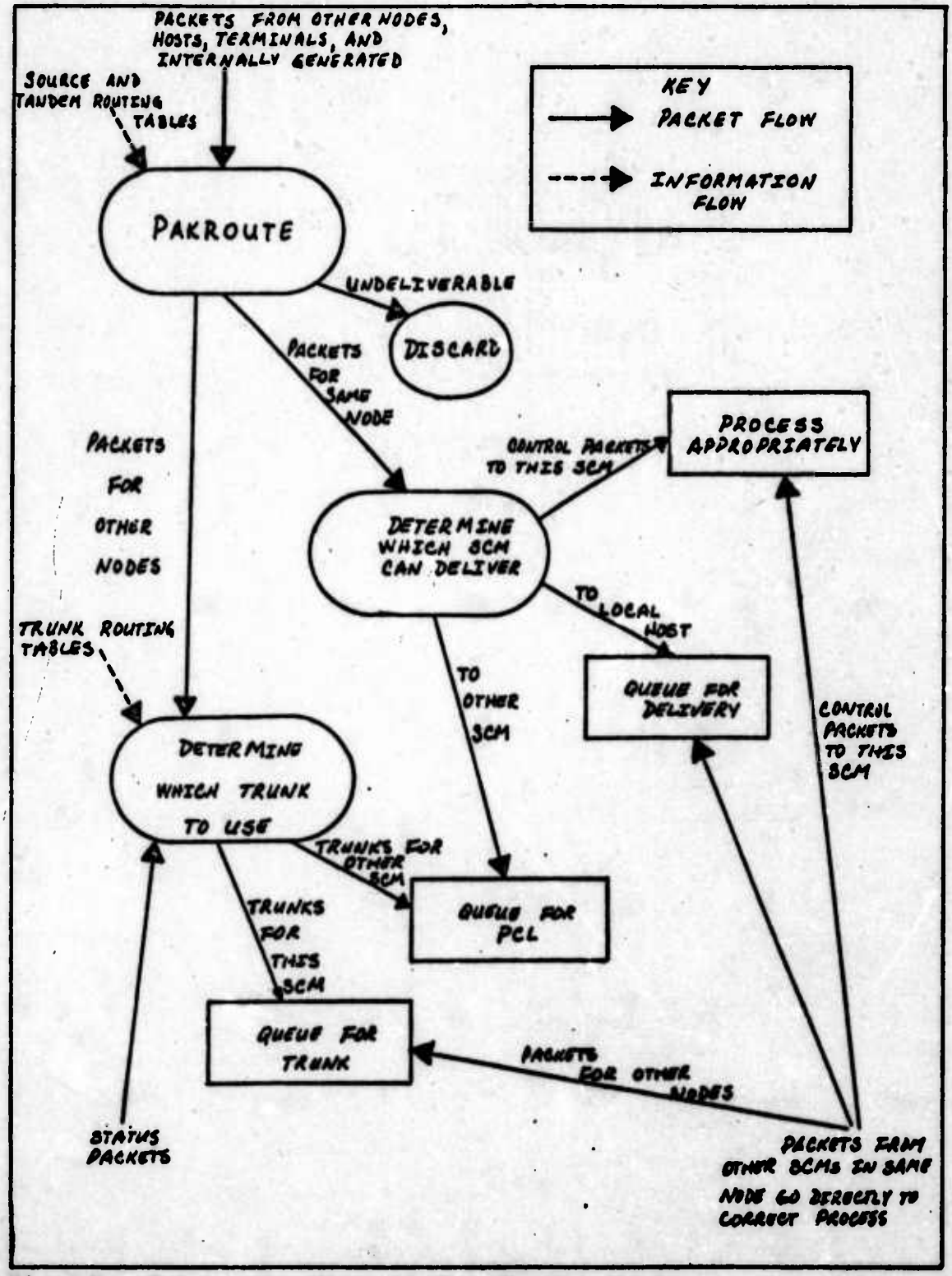


Fig 15. Schematic View of Per Packet Tasks

PAKROUTE splits the packets into three broad categories:

1. Packets destined for other nodes,
2. Packets destined for the present node,
3. Undeliverable packets (due to hardware failures separating part of the network, etc.)

Figure 16 shows the PAKROUTE module overview. PAKROUTE determines the path packets will take by first selecting the outgoing link through a table (ROUTE and SROUTE) lookup. The ROUTE and SROUTE tables are produced by RTABLE in the background functions. Secondly, the outgoing trunk is selected by another table (TROUTE) lookup. TROUTE (not shown in Figure 16) is produced by TTABLE in the background functions. Packets arriving at an SCM from another SCM in the same node can be queued directly to the correct trunk, that is, it is not necessary for a packet to process through two or more SCMs in the same node even when the appropriate trunk is connected at another SCM in the node. PAKROUTE is a simple module, however, it is executed for every packet which enters the SCM.

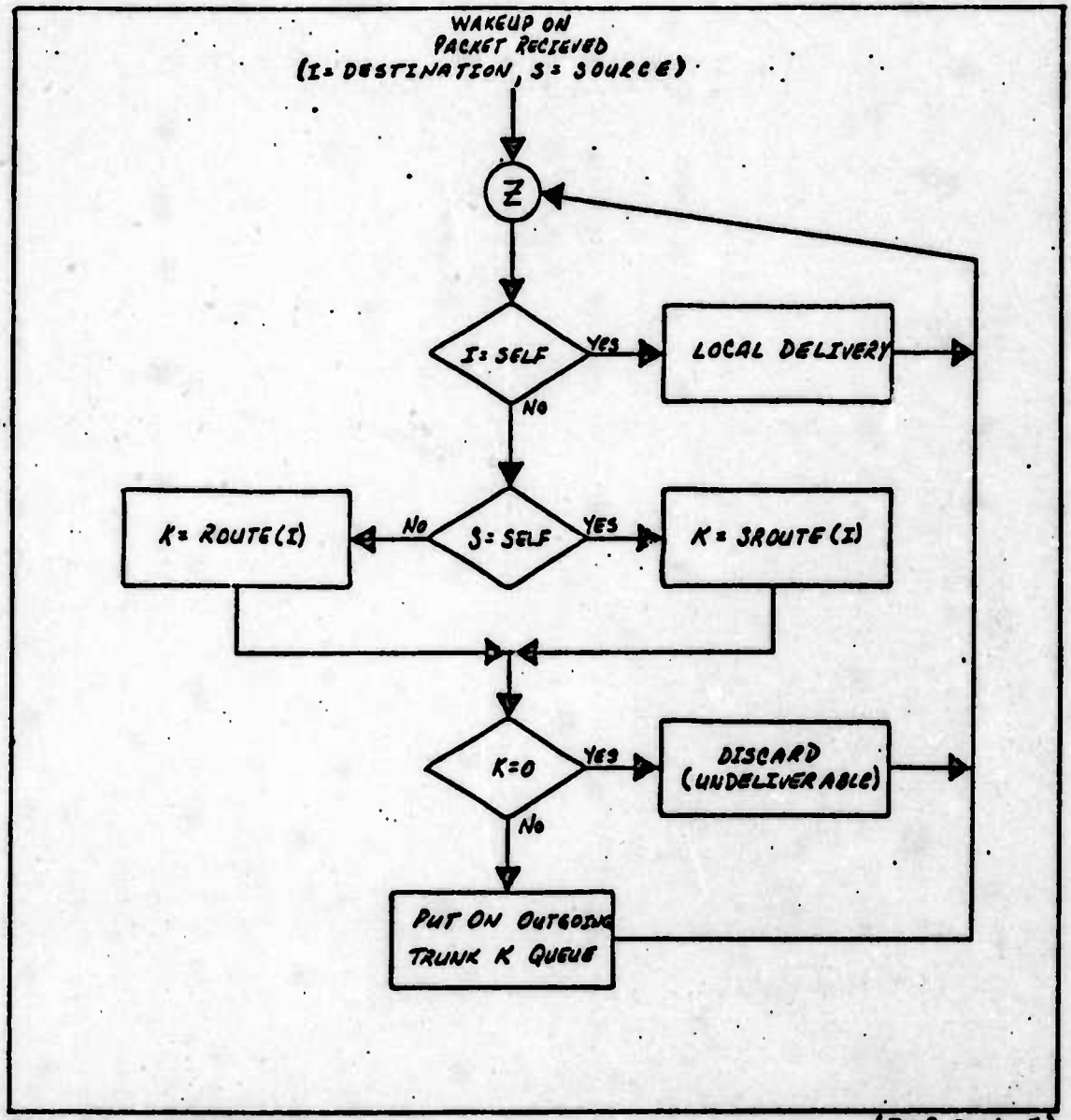


Fig 16. PAKROUTE (Ref 2:A-5)

CHAPTER 4

AUTODIN II SIMULATION

Constraints and Parameters

The AUTODIN II simulation model described in this chapter was developed to fill a void left by other AUTODIN II simulations. Rockwell International simulated a single Packet Switching Node (PSN), including Switch Control Modules (SCM), Terminal Access Controllers (TAC), and Line Control Modules (LCM) (Ref 3). Systems Control, Inc. and Capt. J. Whittenton simulated the complete network but only down to the Packet Switching Node (PSN) level (Ref 4; and 6:C1-C11). Input, output and processing control for the network, however, is accomplished at the SCM level. Therefore, the simulation developed as a part of this thesis models the AUTODIN II network down to the SCM level.

The model is limited to the network itself and does not include any provision for simulating external host computers and TAC line connections. Source packets are introduced at the SCM input queue. The network model maximum limits are eight PSNs, 25 SCMs, and 125 transmission lines. Any smaller configuration may be simulated. Any

single PSN may contain from one to five SCM's. Each SCM has one input queue (input line connection) and from one to five output queues (output line connections). The simulation model is constructed using the capabilities of the Q-GERT simulation language and is established through standard Q-GERT control statements and user supplied parameters.

The objects which will be tracked through the simulation, to provide statistical information concerning the AUTODIN II routing algorithm and modelled network, are Q-GERT transactions. These packets should not be confused with AUTODIN II packets.

> As an introduction to Q-GERT network modelling let us consider a one server system in which a single line of items forms before the server. The server of the system has a status: busy or idle. The server status changes as system conditions change. The server is busy when he is processing an item, otherwise he is idle. The items flow through the system. They arrive, possibly wait, are served, and depart the system. Such a sequence of events, activities and decisions is referred to as a process. Entities that flow through a process are called transactions. Thus, items are considered as transactions. A transaction can be assigned attribute values that enable a modeler to distinguish between individual transactions of the same type or between transactions of different types. For example, the time a transaction enters the system is an attribute of the transaction (Ref 8).

The characteristics of AUTODIN II packets are imposed upon Q-GERT transactions as attributes. Therefore, the arrival and flow of AUTODIN II packets is modelled by the generation and flow of Q-GERT transactions. When a transaction is discussed as an item flowing through Q-GERT it

will be called a Q-GERT transaction. When a transaction is discussed as an item flowing through AUTODIN II it will be called an AUTODIN II packet.

The packet flow may be separated into two segments for statistical tracking. This is done by placing the proper values in user supplied parameters. The capability was supplied to allow tracking a specific portion of AUTODIN II traffic against a background of all other traffic (see the NORAD scenario in Chapter 8), but can be used for other purposes.

The remainder of this chapter will describe the Q-GERT model of a Switch Control Module, and the User Input (UI), User Function (UF), User Subroutine (US), and User Output (UO), FORTRAN subroutines written to model the AUTODIN II routing algorithm.

Q-GERT

Only those portions of the Q-GERT simulation language which were used in the AUTODIN II model will be described. Anyone wishing more information concerning Q-GERT should refer to "Modeling and Analysis Using Q-GERT Networks" by A. Alan B. Pritsker (Ref 8).

There are two basic types of symbols used in Q-GERT networks. These are ovals, used to represent simulation nodes (not to be confused with AUTODIN II packet switching nodes), and lines, representing activities. A node

can represent a queue or be a branching point for simulation transactions. Activities occur between nodes, act as connections between nodes, and can accumulate time. Activities represent services rendered. Activities in the AUTODIN II model are used to accumulate SCM processing time and line transmission time.

The Q-GERT oval length is divided into three sections. The first section is divided into an upper and a lower portion. The top portion of a regular node (also source and sink nodes) contains the number of Q-GERT transactions which must enter the node the first time to cause a Q-GERT transaction to leave the node. The lower portion contains the number of Q-GERT transactions which must enter the node thereafter to release a transaction. The top portion of a Q-node contains the initial number of transactions in the queue and the lower portion is the number of transactions the queue can hold.

The middle section of a Q-node identifies the queueing discipline which is used within the queue. In a source node, it identifies the arrival rate for transactions. In regular or sink nodes, the middle section decides whether or not statistics will be collected at the node. The final section contains the identification number of the node.

The AUTODIN II model consists of three different Q-GERT networks. Network N1 in Figure 17 is the clock which establishes the FTICK and STICK time intervals discussed in the

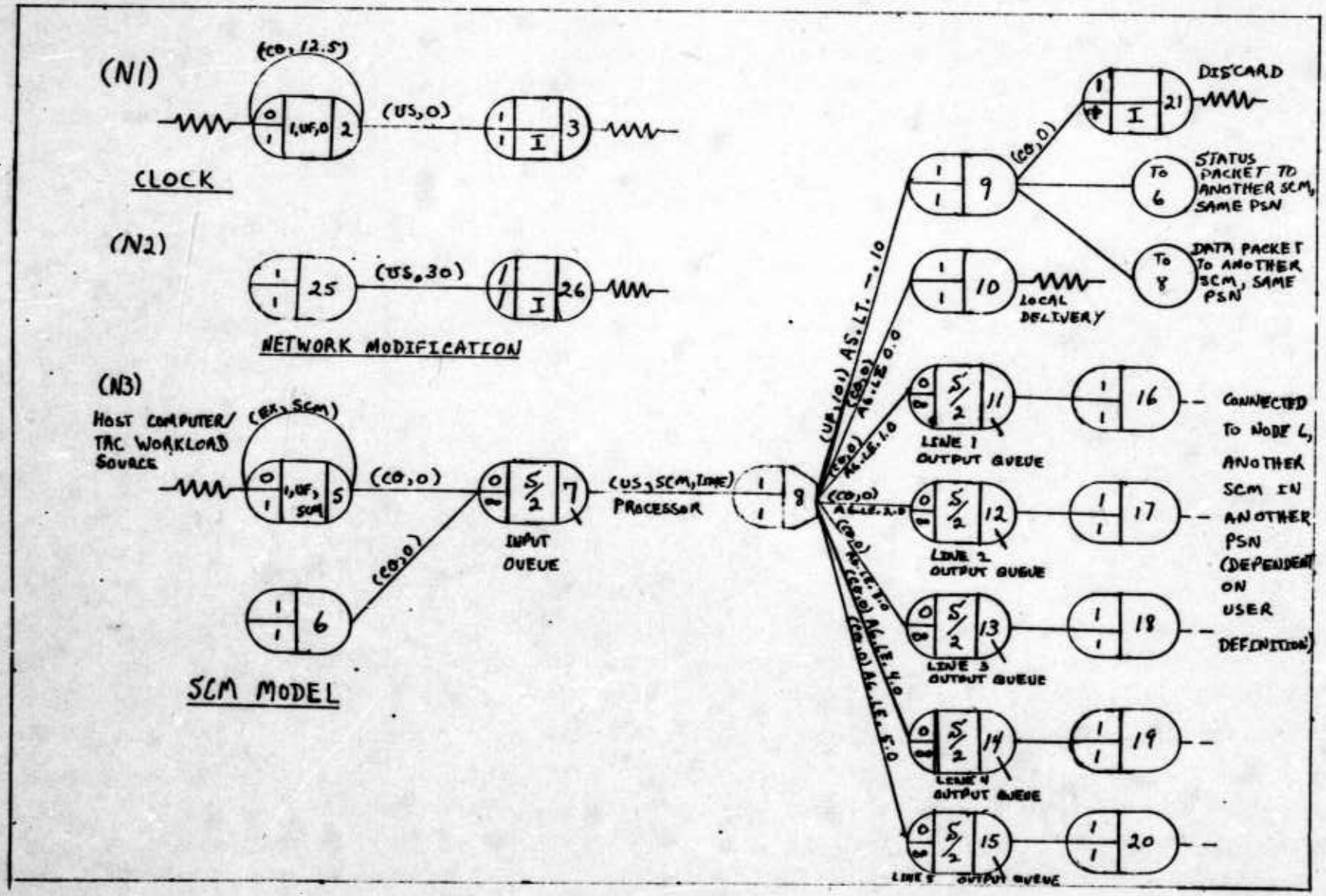


Figure 17 - Q-GERT Model of an SCM

description of MESGEN in Chapter Three. The basic time unit for the clock is one millisecond. The clock consists of a source node, an activity, and a sink node. The source node is identified by the jagged line entering it, while the sink node is identified by the jagged line leaving it. This is true of all source and sink nodes. The first time the source node becomes active it does not need a simulation transaction to activate it. The zero in the upper left portion of the oval signifies that this node can become active the first time without a Q-GERT simulation transaction. Thereafter, a transaction is required to activate the source node, as signified by the one in the lower left portion of the oval. When the node is activated, it generates transactions along all activities extending from the right portion of the oval. One of these activities returns to the left portion of the oval, and therefore, during the first activation, and subsequent activations, the source node generates the transaction which activates it next. The transaction traversing the loop activity is given a time delay of 12.5 milliseconds. This time delay was chosen to divide a 100 millisecond time interval into eight segments, so that for each Packet Switching Node, FTICK and STICK processing would occur at a different time. This simultaneous occurrence could cause simultaneous generation of AUTODIN II status packets at all nodes and was considered unrealistic by the authors.

While the source node is active a call is made to user

subroutine UF (portrayed in the central portion of the oval) and a parameter of zero is passed to that subroutine. This subroutine controls updating of the clock. The actions that occur due to the call will be discussed in the section concerning subroutine UF. Finally, there is an identification number in the right portion of the node.

An activity extends from the clock source node to the clock sink node. Transactions generated by the source node traverse this activity. In so doing they cause a call to user subroutine US and a parameter of zero is passed to that subroutine. This activates the message generation routine MESGEN. The actions resulting from the call will be discussed in the section concerning Subroutine US.

The clock sink node requires a transaction for every activation, including the first one. This is signified by the two ones in the left portion of the oval. The I in the lower middle portion of the oval identifies it as a node that collects interval statistics and its identification number is three.

The second network, N2, in Figure 17 allows modification of AUTODIN II network parameters (such as the rate packets arrive in the network, whether lines are up or down, priority and short packet percentages, and others) during the running of the simulation. N2 consists of a basic Q-GERT node, an activity and a sink node. The initial node requires a transaction for every activation, but has no entering activity.

Transactions which activate this node are generated in user subroutine UI and will be discussed in the section about Subroutine UI. Transactions traversing the activity from node 25 (the initial nodes identification number) cause a call to user subroutine US and a number 30 is passed to that subroutine as a parameter. The final node in N2 is the sink node, which acts in the same fashion as node three in network N1.

The final network, N3, represents an SCM, and occurs in multiple copies, one for each SCM (up to 25), but all copies are the same except for identification numbering. Q-GERT provides a method for creating multiple copies of portions of a simulation model which are similar (minor differences are allowed). This method was used to provide for multiple SCMs.

The SCM contains a source node, six queue nodes, a branching node, six basic nodes, and two sink nodes plus all connecting activities. The source node (node 5) generates the traffic which would come into this SCM from terminal access controllers and host computers. A call is made to user subroutine UF where AUTODIN II packet parameters become a part of the generated transaction. These parameters include priority, type of packet (long or short), source and destination Switching Control Modules, and for which segment of traffic the statistics are gathered. An activity, which does not delay the transaction, connects node five

to node seven. This activity is a dummy and provides only a connection. A basic node, number six, is also connected to node seven by the same type of activity. Node six represents the input connection for lines from SCMs which are contained in other Packet Switching Nodes.

Node seven represents the input queue at this Switching Control Module. It is a Q-GERT queue node and is recognizable by the tail at the lower right of the node oval. The infinity symbol in the lower left of the oval identifies the queue limit as infinite. The S identifies the queue order as ranking by priority and the two beneath the S states that parameter, or attribute, two is the priority parameter. The activity which follows node seven represents the SCM processor and does provide a time delay to transaction traversal. Since a queue node will not release a transaction to an activity until that activity is clear, the processor delay will cause transactions to back up in the input queue.

Activity seven calls subroutine US, passing it the identification number of the SCM this portion of the Q-GERT network represents. Further routing of the transactions within the Q-GERT network is determined here, and a processor delay time is returned to the activity.

Node eight is a branching node. This is evidenced by its shape and by the number of activities leaving it. The conditions for branching, which are listed one to an activity, are tested beginning with the top activity and

proceeding to the bottom activity. The first activity with a condition which tests true receives the transaction.

If transaction attribute five is less than -.10, the transaction will traverse the activity connecting node eight to node 9. A call will be made to user subroutine UF and a parameter of 101 passed to it during that traversal. User subroutine UF will determine whether the transaction should be discarded (passed to sink node 21) if it cannot be delivered, should be treated as an AUTODIN II status packet for another SCM in the same node, or should be treated as an AUTODIN II data packet for another SCM in the same node. If one of the last two decisions is reached, Subroutine UF will make the connection to route the transaction to the proper copy of the Q-GERT SCM model; to node six or node eight in that copy. If attribute five is equal to or larger than -.10 condition testing will continue.

If transaction attribute six is less than or equal to zero, the AUTODIN II packet which is represented by the Q-GERT transaction should leave the AUTODIN II network from this SCM. It is therefore sent to sink node ten, which is labeled "Local Delivery".

The remaining five branches from node eight have identical functions and will be described in common. Transaction attribute six, for any transaction selecting one of these activities, represents the transmission line over which the corresponding AUTODIN II packet would be routed. Nodes 11

through 15 are queue nodes and correspond to the AUTODIN II output queues for lines one through five. The activities which connect nodes 11 through 15 to nodes 16 through 20 all call user subroutine UF, passing it a parameter value 102. A transmission delay for the specified line is returned to the activity. After the delay time is complete, the transaction representing an AUTODIN II packet will pass through the appropriate basic node (16-20) to node six of the SCM to which that transmission line is connected. The total network line configuration is established by user input parameters to user subroutine UI and will be discussed in the next section, which describes that subroutine.

The Q-GERT network described here is deceptively simple since many of the functions of the simulation are performed in user subroutines UI, UF, US and UO. The functions contained in those subroutines correspond to the functions MESGEN, UPDATE, MINHOP, RTABLE, TTABLE and PAKROUT described in Chapter Three. The following four sections describe the contents of UI, UF, US and UO.

Subroutine UI (User Input)

Subroutines UI, US, UO and function UF are placed in the Q-GERT simulation language to provide a means by which Q-GERT users could insert their unique processing requirements. Since Q-GERT is written in FORTRAN these processes also should be written in FORTRAN. This is true for Q-GERT simulation in

general. Subroutine UI is called by processing internal to the Q-GERT simulation language after the standard Q-GERT network configuration input is processed. UI allows the simulation user to read in any user unique data records which will effect the Q-GERT network processing and to accomplish any processing initialization the user requires. A return must be made from UI to the calling process before any simulation transactions can be generated.

Early in the development of the AUTODIN II simulation model a decision was made to configure the network as much as possible through user input parameters. Therefore, six types of user input records were developed. Four of these (types 001 through 004 - Tables I through IV) establish the initial AUTODIN II network configuration for each simulation run. The remaining two (types 900 and 910 - Tables V and VI) provide the ability to vary selected parameters of the network while the simulation is in progress.

Card types 001 through 004 are required for each Switch Control Module. They provide the basic description of that SCM and the Packet Switching Node that contains it.

Each of the user input records contains the identification numbers of the Switch Control Module and Packet Switching Node which it's data describes. These are identified as ISCM and INODE in the record. The type 001 record also contains the number of Switch Control Modules in the Packet Switching Node specified. This number is used to insure the

TABLE I

CARD TYPE 001 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
INRSCM	5-6	Number of SCMs in this PSN.
GNMCHRS	7-16	The normal, or general, traffic rate originating at this SCM, in millions of characters per hour.
SHRTPKT	17-19	The proportion of total packets originating at this SCM, to be identified as a short packet.
ICONSCM	20-21 22-23 24-25 26-27 28-29	Other SCMs connected by trunk to this SCM.
SPMCHRS	30-38	Special, or identified, traffic originating at this SCM, in millions of characters per hour.
LINDLA	39-43 44-48 49-53 54-58 59-63	The number of miles between this SCM and other connected SCMs (the mileage must be in the same order as the connected SCMs in ICONSCM described above).
PRECDNC	64-66 67-69 70-72 73-75	The proportion of packets to be assigned to each precedence level (highest level to lowest level, respectively).
CRDTYPE	78-80	Type of card (001)

TABLE II

CARD TYPE 002 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
NODTRAP	44-48	Proportion of traffic, originating at this SCM, destined for node 1.
NODTRP 2	49-53	Proportion of traffic, originating at this SCM, destined for Nodes 2 through 8
3	54-58	
4	59-63	
5	64-66	
6	67-69	
7	70-72	
8	73-75	
CRDTYPE	78-80	Type of card (002)

TABLE III

CARD TYPE 003 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
SCMTRP1	44-48	Proportion of traffic, from node 1, to be locally delivered by this particular SCM.
SCMTRP2	49-53	Proportion of traffic, from nodes 3 through 8, to be locally delivered at this particular SCM.
3	54-58	
4	59-63	
5	64-66	
6	67-69	
7	70-72	
8	73-75	
CRDTYPE	78-80	Type of card (003).

TABLE IV

CARD TYPE 004 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
SATLEVEL	7-15	Saturation level for this SCM (node saturation level is determined by adding the SATLEVELs of all SCMs in the node).
CNGST1	39-43	Congestive level for the Link corresponding to ICONSCM on card 001.
CNGST2		Same as CNGST1
CNGST3		" " "
CNGST4		" " "
CNGST5		" " "
CRDTYPE	78-80	Type of card (004)

TABLE V

CARD TYPE 900 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
LNSTAT1 LNSTAT2 LNSTAT3 LNSTAT4 LNSTAT5	20-21 22-23 24-25 26-27 28-29	Line status. LNSTAT (i) must correspond in order with ICONSCM on the 001 card. Minus one is line down. Plus one is line up.
LNCHTM1 LNCHTM2 LNCHTM3 LNCHTM4 LNCHTM5	38-43 44-48 49-53 54-58 59-63	The time (in milliseconds) into the simulation run to change the lines status as indicated by the LNSTAT (i) code.
CRDTYPE	78-80	Card Type (900).

TABLE VI

CARD TYPE 910 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
GNMCHRS	7-16	General traffic originating at this SCM (in millions of characters per hour).
SHRTPKT	17-19	Percentage of short packets originating at this SCM.
SPMCHRS	30-38	Special, or identified, traffic originating at this SCM (in millions of characters per hour).
TIME	39-43	The number of milliseconds into the simulation the action will take place.
PRECDNC	64-66	Four precedence (priority) levels.
	67-69 70-72 73-75	Probability of traffic for each SCM.
CRDTYPE	78-80	Type of card (910).

correct number of SCMs are established for each node and is used as a cross reference against the type 001 received for each SCM. Two fields provide the parameters for establishing arrival rates of AUTODIN II packets from sources outside the network to node five of the SCM model (see Figure 17). Both fields have units in millions of characters per hour. The first of those fields is identified as GNMCHI` and represents the combination of all different types of traffic which can flow across the network. The second field is used to segregate a portion of the traffic and develop statistics on that portion only. This field is identified as SPMCHRS. The input unit of measure for these two fields was chosen to be understandable by users. The simulation, however, has a base unit time of one millisecond and a base unit transaction of a packet. Therefore, the input information must be converted to the correct unit of measure for the simulation by subroutine UI. A reference to the transaction generation loop for node five (Figure 17) shows that an exponential (EX) distribution is being used to model source packet arrival. A mean interarrival time (time between arrival of packets) must be supplied to use this distribution. Since packets come in two sizes, 4704 bits and 600 bits (Ref 4:77), with eight bits to the character, the percentage either of short packets or of long packets must be known. The percentage of short packets is contained in the type 001 input record as a decimal figure and is identified as SHRTPKP.

The method for developing the equation to convert millions of characters per hour to interarrival time in milliseconds followed this reasoning:

1. Total bits/hr = (GNMCHRS + SPMCHRS) X 8 bits/char

2. Total bits/hr + (Total packets/hr X (1.00-percent short packets) X 4704 bits/ long packer) + (total packets/hr X percent short packets X 600 bits/short packet)

3. (GNMCHRS + SPMCHRS) X 8 = (total packets/hr X (1.00 - SHRTPKP) X 4704 bits/ long packet) + (total packets/hr X SHRTPKP X 600 bits/short packet)

4. Total packets/hr = (((1.00-SHRTPKP) X 4704 bits/ long packet) + (SHRTPKP X 600 bits/short packet))÷ ((GNMCHRS + SPMCHRS) X 8 bits/char)

5. Total packets/millisecond = total packets/hr ÷ 3,600,000 millisecond/hr

6. Interarrival time between packets = 1 ÷ total packets/millisecond

The final equation is:

Interarrival time = 3,600,000 milliseconds/hr ÷ (((1.00-SHRTPKP) X 4704) + (SHRTPKP X 600)) ÷ ((GNMCHRS + SPMCHRS) X 8)

This interarrival time is computed by Subroutine UI from SHRTPKP, GNMCHRS and SPMCHRS supplied by user in card type 001.

The short package percentage, SHRTPKP, is also used in function UF for assignment of type of packet code to attribute one.

The type 001 contains four precedence levels (priority) fields identified as PRECDNC. Each field contains the percentage (in decimal point form) of all traffic originating

at that SCM which will carry the corresponding priority. These percentages must add to 1.0 and are used in function UF for assigning precedence code to attribute two of the Q-GERT simulation transaction.

The remaining fields in the type 001 record establish the network configuration and the transmission line mileage. There are five connectivity fields. Each one corresponds to a line from this SCM and contains the identification number of the SCM to which that line is connected. If a line is not connected, the field is left blank. A check is made that the connection between two SCMs is shown at both SCMs. The NFTBU array (Ref 8:243) in the Q-GERT simulation language processor is updated from this information to establish the connections between Q-GERT SCM models (as described in the preceding section). There are also five mileage fields; one corresponding to each line. These fields are used as part of the calculation which determines transmission delay on outgoing lines from the SCM. This computation is done in PAKROUT, which is contained in subroutine UO.

The type 002 user record contains eight fields (excluding the SCM and PSN identification number fields). Each field contains the percentage of traffic originating at this switching control module which is to be sent to the corresponding packet switching node, i.e. field one contains the percentage (in decimal point form) which when multiplied by the total number of packets gives the number to be sent to PSN one.

These percentages and the percentages from the type 003 user record are used to establish a destination SCM percentage which is then used in function UF to assign destination SCM codes to attribute four of the Q-GERT simulation transaction.

The type 003 user record also contains eight fields (again, excluding SCM and PSN fields). These fields contain a decimal number which represents the percentage of traffic to this PSN originating at the PSN whose identification number corresponds to each field and which is directed to this SCM. Chapter One of the User Manual (Appendix A) provides an example of these percentages.

The type 002 and type 003 records establish, respectively, a 25 by 8 matrix and an 8 by 25 matrix. When a matrix multiplication is done using these two matrices, the result is a 25 by 25 matrix containing the percentage of all traffic at a specific originating SCM which will be directed to a specific destination SCM. As specified before, this matrix is used to establish destination SCMs for the Q-GERT simulation transaction. This is called the DISTRIB Table.

The type 004 user record for each SCM contains a saturation level field and five congestion level fields. The saturation levels (in number of packets in queues) for all SCMs in a PSN are added together to establish a saturation level for the PSN. Each of the congestion levels (in number of packets in queue) in each SCM represents the corresponding transmission line's output queue. When the simulation is in progress if

the average of the contents of the output queues for a link exceeds the average of the congestion levels for the lines contained in the link, the link is declared to be congested. A link is defined as all the lines (trunk is used interchangeably with line) which connect one PSN with another PSN, no matter which SCMs within the two PSNs they connect. A congested link is weighted by the AUTODIN II routing algorithm so that it is used less often. When all input queue levels added together exceed the saturation level all links for that PSN are declared congested.

The type 900 and type 910 user records, as stated before, allow changes of parameters while the simulation is in progress. The type 900 transaction contains five line status change fields (one for each line in the SCM) and five corresponding status change time fields. A minus one in the line status change field means the line is going down, a plus one means it is coming up and a zero means no change. The corresponding change time field contains the number of milliseconds into the simulation at which the change is to occur. Each line status change field which is nonzero generates a Q-GERT transaction through use of the Q-GERT subroutine PTIN (REF 8:259) which is entered into Q-GERT node 25 (Figure 17) at the simulation time specified in the status change time field.

The type 910 record (Table VI) allows changes in GNMCHRS, SPMCHRS, SHRTPKP and the four PRECDNC fields and a recalculation of interarrival time. The structure of the 910

record is similar to the type 001 record. The type 910 also contains a time change field and the generation of Q-GERT transaction occurs in a manner similar to the ones generated for the type 900.

Several tables are generated and initialized from the data contained in user record types 001 through 004. The original data on the type 001 card is held in one of two arrays, depending on whether it was in real or integer format. These two arrays contain 25 lines each and are line indexed by SCM number. This gives such benefits as a cross reference from SCM to the PSN which contains it or from SCM to the SCMs which are connected with it by transmission lines.

There is also a PSN to SCM cross reference table (Table VII). The DISTRIB table has already been described (Table VIII). It is constructed from the two halves of the SCMEDIT table, which contains the information from the type 002 and type 003 user records (Table IX). There is a LINKS table (Table X) for each PSN. Each LINKS table contains 8 lines of 20 entries. Each line represents a PSN. The entrys in the table identify the transmission trunks which make up the link between the table owner PSN and the line PSN. The first entry in each line is the number of trunks in the link. The second entry is trunk one, the third is trunk two and so on, until all trunks are listed. Each entry contains four pieces of information. Positions

TABLE VII

PSN to SCM

PSN		SCM			
NODESCM	1	1	2	1000	1000	1000
Number of SCMs		2				
NODESCM	2	3	4	1000	1000	1000
Number of SCMs		2				
NODESCM	3	5	6	7	1000	1000
Number os SCMs		3				
NODESCM	4	8	9	1000	1000	1000
Number of SCMs		2				
NODESCM	5	10	1000	1000	1000	1000
Number of SCMs		1				
NODESCM	6	11	12	1000	1000	1000
Number of SCMs		2				
NODESCM	7	13	1000	1000	1000	1000
Number of SCMs		1				
NODESCM	8	14	15	16	1000	1000
Number of SCMs		3				

TABLE VIII

DISTRIB (DISTRIBUTION) TABLE

SCMs	SCMs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	DIST	.070	.070	.110	.110	.007	.007	.007	.035	.035	.060	.020	.020	.090	.119	.119	.1220
2	DIST	.070	.070	.110	.110	.007	.007	.007	.035	.035	.060	.020	.020	.090	.119	.119	.1220
3	DIST	.060	.060	.215	.215	.013	.013	.014	.035	.035	.120	.030	.030	.060	.033	.033	.0340
4	DIST	.060	.060	.215	.215	.013	.013	.014	.035	.035	.120	.030	.030	.060	.033	.033	.0340
5	DIST	.025	.025	.215	.215	.003	.003	.003	.040	.040	.060	.070	.070	.010	.073	.073	.0750
6	DIST	.025	.025	.215	.215	.003	.003	.003	.040	.040	.060	.070	.070	.010	.073	.073	.0750
7	DIST	.025	.025	.215	.215	.003	.003	.003	.040	.040	.060	.070	.070	.010	.073	.073	.0750
8	DIST	.045	.045	.090	.090	.007	.007	.007	.150	.150	.060	.060	.060	.070	.053	.053	.0540
9	DIST	.045	.045	.090	.090	.007	.007	.007	.150	.150	.060	.060	.060	.070	.053	.053	.0540
10	DIST	.030	.030	.190	.190	.010	.010	.010	.035	.035	.180	.030	.030	.060	.053	.053	.0540
11	DIST	.030	.030	.070	.070	.013	.013	.014	.030	.030	.050	.165	.165	.090	.076	.076	.0780
12	DIST	.030	.030	.070	.070	.013	.013	.014	.030	.030	.050	.165	.165	.090	.076	.076	.0780
13	DIST	.075	.075	.125	.125	.003	.003	.003	.035	.035	.110	.080	.080	.120	.043	.043	.0440
14	DIST	.070	.070	.065	.065	.007	.007	.007	.070	.070	.080	.070	.070	.070	.092	.092	.0950
15	DIST	.070	.070	.065	.065	.007	.007	.007	.070	.070	.080	.070	.070	.070	.092	.092	.0950
16	DIST	.070	.070	.065	.065	.007	.007	.007	.070	.070	.080	.070	.070	.070	.092	.092	.0950

TABLE IX

SCM EDIT TABLE

Source SCM \ Destination Nodes	1	2	3	4	5	6	7	8
1	.1400	.2200	.0200	.0700	.0600	.0400	.0900	.3600
2	.1400	.2200	.0200	.0700	.0600	.0400	.0900	.3600
3	.1200	.4300	.0400	.0700	.1200	.0600	.0600	.1000
4	.1200	.4300	.0400	.0700	.1200	.0600	.0600	.1000
5	.0500	.4300	.0100	.0800	.0600	.1400	.0100	.2200
6	.0500	.4300	.0100	.0800	.0600	.1400	.0100	.2200
7	.0500	.4300	.0100	.0800	.0600	.1400	.0100	.2200
8	.0900	.1800	.0200	.3000	.0600	.1200	.0700	.1600
9	.0900	.1800	.0200	.3000	.0600	.1200	.0700	.1600
10	.0600	.3800	.0300	.0700	.1800	.0600	.0600	.1600
11	.0600	.1400	.0400	.0600	.0500	.3300	.0900	.2300
12	.0600	.1400	.0400	.0600	.0500	.3300	.0900	.2300
13	.1500	.2500	.0100	.0700	.1100	.1600	.1200	.1300
14	.1400	.1300	.0200	.1400	.0800	.1400	.0700	.2800
15	.1400	.1300	.0200	.1400	.0800	.1400	.0700	.2800
16	.1400	.1300	.0200	.1400	.0800	.1400	.0700	.2800

Destination SCM \ Source Node	1	2	3	4	5	6	7	8
1	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
2	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
3	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
4	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
5	.3300	.3300	.3300	.3300	.3300	.3300	.3300	.3300
6	.3300	.3300	.3300	.3300	.3300	.3300	.3300	.3300
7	.3400	.3400	.3400	.3400	.3400	.3400	.3400	.3400
8	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
9	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
11	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
12	.5000	.5000	.5000	.5000	.5000	.5000	.5000	.5000
13	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
14	.3300	.3300	.3300	.3300	.3300	.3300	.3300	.3300
15	.3300	.3300	.3300	.3300	.3300	.3300	.3300	.3300
16	.3400	.3400	.3400	.3400	.3400	.3400	.3400	.3400

TABLE X

LINKS TABLE

From		To Node	LINKS			
NODE= 1	LINKS	1	0	0	0	0
NODE= 1	LINKS	2	1	402103	0	0
NODE= 1	LINKS	3	3	201205	301306	302407
NODE= 1	LINKS	4	0	0	0	0
NODE= 1	LINKS	5	0	0	0	0
NODE= 1	LINKS	6	0	0	0	0
NODE= 1	LINKS	7	2	101113	102213	0
NODE= 1	LINKS	8	1	202316	0	0
NODE= 2	LINKS	1	1	103402	0	0
NODE= 2	LINKS	2	0	0	0	0
NODE= 2	LINKS	3	2	303105	204307	0
NODE= 2	LINKS	4	0	0	0	0
NODE= 2	LINKS	5	1	304410	0	0
NODE= 2	LINKS	6	0	0	0	0
NODE= 2	LINKS	7	0	0	0	0
NODE= 2	LINKS	8	2	203315	104216	0
NODE= 3	LINKS	1	3	205201	306301	407302
NODE= 3	LINKS	2	2	105303	307204	0
NODE= 3	LINKS	3	0	0	0	0
NODE= 3	LINKS	4	3	405408	106309	107409
NODE= 3	LINKS	5	2	206210	207310	0
NODE= 3	LINKS	6	0	0	0	0
NODE= 3	LINKS	7	0	0	0	0
NODE= 3	LINKS	8	2	305314	406215	0
NODE= 4	LINKS	1	0	0	0	0
NODE= 4	LINKS	2	0	0	0	0
NODE= 4	LINKS	3	3	408405	309106	409107
NODE= 4	LINKS	4	0	0	0	0
NODE= 4	LINKS	5	1	509110	0	0
NODE= 4	LINKS	6	2	108111	109112	0
NODE= 4	LINKS	7	1	208313	0	0
NODE= 4	LINKS	8	2	308214	209116	0
NODE= 5	LINKS	1	0	0	0	0
NODE= 5	LINKS	2	1	410304	0	0
NODE= 5	LINKS	3	2	210206	310207	0
NODE= 5	LINKS	4	1	110509	0	0
NODE= 5	LINKS	5	0	0	0	0
NODE= 5	LINKS	6	0	0	0	0
NODE= 5	LINKS	7	0	0	0	0
NODE= 5	LINKS	8	0	0	0	0
NODE= 6	LINKS	1	0	0	0	0
NODE= 6	LINKS	2	0	0	0	0
NODE= 6	LINKS	3	0	0	0	0
NODE= 6	LINKS	4	2	111108	112109	0
NODE= 6	LINKS	5	0	0	0	0
NODE= 6	LINKS	6	0	0	0	0
NODE= 6	LINKS	7	1	312413	0	0
NODE= 6	LINKS	8	2	211114	212115	0

two and three contain the SCM number to which the trunk is connected in the owner PSN. Position one identifies which line it is of that SCM. Positions five and six identify the SCM to which the transmission trunk is connected and position four identifies which trunk it is in that SCM.

An LD (Link Distance) table is initialized as described in Chapter Three and Figure 4. This is done by setting the entire table to infinity (or 1000) and then those positions that have PSN to PSN links to one.

The NC (Node Connectivity) array (as portrayed in Figure 13) was changed to an eight by eight array containing the same information with infinity (1000) where no link exists. This eliminated the need for a P array which is not included in the simulation.

The final tasks performed in UI are calls to MINHOP and RTABLE in function UF to complete initialization. After this a return to the caller of UI initiates the simulation.

FUNCTION UF (USER FUNCTION)

Function UF performs four of the background tasks described in Chapter Three. They are MINHOP, UPDATE, RTABLE and TTABLE. It also assigns the attributes to Q-GERT simulation transactions. This includes not only those which represent AUTODIN II packets, but those which flow across the clock portion of the Q-GERT network. It makes the required branch-

ing decision when a transaction traverses the activity from Q-GERT node eight to Q-GERT node nine. Finally UF supplies the line transmission delay time for transactions simulating AUTODIN II packets flowing to other SCMs in other PSNs.

UF ENTRY. The UF function is called from Q-GERT nodes two and five. It is called from the activity between Q-GERT nodes eight and nine and from the five activities between Q-GERT nodes 11,12,13,14,15 and Q-GERT nodes 16,17, 18,19,20. When UF is called from Q-GERT node two a zero is passed as a parameter. This identifies the call as coming from the clock portion of the Q-GERT network and UF assigns attributes (there are 10 attributes in this simulation's transactions) to the clock transaction. Attributes one and two are assigned constants of 1.0 and 0.0 respectively. Attribute three is supplied by a counter called ICLOCK. ICLOCK is initially zero. Each time UF is called from Q-GERT node two ICLOCK is increment by one. If ICLOCK becomes greater than eight it is reset to one. After this process is accomplished the contents of ICLOCK are assigned to attribute three of the Q-GERT simulation transaction. Since a transaction is generated across the clock network every 12.5 (simulation) milliseconds, every 100 millisecond interval is divided into eight segments. The clock transaction is used to initiate the MESGEN process in subroutine US and provides an FTICK/STICK time interval, as described in Chapter Three, for each Packet Switching Node. Attribute

three selects the PSN which will generate "Bad News/Good News" status messages (Chapter 4, Subroutine US, MESGEN). Keying all nodes every 100 millisecond could cause peaks in network traffic due to the possible release of 56 status packets at the same time. Initiating a separate node every 12.5 milliseconds precludes that from happening.

When UF is called from Q-GERT node five the identification number of the Q-GERT SCM Model network is passed to UF as a parameter. This number corresponds to a particular SCM and is used as the SCM's identification. The number will be between one and twenty-five. Using the SCM number (as an index) and the data arrays established in subroutine UI from user transactions 001 through 004, attributes which describe the simulated AUTODIN II packet can be assigned to the Q-GERT transaction. Attribute one is packet type. The code for attribute one is determined by using a random number obtained from the Q-GERT subroutine DRAND (Ref 8:444) and the SCMs percent of short packets (SHRTPKP). If the packet is a short packet a two is placed in attribute one. If it is a long packet a three is placed in attribute one. A one in attribute one signifies a AUTODIN II system status packet. Status packets are generated in MESGEN.

Attribute two is packet priority. A cumulative array of four values is developed using the SCMs precedence percentages (PRECDNC). A corresponding array of four codes, (10,20,30,40) is also built. These two arrays are passed to

Q-GERT subroutine DPROB (Ref 8:251) which select the packet priority from the codes.

Attribute three identifies the source SCM and is the SCM number. Attribute four is the Destination SCM. Another cumulative array with 25 entries is established using the distribution table (DISTRIB) developed by the matrix operation in UI. A code array containing the numbers 1 to the maximum SCM number is also built. These two arrays are passed to the Q-GERT subroutine DPROB and it selects the destination SCM to be placed in attribute four. If the packet is a long packet, it may be from either the special traffic or the general traffic. A percentage is developed by dividing SPMCHRS by SPMCHRS plus GNMCHRS. If this percentage is greater than a random number between zero and one (obtained from DRAND), attribute ten is assigned a two for special traffic. If the percentage is less than the random number, attribute ten is assigned a one for general traffic.

Attributes five through nine are assigned by foreground task PAKROUT, which is in subroutine UO. PAKROUT is called from subroutine US when a Q-GERT transaction passes between Q-GERT nodes seven and eight.

When UF is called from the activity between Q-GERT nodes eight and nine a parameter containing 101 is passed to UF. This indicates one of three conditions have occurred:

1. The packet's destination has been disconnected and the packet is to be discarded. Also network status packets that have reached their destination and have been processed are sent

here to be discarded.

2. A status packet is to be delivered to another SCM in the same PSN for background processing.

3. A status or data packet's destination is in another node but the packet must enter the line output queue of another SCM within the current PSN to reach that node.

The conditions were detected during the Q-GERT simulation transaction's traversal of the activity between Q-GERT nodes seven and eight. A call is made to subroutine US, which then calls PAKROUT (in subroutine UO). PAKROUT does the required tests and assigns the values to attributes five, six and eight to cause processing along this activity. The primary condition (see Figure 17) was that attribute five be less than -.10.

Immediately upon entering UF, attribute five is made positive by using the FORTRAN ABS Function. If condition two or three is the reason for entering, attribute five is the next SCM for packet delivery. If attribute six is equal to or less than minus one, condition one holds and the packet is discarded at this SCM in Q-GERT node nine. The requires use of Q-GERT subroutine NFTBU to insuring routing to Q-GERT node nine of this SCM model.

If attribute six is greater than minus one, and attribute one is less than two, the packet is a status packet and either condition two or three holds. The packet, through use of Q-GERT subroutines NODCV (Ref 8:344) and NFTBU, is routed to Q-GERT node six of the SCM model identified in

attribute five.

If attribute six is greater than minus one, and attribute one is equal to or greater than two, the packet is a data packet and condition three holds. The packet is routed to Q-GERT node eight of the SCM identified in attribute five.

When UF is called from one of the activities between Q-GERT nodes 11,12,13,14,15 and Q-GERT nodes 16,17,18,19,20 a parameter containing 102 is passed to UF. This signifies that a packet will be transmitted across a transmission line. UF returns the contents of attribute nine to the activity. Attribute nine contains the transmission delay time to be used by the activity. The transmission delay time is calculated as follows:

delay time = ((56kilobits/sec ÷ 1000 millisecond/sec) X number chars/packet) + (trunk miles ÷ 100 miles/millisecond)

UPDATE. The simulation subroutine UPDATE (discussed in Chapter Three) consists of MINHOP, RTABLE and TTABLE. If UPDATE is called by MESGEN due to a change in PSN status at this node, all three background tasks are processed. If UPDATE is called due to receipt of a status packet from another node only RTABLE and TTABLE are initiated.

MINHOP. The simulation subroutine MINHOP builds three tables. These are the D table or distance table (Table XI), the HOP table (Table XII) and the LIST table (Table XIII). No function could be found for the LIST table. Therefore the process to build it will not be described.

TABLE XI

DISTANCE TABLE

To Node	From Node		LINKS					
			1	2	3	4	5	6
1	1	D	0.00	0.00	0.00	0.00	0.00	0.00
2	1	D	1.00	2.00	1000.00	2.00	1000.00	1000.00
3	1	D	2.00	1.00	1000.00	2.00	1000.00	1000.00
4	1	D	3.00	2.00	2.00	2.00	1000.00	1000.00
5	1	D	2.00	2.00	3.00	3.00	1000.00	1000.00
6	1	D	1000.00	3.00	2.00	2.00	1000.00	1000.00
7	1	D	1000.00	1000.00	1.00	1000.00	1000.00	1000.00
8	1	D	2.00	2.00	1000.00	1.00	1000.00	1000.00
1	2	D	1.00	2.00	1000.00	2.00	1000.00	1000.00
2	2	D	0.00	0.00	0.00	0.00	0.00	0.00
3	2	D	2.00	1.00	2.00	2.00	1000.00	1000.00
4	2	D	3.00	2.00	2.00	2.00	1000.00	1000.00
5	2	D	1000.00	2.00	1.00	1000.00	1000.00	1000.00
6	2	D	1000.00	3.00	1000.00	2.00	1000.00	1000.00
7	2	D	2.00	3.00	1000.00	3.00	1000.00	1000.00
8	2	D	2.00	2.00	1000.00	1.00	1000.00	1000.00
1	3	D	1.00	2.00	1000.00	1000.00	2.00	1000.00
2	3	D	2.00	1.00	1000.00	2.00	2.00	1000.00
3	3	D	0.00	0.00	0.00	0.00	0.00	0.00
4	3	D	1000.00	1000.00	1.00	2.00	2.00	1000.00
5	3	D	1000.00	2.00	2.00	1.00	1000.00	1000.00
6	3	D	1000.00	1000.00	2.00	1000.00	2.00	1000.00
7	3	D	2.00	1000.00	2.00	1000.00	3.00	1000.00
8	3	D	2.00	2.00	2.00	1000.00	1.00	1000.00
1	4	D	2.00	3.00	3.00	2.00	2.00	1000.00
2	4	D	2.00	2.00	1000.00	3.00	2.00	1000.00
3	4	D	1.00	2.00	1000.00	1000.00	2.00	1000.00
4	4	D	0.00	0.00	0.00	0.00	0.00	0.00
5	4	D	2.00	1.00	1000.00	1000.00	1000.00	1000.00
6	4	D	1000.00	1000.00	1.00	2.00	2.00	1000.00
7	4	D	1000.00	1000.00	2.00	1.00	1000.00	1000.00
8	4	D	2.00	1000.00	2.00	1000.00	1.00	1000.00
1	5	D	2.00	2.00	1000.00	1000.00	1000.00	1000.00
2	5	D	1.00	2.00	1000.00	1000.00	1000.00	1000.00
3	5	D	2.00	1.00	2.00	1000.00	1000.00	1000.00
4	5	D	1000.00	2.00	1.00	1000.00	1000.00	1000.00
5	5	D	0.00	0.00	0.00	0.00	0.00	0.00
6	5	D	1000.00	3.00	2.00	1000.00	1000.00	1000.00
7	5	D	1000.00	3.00	2.00	1000.00	1000.00	1000.00
8	5	D	2.00	2.00	2.00	1000.00	1000.00	1000.00
1	6	D	1000.00	2.00	2.00	1000.00	1000.00	1000.00
2	6	D	3.00	3.00	2.00	1000.00	1000.00	1000.00
3	6	D	2.00	3.00	2.00	1000.00	1000.00	1000.00
4	6	D	1.00	2.00	2.00	1000.00	1000.00	1000.00
5	6	D	2.00	1000.00	1000.00	1000.00	1000.00	1000.00
6	6	D	0.00	0.00	0.00	0.00	0.00	0.00
7	6	D	2.00	1.00	1000.00	1000.00	1000.00	1000.00
8	6	D	2.00	1000.00	1.00	1000.00	1000.00	1000.00

TABLE XII

HOP TABLE

NODES

	Node	1	2	3	4	5	6	7	8
HOP	1	0.00	1.00	1.00	2.00	2.00	2.00	1.00	1.00
HOP	2	1.00	0.00	1.00	2.00	1.00	2.00	2.00	1.00
HOP	3	1.00	1.00	0.00	1.00	1.00	2.00	2.00	1.00
HOP	4	2.00	2.00	1.00	0.00	1.00	1.00	1.00	1.00
HOP	5	2.00	1.00	1.00	1.00	0.00	2.00	2.00	2.00
HOP	6	2.00	2.00	2.00	1.00	2.00	0.00	1.00	1.00
HOP	7	1.00	2.00	2.00	1.00	2.00	1.00	0.00	2.00
HOP	8	1.00	1.00	1.00	1.00	2.00	1.00	2.00	0.00

TABLE XIII

LIST TABLE

	Node		Node		Identification				
LIST	1	2	3	7	8	1000	1000	1000	1000
LIST	2	1	3	5	8	1000	1000	1000	1000
LIST	3	1	2	4	5	8	1000	1000	1000
LIST	4	3	5	6	7	8	1000	1000	1000
LIST	5	2	3	4	1000	1000	1000	1000	1000
LIST	6	4	7	8	1000	1000	1000	1000	1000
LIST	7	1	4	6	1000	1000	1000	1000	1000
LIST	8	1	2	3	4	6	1000	1000	1000

TABLE XIV

NODE CONNECTIVITY TABLE

	Node		Node		Identification				
NC	1	2	3	7	8	1000	1000	1000	1000
NC	2	1	3	5	8	1000	1000	1000	1000
NC	3	1	2	4	5	8	1000	1000	1000
NC	4	3	5	6	7	8	1000	1000	1000
NC	5	2	3	4	1000	1000	1000	1000	1000
NC	6	4	7	8	1000	1000	1000	1000	1000
NC	7	1	4	6	1000	1000	1000	1000	1000
NC	8	1	2	3	4	6	1000	1000	1000

A call to MINHOP from UI established these tables initially. Entries to the tables are produced from the LD or link distance table and the NC or node connectivity table. The LD table is a matrix with rows identified as source PSN's and columns identified as destination PSNs. If there is a line connecting a source and a destination PSN and that line is not congested, there is a 1.00 in that matrix position. If there is a direct connection, but the line is congested, there is a 1.01 in that matrix position. If there is no direct connection, the matrix position contains 1000. Both the LD and NC tables are produced in UI.

There is an NC table for each PSN. There are as many entry positions for each NC table as there are PSNs. The table contains the identification numbers of the PSN's to which the owner PSN is directly connected, beginning with lowest numbered PSN to the highest numbered PSN in each succeeding entry position. Entry positions not filled have 1000 in them (Table XIV).

There is a D table for every PSN. The D table rows are assigned to PSNs and the columns are assigned to links. Since there can be only one link between the table owner and each PSN, the table has as many columns as exist PSN's.

The D table entries are either minimum hop distances (minimum number of transmissions line traversals needed to travel from the source PSN to the destination PSN), minimum hop distances plus one, or 1000 (signifying this entry is

not to be used). If a destination PSN can be reached in minimum hops or minimum plus one hops on this link, the entry is that number of hops.

There is a HOP table for each PSN. There is one HOP table entry for each PSN. That entry is the minimum hops required to reach that PSN from the owner PSN. If there is no connection, direct or indirect, between the two the entry is 1000.

The Algorithm for building the D and HOP tables is:

1. Place 1000 in every entry in the tables. Place zeros in those entries connecting the owner PSN to itself.

2. Set a link index to one.

3. Select the D and HOP table owner PSN row from the LD table. Examine each column entry of the LD table, using the PSN number as an index. If the entry is 1.00 or 1.01 (positive) place a one in that PSN position in the HOP table and a one in the matrix position indexed by the PSN number for row and link index number for column. Add one to the link index. If the entry is -1.00 or -1.01 or 1000 do not change the entry. Repeat this step until all entries for that LD row are examined. Note that a minus in the LD table means there is a connection normally, but not when the entry is minus (i.e. the link is down).

4. Set a hop count to one. This count indexes the level of hops used as a basis for the remaining search.

5. Examine each link column from table D in turn. If an entry in the link column is equal to the hop count, determine the row index (Destination PSN number). Select each succeeding entry in that PSN's NC table. If the NC entry is a PSN number examine that row position in D table link column in process. If the entry is greater than the hop count replace that entry with the

hop count plus one. When every link column has been examined add one to hop count and do step five again. Continue this process until no change is made to the table during an examination of all link columns.

6. In each row of the D table find the minimum number. This is the minimum hop. Replace any row entry which is not equal to minimum hop or minimum hop plus one with 1000. Place each row's minimum hop in the corresponding position in the HOP table.

This process is repeated for each PSN to construct all eight D tables and HOP tables.

The reasoning behind this algorithm is that since direct connectivity is known, those PSNs directly connected to a PSN which is directly connected to the owner PSN can be no more than two hops away (they may be only one). Those PSN's directly connected to a PSN two hops away, can be no more than three hops away. Thus the D table is built in layered increments.

RTABLE. RTABLE produces two tables, SROUTE and ROUTE. RTABLE works on a per PSN basis. When RTABLE is called initially from UI, it is called once for each PSN. Subsequent calls occur due to processing within a specific Q-GERT SCM model, which defines the SCM and, therefore, the PSN for which RTABLE will process.

RTABLE uses the LINKS table to determine which lines are in each link from the processing PSN to other PSNs. Recall that a link consists of all lines between two PSNs. The total number of packets in output queues for a single link is accumulated by accessing each line's output queue

in that link using the Q-GERT function XNINQ (Ref 8:260). This is then divided by the number of lines in the link to obtain the average number of packets in output queues for that link. This is done for each link from the processing PSN.

With this information and the processing PSN's D and HOP tables, RTABLE is ready to build tables ROUTE and SROUTE. Each of these tables has eight entries, one for each PSN. The SROUTE table is used when a packet's source SCM is in the PSN processing the packet. ROUTE is used when the PSN processing the packet is an intermediate node between the packet's source and destination SCM's.

The algorithm for building SROUTE and ROUTE is:

1. Begin with the PSN index set to one and SROUTE and ROUTE entries set to 1000.

2. Examine the D table row indexed by the PSN index a column at a time. If the integer portion of the link entry is equal to the HOP table entry for the same PSN index and is not infinite (1000), determine if the link is congested (entry has .01 added to it). If it is multiply the integer portion times 7 and add it to the average number of packets queued for the link. If the resulting number is less than the corresponding entry (indexed by the PSN index) in the SROUTE table, replace the entry and do the same for the entry in the ROUTE table.

3. If the integer portion of the link entry is equal to the corresponding HOP entry plus one, do the computation outlined in step 2 but add seven for the extra hop. If the resulting number is less than the corresponding entry in SROUTE (and SROUTE only) replace the entry.

4. After all links are examined add one to the PSN index and redo steps 2 and 3. Continue until the PSN index is larger than the number of PSNs.

The algorithm determines the best links to be used from source PSN to destination PSN and places them in tables ROUTE and SROUTE.

TTABLE. TTABLE builds table TROUTE. Whenever RTABLE is called TTABLE will also be processed. TTABLE also works on a per PSN basis. TROUTE contains the best Switch Control Module and line to the next Packet Switching Node.

There is a TROUTE table for each PSN. Each TROUTE table contains two entries for each link in the PSN. These two entries define which line in the link contains the minimum queue length. The first entry is the SCM which contains the line and the second entry is the line number.

The processing is a straight forward testing of output queue for each line in a link to find the one with the minimum queue. This is done using the Q-GERT function XNINQ. That line and the SCM it is in make up the entries in TROUTE.

SUBROUTINE US (USER SUBROUTINE)

Subroutine US (User Subroutine) is called from three activities in the Q-GERT network. These are the activity between Q-GERT nodes two and three in the clock network, the activity between Q-GERT nodes 25 and 26 in the modification network, and between Q-GERT nodes seven and eight (the processor activity) in the SCM model network. The parameters passed are zero from the clock network, 30 from the modification network and the SCM model copy identification number

(1 to 25) from the SCM model network, to identify the actions to be taken by Subroutine US.

If the parameter passed is one through twenty-five attribute two of the simulation transaction is checked. If the attribute (AUTODIN II packet priority) is above ten a processor delay time of 5 milliseconds is passed back to the activity. If the packet priority is ten or less a processor delay time of 3 milliseconds is passed back to the activity. Finally, a call is made to PAKROUT (in Subroutine UO) before control is returned to the activity. PAKROUT determines the next destination of the packet.

If the parameter passed is 30, attribute four is either 900 or 910. If it is 900, the simulation transaction is either a line up or a line down transaction. Attribute three is the SCM which is connected to one end of the line. The SCM at the other end of the line is obtained from the connectivity information supplied in the user type one record. This requires the use of attribute three and the line number in attribute two. From this information the PSNs with a change in line status are determined and the appropriate LINKS table entry is selected. The line positions in both SCM's connectivity information fields are turned negativity if the line is going down or turned positive if the line is coming up. If attribute one is minus one the line is going down. If it is plus one the line is coming up.

If attribute four is 910, the simulation transaction

is a change transaction that changes either GNMCHRS, SPMCHRS, SHRTPKP or the four PRECDNC fields or any combination of those fields. These fields are in attributes five, six, two and seven through ten. If the attribute is zero no change is made. If there is a value it replaces the original value in the appropriate array. The interarrival time is recomputed using the equation stated in the UI section of this chapter based on the new values.

MESGEN. If the parameter passed to US is zero, it came from the clock network and background task MESGEN is to be executed. MESGEN consists of two sections, "BADNEWS" and "GOODNEWS". BADNEWS is processed for each PSN every 100 millisecond. GOODNEWS is processed for each PSN every 400 milliseconds.

BADNEWS checks to see if any links have become congested or have gone down. The row corresponding to the processing PSN is selected from the LD table. Each entry is examined. If an entry is either negative (connected but out of service) or infinite (1000), the next entry is examined. If the entry is positive but not infinite, each line in the corresponding link is examined (using the LINKS table and the connectivity fields from the type 001 user record). If either end of the line (or both ends) is negative, the entry in the LD table is set negative. If the entry is set negative, the corresponding entry in the NC table is set negative.

If the entry is not set negative, it is tested to see

if it indicates congestion. If it does not indicate congestion the number of packets in all line output queues for the corresponding link are added together and divided by the number of lines to obtain average output queue backup. Then all congestion levels for the lines in the link are added together and divided by the number of lines to obtain an average congestion level. If the average queue backup exceeds the average congestion level the entry is set congested.

When all entries have been checked for link-down and for output link congestion, the total number of packets backed up in the SCM input queues (Q-GERT node seven) belonging to the active PSN is compared to the total obtained by adding the saturation level for those SCMs together. If the backup exceeds the accumulated saturation level, all processing PSN LD table entries which are clear are set to congested.

GOODNEWS checks to see if any links have come up or become uncongested. The LD table row corresponding to the active PSN is selected and each entry is examined. If the entry is negative, the corresponding link is examined. If any line in the link has both ends (in the connectivity fields) positive the LD entry is turned positive but marked congested and the corresponding NC entry is set positive. After all entries are examined for link down, each entry is examined for congestion. If an entry indicates congestion,

the number of packets in all line output queues for the corresponding links are added together and divided by the number of lines to obtain an average output queue backup. Then all congestion levels for the lines in the link are added together to obtain an average congestion level for the link. If the average queue backup is less than the average congestion level the LD entry is marked clear (uncongested).

When all entries have been checked, the total number of packets backed up in the SCM input queues belonging to the processing PSN is compared to the total obtained by adding the saturation levels of those SCMs together. If the total input queue backup exceeds the saturation level all processing PSN LD entries which are clear are set congested.

If either BADNEWS or GOODNEWS changes entries in the LD table, UPDATE (in function UF) is called. This call to UPDATE initiates background tasks MINHOP, RTABLE and TTABLE. An LD table change also causes the generation of AUTODIN II status packets to each PSN other than the processing PSN, through Q-GERT node eight of one of the SCMs in the processing PSN.

If no change has been made by GOODNEWS or BADNEWS, UPDATE is still called but only RTABLE and TTABLE are initiated to update SROUTE, ROUTE and TROUTE.

SUBROUTINE UO (USER OUTPUT)

Subroutine UO consists of a single foreground task, PAKROUT.

PAKROUT. PAKROUT determines which activity branch should be selected out of Q-GERT node eight of the processing SCM.

Attribute three and four of the simulation transaction to be routed contain the source and destination SCMs. From this information the source and destination PSN are determined. The current SCM is placed in attribute eight. If the current node is equal to the destination node attribute six is set to minus one, marking the transaction for local delivery. If attribute one contains a one and attribute four is equal to the current SCM, the transaction is a status packet for this SCM. This causes UPDATE to be called and MINHOP, RTABLE and TTABLE to be initiated. Whether it is a status packet or not, if it is marked for local delivery attribute four is moved to attribute five. If attribute five does not equal the current SCM, it is set negative. Control is returned to the process which called PAKROUT.

If the destination node was not equal to the current node, an output line must be designated in attribute six. If the source node does not equal the current node the ROUTE table for the current node is used for routing. The entry corresponding to the destination node is selected. If the ROUTE table entry is zero the node cannot be reached. Attribute five is set to minus the current SCM, attribute

six is set to minus ten and control is returned to the process which called PAKROUT. The packet will be discarded. If the entry was not zero, find the corresponding link in the current node's LINKS table. The node that link corresponds to is the next node for routing. Find the next SCM by using the TROUTE table, the current node and the next node. If the next SCM is zero, discard the packet by setting attribute five to minus the current SCM, attribute six to minus ten and returning control to the process which called PAKROUT. If the next SCM was not zero the proper transmission delay is placed in attribute nine. If next SCM (outgoing SCM from this PSN) is equal to the current SCM, set attribute five to the current SCM, set attribute six to the TROUTE table line entry corresponding to the next SCM and return control to the process which called PAKROUT. If the next SCM is not equal to the current SCM set attribute five to minus the next SCM, set attribute six to the TROUTE line entry corresponding to next SCM and return control to the process which called PAKROUT.

If the source node does equal the current node the processing is the same as that specified in the last paragraph except that SROUTE is used instead of ROUTE to find the best link route. The ROUTE table allows only minimum hops, whereas the SROUTE table allows both minimum hops and minimum hops plus one.

Summary

The use of Q-GERT's duplication capability to provide many SCM's, and user input records to configure the line connections provides a convenient means to reconfigure the simulated network. The ability to vary the AUTODIN II network load and to disconnect and reconnect lines using user input records, allows testing the network under varying workload conditions. The extensive use of Q-GERT queue nodes (six for each SCM), however, causes the AUTODIN simulation to require extensive computer core memory (233000 words of CDC CYBER 750 memory) and the simulation requires at least 500 seconds of CPU time (for the entire job stream) to generate sufficient data to be useful. This limits the usability of the model.

CHAPTER 5

METHODS FOR ANALYZING THE SIMULATION RESULTS

Background

The computer program to simulate the AUTODIN II network in this thesis effort may generate a large quantity of output data. The problem arises as to how to reduce this data to a form which can be comfortably manipulated and, on the other hand, can provide a sufficient quantity and quality of data for a thorough analysis of the AUTODIN II network. The methods for analyzing the simulation results used by the authors are presented in this chapter, and illustrated by an example simulation run in Appendix A.

The Q-GERT modeling and analysis program provides a summary of the results of each simulation run in the output listings. For an explanation of this summary information, the reader should study Pritsker's text (Ref 8).

Within Q-GERT, there exists a unique capability to perform a transaction (or packet) trace. A trace provides a chronological listing of packets' events from their arrival at a source SCM, to a tandem PSN (if any) and to the final destination SCM. For each event of a packet (input queue, processor, output queue, trunk transmission, discard and

delivery), a record is printed (two output lines) with the following data name variables (Ref 8:245):

1. Q-GERT START NODE
2. Q-GERT ENDNODE
3. ACTIVITY NUMBER
4. START TIME (milliseconds)
5. END TIME (milliseconds)
6. ACTIVITY NUMBER
7. MARK TIME (milliseconds)
8. TRANSACTION NUMBER
9. PACKET ATTRIBUTES (10):
    1. Packet type: status packet, short data packet or long data packet.
    2. Priority level: system packet, CATI, CATII, CATIII ot CATIV.
    3. Source SCM
    4. Destination SCM
    5. Departure SCM (SCM from which the packet will depart this node).
    6. Departure line (line or trunk from which the packet will leave the departure SCM to reach the next SCM).
    7. Processor Time
    8. SCM the packet is presently visiting
    9. Time, in milliseconds, for the packet to be loaded into the output line or transmitted to the destination SCM.
    10. Code to identify the packets source workload: General or Specific.

In only a few seconds of the simulation run, the total packet activities may generate thousands of records. The trace records are produced in chronological order, that is, a single packet's activities are not grouped together. Packet activities are interspersed among each other. Manually searching through the trace records, to organize and summarize packet activities would be an enormous task. For example, an 8 PSN, 16 SCM simulation of AUTODIN II, at 20% of network capacity, for 45 seconds, produced over 60,000 records.

To provide output data for an effective and efficient analysis of the AUTODIN II system, a series of three programs, STRIP, SORTMRG and SUMARIZ, were developed to provide a single data file for analysis. This data file was then processed by the Statistical Package for Social Science (SPSS) for analysis. The following sections will give a description of the three data reduction programs and the SPSS analysis program.

STRIP

Only a portion of the trace records, generated for each packet, are necessary to describe the important events of the packet while traversing through the network. STRIP is a FORTRAN program which extracts pertinent trace data to show the historical events of packets from the time they enter a source PSN to the time they leave a destination PSN. A listing of the program is contained in Appendix B.

The output from the Q-GERT simulation model, including the

trace data, is stored on a disk file, TAPE1. STRIP selects records of Q-GERT activities beginning at the following Q-GERT nodes:

1. Q-GERT node 10, the beginning of the activity to discard packets or to send packets to another SCM within the same PSN.
2. Q-GERT nodes 16 through 20, the beginning of the activity to transmit packets to another SCM within another PSN.
3. Q-GERT node 21, the beginning of the activity to deliver packets to a host or terminal.

These records contain sufficient information to describe or compute the pertinent packet events, for example, time spent in the input queues and the output queues.

Records containing information of the AUTODIN II network configuration (input by the user) are also selected to provide a cross reference between PSN and SCM required in the SUMARIZ program. The Q-GERT simulation summary is written on the output listing and the selected records from the trace data are written to the TAPE2 disc file. TAPE1 is returned.

SORTMRG

Although TAPE2 is a significant reduction in quantity of data from TAPE1, the records are not organized in packet groups. The SORTMRG is a CDC library program to sort and merge data files. In this process, TAPE2 is sorted and the resulting output stored on the TAPE3 disk file. A listing of the SORTMRG cards is contained in Appendix B. The records

are sorted first by packet activities and then by MARKTIME (the time the packet entered the system). TAPE2 is returned.

SUMARIZ

The SUMARIZ program was developed to reduce the packet activity records (on TAPE3) into a single record per packet that could easily be interpreted by the Statistical Package for Social Sciences (SPSS) or other statistical programs. A source listing of SUMARIZ is given in Appendix B.

SUMARIZ selects all of a packet's activity records, until the packet is delivered or discarded, and builds a single record containing:

1. General information about the type of packet and the time in the AUTODIN II network.
2. Detailed information about the packet's activities within each node of the network.

Fields for all eight PSNs were included in each record (even though a packet did not visit all eight PSNs), to provide easy processing by SPSS. The output records are stored on the TAPE4 disk file. The output records could easily be punched on cards, written to magnetic tape or placed on a permanent file, by appropriately routing or copying TAPE4 in the job control stream. The output records are in chronological sequence and consist of five lines (or cards). The format of the output records are given in the User's Manual (Appendix A). TAPE 3 is returned.

STATISTICAL PACKAGE FOR SOCIAL SCIENCE (SPSS)

SPSS was selected over other statistical analysis packages because of the thorough documentation available to AFIT students and its prior use by the authors of this thesis. Users may prefer to use other statistical packages or to develop their own analysis program. A listing of an example of the SPSS program cards is given in Appendix B. For an explanation of the SPSS cards, the reader should study Ref 9 and 10. In the example listing, only the fields on card 1 are used in a total data analysis. While the fields on cards 2 through 5 require specific selection or editing. These fields (on cards 2 through 5) are provided to collect data for each PSN. However, since a packet visits only a few of the PSNs in the network, the PSNs not visited must be omitted from the analysis of a single PSN. In SPSS this is done by selecting only those PSNs which were actually visited. This is discussed in more detail in the User's Manual (Ref Appendix A).

This chapter briefly described the methods the authors used to reduce the Q-GERT data into a form for use by a computer library statistical analysis package. These methods proved useful to the authors, but users may desire to incorporate their own methods of analyzing the Q-GERT data. More detailed information on the structures of the output files (produced from the Q-GERT simulation, STRIP, SORTMRG and SUMARIZ programs) is provided in the User's Manual (Appendix A).

CHAPTER 6

VALIDATION AND VERIFICATION

Introduction

Before a simulation model is used to gather data on new situations, validation and verification should be performed. Validation insures that the model performs to the expectations of the modelers. Verification establishes that the model produces results close enough to those obtained from the process modeled that when the model is applied to scenarios for which there is no previous experience, the results obtained will be useful.

Validation of the AUTODIN II model was not difficult. Verification on the other hand, proved very difficult as AUTODIN II is not functional and actual results were not available for comparison. An effort, however, was made to both validate and verify the model.

Validation

The Q-GERT simulation language provides a trace capability. This trace capability allows the modeler to follow a Q-GERT simulation transaction from it's source to it's

termination. A line of information is printed when a transaction arrives at a Q-GERT node and when it is leaving a node. When a transaction leaves a node the information printed includes the node it is leaving, the destination node it is going to, the time it leaves the node, the time it will arrive at the next node, the number of the activity it is traversing, the time it was last marked (for AUTODIN II, the origination time for the transaction), the transaction identification number, and the values assigned to it's attributes. When the transaction arrives at a node the trace print line includes all the same types of information, except for the identification number of the node it left. This field is replaced by three asterisks, to identify the print line as presenting information concerning an arrival at a node. Table XV presents examples of both types of print lines as they appear in an AUTODIN II simulation trace.

The authors insured that the simulation performed according to expectations by establishing scenarios for the simulation, then by tracing simulated AUTODIN II packets through the simulated AUTODIN II network to insure they traveled the expected routes, with the expected delays, through the network. The scenarios included normal traffic, congested transmission lines, dropped lines, dropped links, and dropped Packet Switching Nodes (PSNs). Several anomalies were discovered and corrected during validation, including lines not dropping properly, excessive arrivals

TABLE XV

Q-GERT TRACE DATA

Start Node	End Node	Start Time	End Time	Activity Number	Mark Time	Trans Number	Attributes				
7-4	8-4	135.45	140.45	832	135.45	10	2.00	20.00	4.00	12.00	-3.00
							2.00	0.00	4.00	20.60	1.00
***	8-5		135.52	826	103.46	1	2.00	20.00	4.00	8.00	5.00
							4.00	0.00	5.00	24.02	1.00
8-5	14-5	135.52	135.52	0	103.46	1	2.00	20.00	4.00	8.00	5.00
							4.00	0.00	5.00	24.02	1.00
***	14-5		135.52	0	103.46	1	2.00	20.00	4.00	8.00	5.00
							4.00	0.00	5.00	24.02	1.00
14-5	19-5	135.52	159.54	822	103.46	1	2.00	20.00	4.00	8.00	5.00
							4.00	0.00	5.00	24.02	1.00
***	8-3		136.09	838	127.82	36	2.00	20.00	3.00	12.00	3.00
							2.00	0.00	3.00	33.08	1.00
8-3	12-3	136.09	136.09	0	127.82	36	2.00	20.00	3.00	12.00	3.00
							2.00	0.00	3.00	33.08	1.00
***	12-3		136.09	0	127.82	36	2.00	20.00	3.00	12.00	3.00
							2.00	0.00	3.00	33.08	1.00
12-3	17-3	136.09	169.17	836	127.82	36	2.00	20.00	3.00	12.00	3.00
							2.00	0.00	3.00	33.08	1.00
***	2		137.50	0	125.00	5	1.00	0.00	3.00	0.00	0.00
							0.00	0.00	0.00	0.00	0.00
2	2	137.50	150.00	0	137.50	5	1.00	0.00	4.00	0.00	0.00
							0.00	0.00	0.00	0.00	0.00

at the discard Q-GERT termination nodes, and status packets not performing UPDATE subroutine at their destination node.

The Q-GERT trace capability proved to be a useful tool during validation.

Verification

Since AUTODIN II is not functional and since the AUTODIN II configuration is unique, verification using actual data for comparison was not possible. Since verification against actual data was not possible, the next option was to use the data in the thesis by Capt. Whittenton to compare results from two different simulations of AUTODIN II constructed from similar base assumptions. Capt. Whittenton used 20% of the maximum packet traffic capability projected for AUTODIN II as the normal traffic load for his simulation, and introduced one second increases (spikes) to 60%, 70%, 80%, and 90% of maximum projected capacity to study their effect on the network. The AUTODIN II model was configured to match Capt. Whittenton's model, including traffic load, and a 70% one second spike was introduced in the model's traffic. The resulting data is compared to Capt. Whittenton's data in Figure 18. The vertical scale of the graph is in milliseconds and represents the average delay time accumulated by AUTODIN II packets during each one second time period (horizontal axis scale). The time is accumulated as a transaction enters the network for graphing purposes.

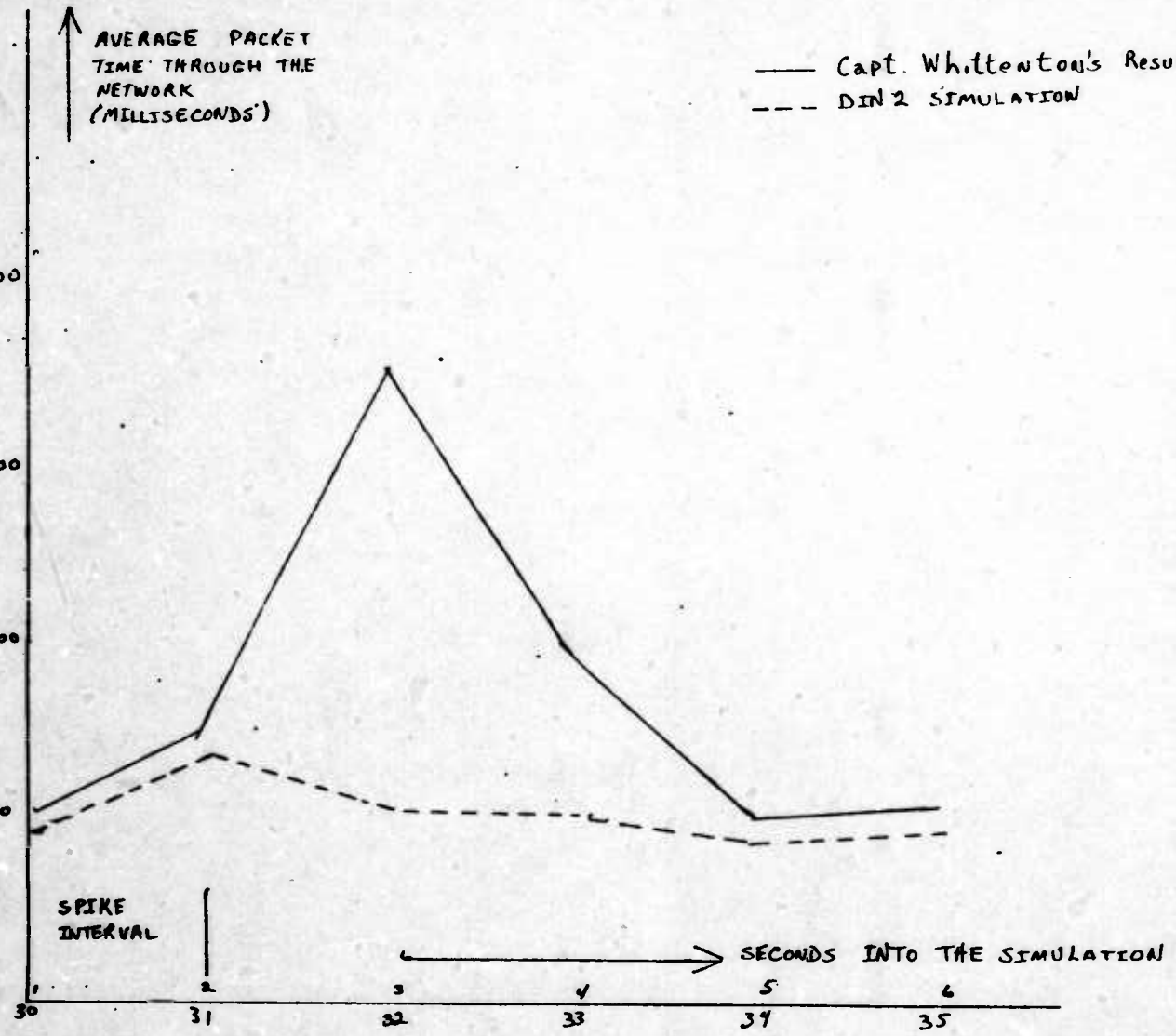


Figure 18 - 70% Spike Comparison

It appears that there is no similarity between the spike in Capt. Whittenton's data (the solid line, Ref 4:F267-271) and the AUTODIN II model data (the dashed line).

The authors examined Capt. Whittenton's thesis, but could not find anything that explained the differences. Capt. Whittenton, during a telephone conversation concerning the possible differences in models, stated that his model generated status packets on a 100 millisecond period basis, whether there was a change in network status or not. The AUTODIN II model only generates status packets when network conditions change. Although Capt. Whittenton's thesis did not confirm his verbal statement, a simple change was made to the AUTODIN II Subroutine US (User Subroutine) to generate status packets on the same periodic basis, rather than on a condition change only basis. Thus, status packets were generated into the network every 100 milliseconds from each Packet Switching Node. This increased traffic by 49 packets each 100 millisecond period. The results were then graphed two ways. First the delay time for each packet was accumulated when the packet entered the network. This produced Figure 19. The path of the dashed line (AUTODIN II results) is clearly much closer to the solid line (Whittenton's results). Since the authors did not know when Whittenton accumulated the delay times, another graph was produced with delay times accumulated when packets left the network. This produced Figure 20, in which the paths of both lines appear

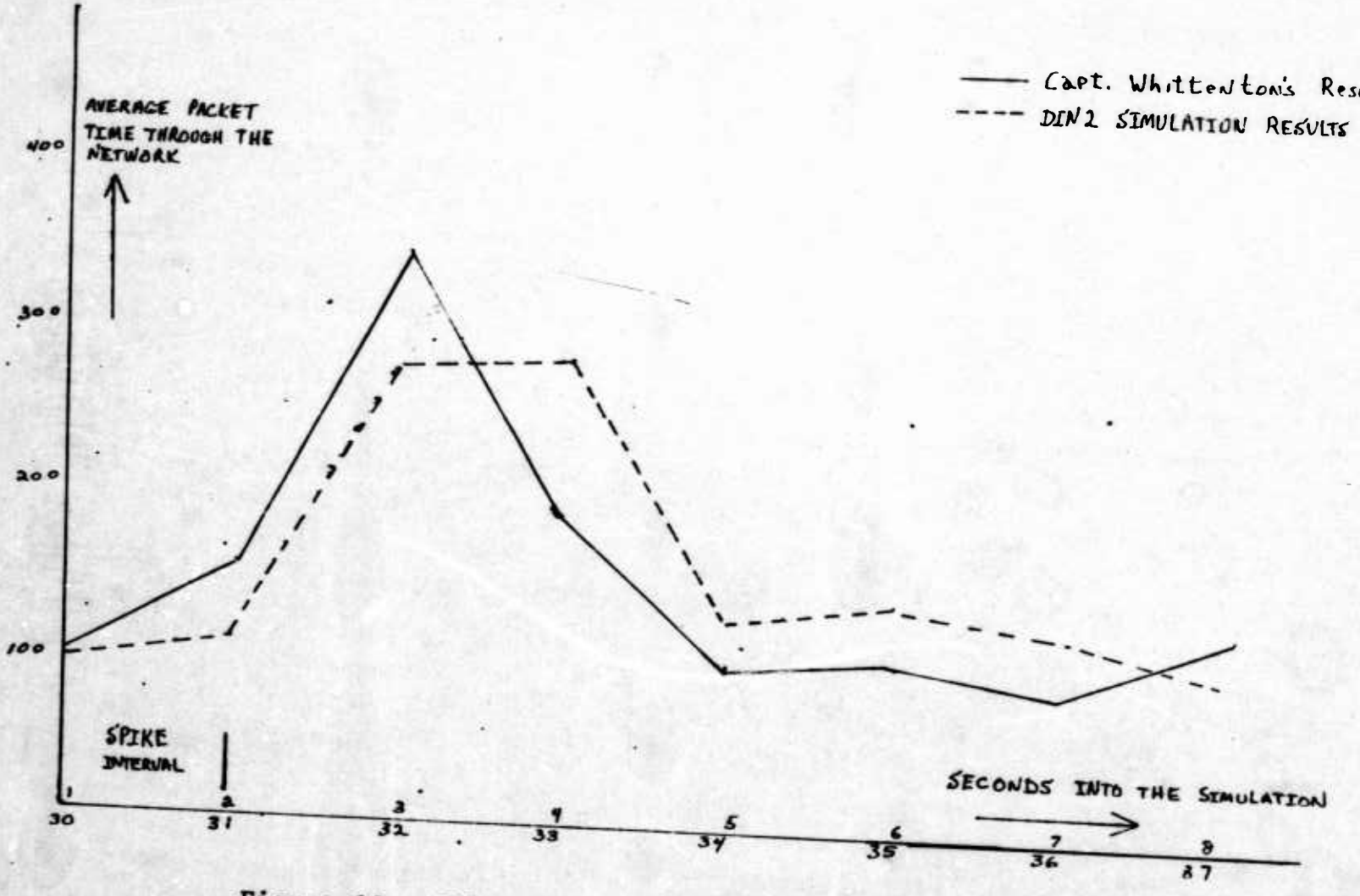


Figure 19 - 70% Spike Comparison With Forced Status Packets

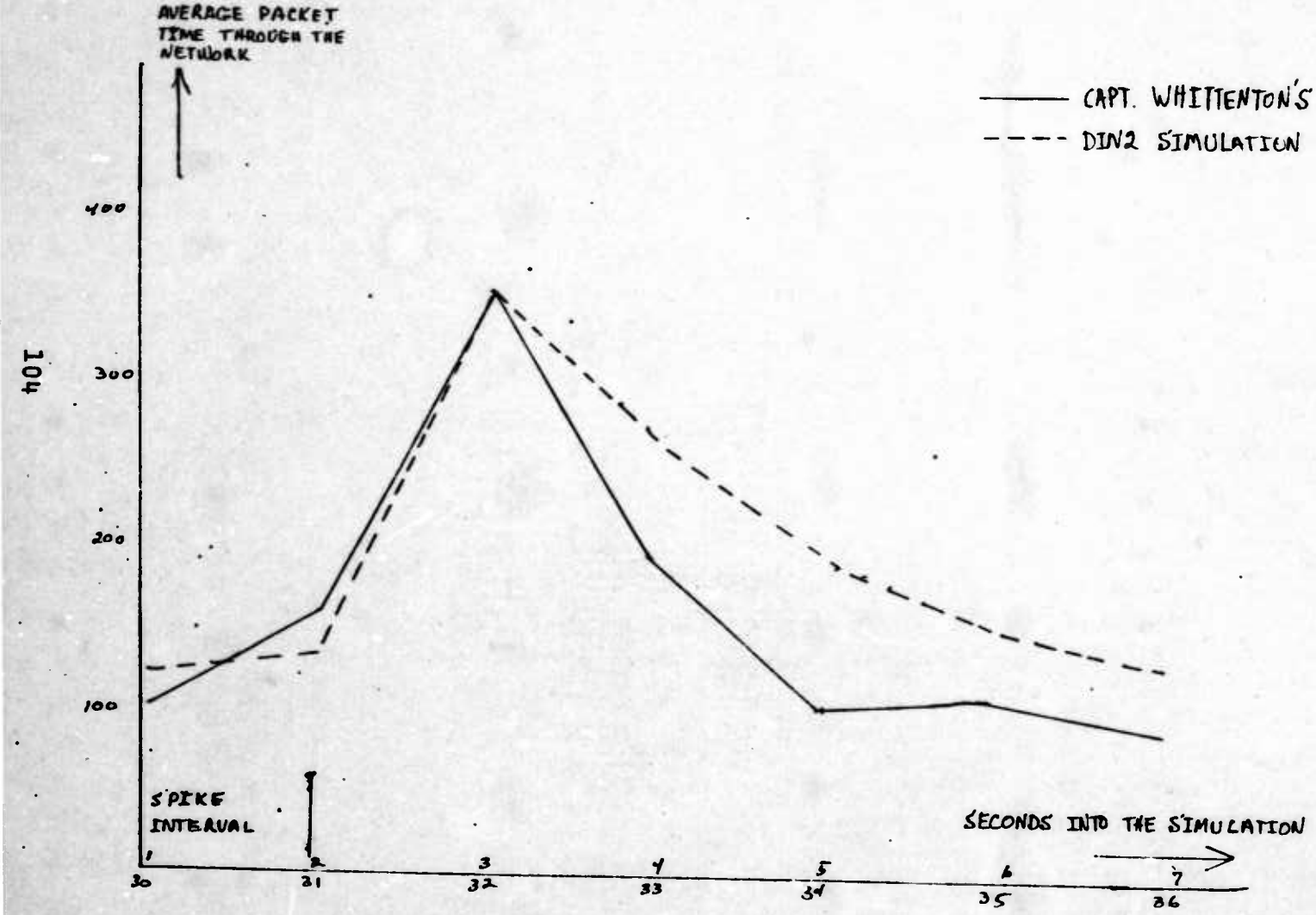


Figure 20 - 70% Spike Comparison as of Packet Termination

even closer together. The differences between the dashed lines in Figures 19 and 20 are easily explained. In the first of the two graphs, the delays were accumulated as the packets arrival into the network. This would cause both short and long time delays to appear in the same interval, with some intermixing of both, lowering and spreading the spike. In the second graph the delays were accumulated when the packets left the network. Short delays, therefore, would fall close to the packets arrival time in the network, while long delays would appear in more distant intervals, peaking the spike sharply and smoothing the drop off, since the long delays would tend to cluster.

The conclusion drawn from Figure 20 was that there was too much similarity for coincidence and a reasonable assumption was that the two different models produced similar results.

One last area should be discussed for rigor. A simulation run was made with no dynamic changes in the configuration or traffic load of the network to investigate the period of time required for the simulation to stabilize. The mean packet delay times over 1000 millisecond intervals, for 45 such intervals, were visually inspected. There did not appear to be any difference in the variability of any of the intervals. Another inspection of 250 millisecond intervals over 180 periods showed similar results, except at the beginning intervals (this was attributed to "start up" of the

simulation run). A decision was made to use only nine seconds data, eliminating the selection of packets generated in the first 1000 milliseconds, for statistical analysis.

To test the validity of this decision and the visual inspection of the intervals, a statistical multiple comparison test was made on the data (Ref 12:206). The first three interval mean delay times were tested against the remaining total interval mean delay times. This was accomplished by selecting constants, C_i, such that $C_i=0$. The selected C_i and the corresponding hypothesis test (where u_i is defined to be the expected mean of the ith interval and N is the total intervals, 45) are:

1. $C_1=-1$, $C_i=1/(N-1)$ for $i=2,\cdots, N$
   H_0: $u_1=u_1$ where $u_1=\sum_{i=2}^{N} u_i/(N-1)$

2. $C_1=0$, $C_2=-1$, $C_i=1/(N-2)$ for $i=3,\cdots, N$
   H_0: $u_2=u_2$ where $u_2=\sum_{i=3}^{N} u_i/(N-2)$

3. $C_1=C_2=0$, $C_3=-1$, $C_i=1/(N-3)$ for $i=4,\cdots, N$
   H_0: $u_3=u_3$ where $u_3=\sum_{i=4}^{N} u_i/(N-3)$

Each of these hypothesis were tested at the 95% confidence level. Each hypothesis was accepted well within the 95% confidence level. Therefore cimulations at the 20% traffic level may be run with only a 1000 millisecond "warm-up" with a 95% confidence that the mean delay times are stablized.

Computations

95% confidence interval:

$$\sum C_i u_i = \sum C_i x_i = \sqrt{F_{.05}}\, Sp \sqrt{\frac{(r-1)(\ C_i^2)}{n}}$$

$n \approx 130$ n is actually not equal, i.e., this is an approximate test.

$r=45$ # of means

$F_{.05}=44, \infty = 1.4$

$Sp^2 = 1/r \sum S_i^2 = 7790$

1. $\sum C_i x_i = 108-94.4 = 13.6$

under H_o the $C_i = \pm 1.4(88.3) \sqrt{\frac{44(1.01)}{130}}$

$= \pm 72.3$

13.6 is within $\pm$ 72.3

2. $\sum C_i x_i = 99.4-94.3 = 5.1$

under H_o the $C_i = \pm 1.4(88.3) \sqrt{\frac{43(1.01)}{130}}$

$= \pm 71.4$

5.1 is within $\pm$ 71.4

3. $\sum C_i x_i = 81.8-94.6 = -12.8$

under H_o the $C_i = \pm 1.4(88.3) \sqrt{\frac{42(1.01)}{130}}$

$= \pm 70.6$

-12.8 is within $\pm$ 70.6

Summary

The AUTODIN II model was both validated and verified.

The Q-GERT trace mechanism was used to validate that the model performed as planned.

The first option for verification was to compare the simulation results to actual data. Since AUTODIN II was (and is) not yet operational, actual data was not available. The second option was to verify the model against Capt. Whittenton's model. Verification was completed by comparison to the results from Capt. Whittenton's model.

The remaining chapters (7 and 8) will detail analysis of data from the AUTODIN II model.

CHAPTER 7

RESULTS AND ANALYSIS

Introduction

The purpose of this chapter, and Chapter Eight, is to perform an analysis of a potential problem or management concern about the proposed AUTODIN II System.

Before proceeding, the author would like to clarify the following point: the analyses presented in Chapters Seven and Eight do not make full use of all the capabilities which were built into the simulation model. The analyses were performed for specific problems. As mentioned in the first chapter of this thesis, one of the primary goals of this effort was to develop a simulation model for future users. With this objective, the authors developed the model to allow user flexibility in defining the AUTODIN II network configuration and a variety of network up/down conditions, traffic scenarios, message types and workloads. Therefore, the reader should not consider the analysis in this chapter as a restriction or an application of the full capabilities of the simulation model.

The problem for this analysis was derived from a letter to the author from the DCA sponsor, Captain Wade Nielson. The following excerpt from that letter provided the basis for the analysis.

> SCENARIO: The network is operating normally (ie, in a quiescent mode) and something happens to cause an increase in CATI traffic being offered to the network (maybe CATI increases from its nominal of 1% to 3-5% of total network traffic, or higher - use your own judgment).
>
> DETERMINE: What are the effects on CATII, III and IV traffic? For example, what delays do they experience? What delays do the CATI packets experience? (If CATI traffic does not get delayed much, try increasing the % of CATI traffic). Is the routing algorithm breaking down (you might not be able to tell this with the model you are using)? If your model identifies flow control, are queues building up too high? Is there an increase in network traffic (eg, network status messages, overhead traffic, etc)?
>
> SCENARIO: Using the above example, what happens if, in the middle of this increase of CATI traffic, some links or nodes fail (get bombed or otherwise destroyed). Now what happens to the rest of the packets? Is it any different than the above scenario?

The author's approach to the above letter is to divide the problem into two parts: the effects of increasing CATI traffic and the effects of node (PSN) failure. Each of these parts is further subdivided into the design of the experiment or simulations and the analysis of the data.

Design of CATI Traffic Increase

From the first scenario above, an analysis of the results of increasing CATI traffic was developed. Although,

The primary concern was the increase of CATI traffic with a normal traffic level, it appeared to the author that the network traffic may have a relationship to the percent of CATI traffic. That is, a significant increase in CATI traffic may be caused by a catastrophic event, say, a war readiness state or build up. This would also cause an increase in the total network traffic. (The effects of the AUTODIN II system, in a war readiness build up or in a state of war, including links and nodes being brought down, is a concern to the sponsor). Therefore, the increased level of CATI messages was tested for various levels of nominal traffic load. The levels of CATI messages and network traffic were chosen by the author for comparative analysis (see Table XVI), and they are not based on any known data source. Each of the second and third levels were step increases from the previous level. The network data used in the simulation was obtained from Western Union Telegraph Company documents (Ref 5 and 11). In the AUTODIN II simulation, by Captain H. Whittenton (Ref 4), a similar data base is used from the same source. The data is described in more detail as an example in the User's Manual (Appendix A), and will not be presented here.

TABLE XVI

PARAMETERS FOR COMPARATIVE ANALYSIS

Network Traffic Levels	Percent of CATI Messages		
Normal Traffic	1%	5%	10%

30% Increase	1%	5%	10%
50% Increase	1%	5%	10%

Analysis of CATI Traffic Increase

A summary of the simulation output is displayed in Table XVII. Three levels of priority are considered: Priority 1 is assigned to network status packets, priority 2 is assigned to CATI data packets, and priority 3 is assigned to CATII, III and IV data packets. The elements of the table are the average delay times (in milliseconds) of the packets and, in parenthesis, the number of packets in the traffic interval.

TABLE XVII

SIMULATION DATA SUMMARY

Network Traffic	Network Priority	Network CATI Traffic 1%	5%	10%
NORMAL	STATUS			
	CATI		62(6)	125(16)
	CATII,III,IV	103(54)	92(113)	106(114)
30% INCREASE	STATUS			
	CATI		35(6)	91(15)
	CATII,III,IV	97(176)	117(149)	112(173)
50% INCREASE	STATUS			
	CATI		123(12)	79(24)
	CATII,III,IV	103(202)	114(196)	105(165)

An analysis of the network effects of CATI percent levels of traffic was performed on the CATII, III and IV delay times,

because of 1% CATI level did not generate sufficient data for an analysis, and the packet priorities are not statistically independent. With the CATI percent levels iterated for various traffic levels, Table XVII takes the form of a Randomized Complete Blobk (RCB) design (Ref 12:363). The traffic levels are defined as blocks, and the CATI levels as treatments (or the variable under test). Performing an Analysis of Variance (ANOVA) on the raw data provided the results in Table XVIII.

TABLE XVIII

ANOVA RESULTS

	Sum of Squares	DF	Mean Square	F-Test
Traffic Levels	15,444	2	7,722	.63
CATI Levels	21,910	2	10,955	.89
Interaction	29,897	4	7,474	.65
Explained	68,052	8	8,506	
Residual	11,068,712	907	12,203	

From the F-test values, neither the CATI levels nor the traffic levels were significant (above 80%). Since the CATI packets make up such a small amount of the total packets, it is obvious that the increase to 5% and 10% would not have much effect on packet delay times, unless the network was in a delicate balance or near a saturation level. This statement is backed up by the fact that no network status packets were initiated at any of the nodes, that is, no saturation or con-

gestion thresholds were exceeded. Since there were no significant differences in packet delay times for either the CATI or traffic levels, the author's conclusion is that the network is capable of handling CATI increases with no significant change in packet delay times with normal traffic, and with traffic levels up to 150% of normal.

A follow-up analysis of the above procedure was performed with another seed. Of specific interest were two CATI means in Table XVII: the 10% CATI level with normal traffic, and the 5% CATI level with 50% traffic increase. Both means were high in comparison with the means of the lower priority packets. Using another seed, these two means were reduced. However, the ANOVA for this case depicted similar results of non significance for the CATI and traffic levels. Due to program turnaround times and thesis time constraints, analyses with 5, or even 10 seconds (rather than one second as used in the above test), were not performed.

Although no status packets were generated with normal traffic, the average packet delay time in the output queues of each node will become important later in this section. Therefore, for comparison, Table XIX is defined. The first column contains the quantity of packets entering the node (either as a source node, tandem node or destination node). The next column contains the average time spent in the output queue for each node. Observing the values in Table XIX indicates that node 1 (Albany PSN) and node 8 (Tinker PSN) may be the first

congestion problems in a heavy network traffic.

To attempt to ascertain some limits on when the CATI percent levels make a significant difference in packet delays, the above analysis procedure was performed on a simulation with the network traffic level at 40% saturation (or 40% of system capacity), instead of the normal level of 20% saturation. Simulations with higher CATI traffic were also considered. However, the author did not feel that extreme levels of CATI traffic, say 20% to 40%, would effect the packet delay times, and therefore did not simulate higher traffic levels. However, this hypothesis should be simulated and tested before any final conclusions.

TABLE XIX

AVERAGE OUTPUT QUEUE STATISTICS

Node		Number of Packets	Time In Milliseconds
1	(Albany)	362	23.6
2	(Andrews)	683	16.2
3	(Fort Dietrick)	172	8.7
4	(Gentile)	363	7.8
5	(Hancock)	226	13.6
6	(McCLELLAN)	313	5.7
7	(Norton)	221	10.5
8	(Tinker)	620	22.2

Table XX displays a summary of the packet delays with a

traffic level of 40% network saturation with associated 30% and 50% increases, and CATI levels of 1%, 5% and 10%.

TABLE XX

SIMULATION DATA SUMMARY

Network Traffic	Network Priority	Percent CATI Traffic 1%	5%	10%
40% Saturation	STATUS	80(14)	75(7)	98(28)
	CATI	46(4)	63(18)	95(35)
	CATII,III,IV	150(243)	109(237)	168(234)
30% INCREASE	STATUS			75(42)
	CATI	89(2)	88(19)	112(31)
	CATII,III,IV	145(354)	181(326)	182(343)
50% INCREASE	STATUS	105(49)	88(49)	92(84)
	CATI	35(1)	103(16)	111(42)
	CATII,III,IV	228(404)	213(386)	217(352)

The quantity of network status packets generated in the simulation indicates that the network is periodically becoming saturated.

A comparative analysis between the normal traffic and the 40% saturation traffic simulations was developed. The most interesting comparisons are shown in Table XXI.

TABLE XXI

SIMULATION COMPARISON

	Normal Traffic	40% Saturation Traffic
Qty of Status Packets	0	273

Average Packet Delay	105	171
CATI Delay	93	99
CATII,III,IV Delay	106	182

The comparisons in Table XXI were significant at or above the 80% level using a Student's T-test of two means and assuming equal variances. The generation of the network status messages increased the system's packet traffic by 8.9%. Note that, although statistically significant, the CATI packets maintained approximately the same delay time. The increased traffic (200%) did not effect the CATI packet delay times very much. However, the low priority packets, CATII,CATIII and CATIV, were significantly effected. The delay times increased by 72%. An ANOVA was performed on the raw data from the 40% saturation traffic simulation with the results shown in Table XXII.

TABLE XXII

ANOVA RESULTS

	Sum of Squares	DF	Mean Square	F-Test
Traffic Levels	1,527,756	2	763,878	21.3
CATI Levels	75,144	2	37,572	1.1
Interaction	407,029	4	101,757	2.8
Explained	1,992,641	8	249,080	
Residual	66,189,634	1850	35,778	

Again, at the 80% confidence level, CATI precent levels

do not make a significant difference in delay times (the F-test of 1.1 is only about 65%). However, the traffic levels are very significant (in the ANOVA of the normal traffic test, traffic levels were not significant). This implies that at 40% saturation, any change in traffic will make a significant change in packet delay times. And as noted above, this change is primarily due to the low priority packets (CATII, CATIII and CATIV). The conclusion can be made that the network has become significantly sensitive to traffic changes at the 40% saturation. The increase in percentage of CATI packets does not make a significant change in total packet delay times (but is approaching a level of significance).

A slight deviation from the sponsor's problem has proven interesting. Table XXIII shows the average output queue statistics for the 40% saturation simulation. Comparison of Tables XIX and XXIII indicate the same problem nodes, that is, node 1 (Albany PSN) and node 8 (Tinker PSN) will probably be the first congested nodes in the network (because of their higher waiting time). However, node 2 (Andrews PSN) was found to be of larger concern; in particular, SCM 3 of that node. Of all status packets generated in the network, 38% came from one Andrews SCM. Table XXIV shows the details of the status packets generated by the network in this simulation. The table is divided into the 40% saturation, the 30% increase (30% increase in traffic above the 40% satura-

tion level) and 50% increase. The elements of the table are the times (in milliseconds) into the simulation when the congestion level was found (FTICK or STICK). The times when the output queues fell below the congestion level are not shown, but congestion remained for as long as 1500 milliseconds at one of the SCMs. SCM 3 of the Andrews PSN was the first SCM to become congested, and was congested more than any of the other SCMs.

TABLE XXIII

AVERAGE OUTPUT QUEUE STATISTICS

Node		Number of Packets	Time in Milliseconds
1	(Albany)	856	78.3
2	(Andrews)	1434	51.5
3	(Fort Dietrick)	519	21.7
4	(Gentile)	874	34.2
5	(Hancock)	571	54.6
6	(McClellan)	625	14.7
7	(Norton)	527	36.8
8	(Tinkers)	1304	56.6

The SCM at Andrews should be investigated. Captain John Whittenton, in his thesis (Ref 4:166), stated a problem in the Andrews-Hancock link and the Albany-Andrews link. This analysis indicates the problem lies in the same general location but, specifically in the Andrews SCM. The author

TABLE XXIV

CONGESTED SCMs IN THE
40% SATURATION RUN

	Albany PSN SCM 1	Andrews PSN SCM 3	Gentile PSN SCM 8	Hancock PSN SCM 10	Tinker PSN SCM 14
40% Saturation		412			
		1612			
30% Increase	6000	3512			3687
	6500	6112			6887
50% Increase	7200	7412	8237	7550	7687
	9000	8412		9950	8687
	9500	9212			9887
		9612			
Total Packets	5	8	1	2	5

recommends more data be gathered from additional simulation runs and analyzed to verify what appears to be the weakest link in the network. Also, simulations should be run with an additional SCM in the Andrews node, or other possible configuration changes.

Design of PSN Failures

The problem discussed in this portion of Chapter Seven was derived from the second scenario provided in the letter, previously quoted, from Captain Nielson. To develop data for analysis of this problem, the following simulations were designed:

Network Configuration	CATI Level
All PSNs Up	10%
Fort Dietrick PSN Down	10%

The CATI percent was kept at the high of 10%, based on the previous analysis. Fort Dietrick PSN was selected to bring down for the following reasons:

1. Fort Dietrick generates only about 5.1 million characters per hour (3.09 packets per second). This is only 2.2% of the total system traffic. Therefore, bringing Fort Dietrick down would not change the network traffic very much, that is, a "worst case" situation is simulated.

2. The Fort Dietrick PSN contains three SCMs of the sixteen SCMs in the total network. The loss of Fort Dietrick (or Tinker, which also contains 3 SCMs) would cause a greater loss of SCMs than any

of the other PSNs. Note, however, that the loss of three SCMs may not be as dramatic as the loss of a PSN with only one SCM which may be of greater importance in efficiently routing packets to destination PSNs. This analysis may be considered for a future study.

3. Graphically, Fort Dietrick appears to be the "hub" of the network (see Figure 3). However, this is only a perceptive view and may be of little value if the figure were redrawn.

The Tinker PSN also contains three SCMs. However, the Tinker PSN was not chosen for the simulated PSN failure because of its high traffic input (22.8% of the total system traffic). That is, if the Tinker PSN failed, the network would have 23% less packets.

Analysis of PSN Failures

Two simulations were run for this analysis. The first simulation was with all PSNs functioning normally, CATI traffic making up 10% of the total network traffic and a simulation time of 6500 milliseconds. The second simulation was the same as the first, except, at 3000 milliseconds into the simulation, the Fort Dietrick PSN was brought down for the remainder of the simulation. All links of the Fort Dietrick PSN were down and the input traffic was zero. A study of the raw data showed that two PSNs (Hancock and Tinker), each at their own FTICK time (check for bad news), discovered the Fort Dietrick PSN down and sent information status packets to the other PSNs. The packets left in the network (and all future packets) that were destined for the Fort Dietrick PSN, were

discarded.

Table XXV shows a comparison of the packet delay times between the completely connected network and the network with the Fort Dietrick PSN down. The entries of the table are the average delay times (in milliseconds) of all packets and, in parentheses, the number of packets in the traffic interval. The packets are further broken down into CATI Packets and CATII, III, IV Packets (status packets generated when node 3 went down are not shown). The elements of the table are the average time delays (in milliseconds) and the values in parentheses are the quantity of packets for each case. Node 3 Down (column 1 in Table XXV) statistics exclude the data prior to the PSN going down. A statistical T-test for comparison between two means (assuming equal variances) was performed for each row of Table XXV. The tests showed a significant difference between the two simulations for row 1 (all packets) and row 3 (CATII,III and IV packets) at the 90% confidence level. One of the reasons the CATI packets T-test statistic was smaller than the other row comparisons is due to lower sample sizes. The CATI packets, between the two simulation runs, die show significance at the 77% confidence level.

TABLE XXV

SIMULATION DATA SUMMARY

All Nodes Up Node 3 (Fort Dietrick) Down

All Packets	96(713)	106(552)
CATI Packets	87(72)	97(61)
CATII,III,IV Packets	97(641)	110(477)

In conclusion, bringing down the "worst case" node, as observed by the author of this chapter, caused a significant increase in packet delay times. However, the increase in packet delay times are small enough that the DCA specifications are not exceeded (Ref 6:43). Therefore, the network can be considered as functioning at an acceptable level. The major loss, in the network, is not in the increased packet delay times, but in the loss of messages originating at, and destined to, the downed node (the Fort Dietrick PSN in the above simulation).

Summary

This chapter has attempted to answer some of the problems of concern to DCA. The approach to the problems was two fold. First, design network simulations with various traffic priorities and configuration changes. Second, statistically analyze and compare the results.

For the traffic priorities, various levels of CATI traffic were simulated and, under normal network traffic, there was no significant difference in packet delay times. When the traffic level was increased to 40% saturation, CATI traffic increases made a significant increase in packet

delay times. However, the delay times were within the DCA specifications. In analyzing the simulation data, it was observed that one of the SCMs in the Andrews PSN was continuously becoming congested before any of the other PSNs. If the AUTODIN II network traffic is projected to increase (specifically, the traffic through the Albany PSN), then the Albany PSN may be a major bottleneck in the network. Further analysis was recommended.

To analyze the effects of network configuration changes or failures, two simulations were run. The first simulation contained normal traffic levels with all PSNs connected. For the second simulation, the Fort Dietrick PSN was brought down during the simulation run. The two simulations were compared and, although a significant difference in delay times was observed, the network was functioning at an acceptable level. That is, the packet delay times were within the DCA specifications.

CHAPTER 8

RESULTS OF NETWORK LOADING

Basis For Analysis

The projected normal loading for the AUTODIN II is 20% of total network capacity. The analysis in Chapter 7 demonstrates that at that loading and up to 200% of that loading (or 40% of network capacity), the AUTODIN II network operates will within required time constraints. However, the acquisition delays encountered in past system purchases have resulted in load capacities exceeding expectations by wide margins when systems were finally in place. Therefore the analysis in this chapter will explore the effects of workloads which are 60, 70, and 80% of total network capacity.

One area that will be investigated is similar to a portion of the analysis in Chapter 7. The scenario is: assume NORAD has detected the possibility of a missile attack. We postulate that the network would experience a surge of high priority traffic originating from one site. Impose this postulation on a network already loaded to 70% of system capacity. Are the effects significant enough to

cause excessive delays for this or other traffic in the network?

Another aspect that will be explored is the effect of increasing normal workload from 60 to 70 to 80%. Figures 18, 19, and 20 in Chapter 6 demonstrated that transmitting status packets on a pericdic basis can sharply increase the sensitivity of the network to changes in traffic. Does increasing workload levels cause status packet generation to approach periodic generation? Also, how does increasing network load effect transmission times of Category I and lower priority packets? These questions will be answered within the text of this chapter.

NORAD Scenario

The NORAD scenario was implemented by designating all traffic from Switch Control Module 16 as special traffic with a priority of Category I. This increased Category I traffic in the network from 1.1% of all data packets to 7.1% of all data packets. This constitutes an increase of 645% in Category I data packets. Figures 21,22, and 23 present graphs of the effects of the increase on transmission times for Category I data packets, Category II data packets, and status packets. The solid line in each graph is the average transmission time (portrayed at each 250 millisecond interval) when the network is loaded with 7.1% priority packets and the dashed line is when the network is loaded with 1.1%

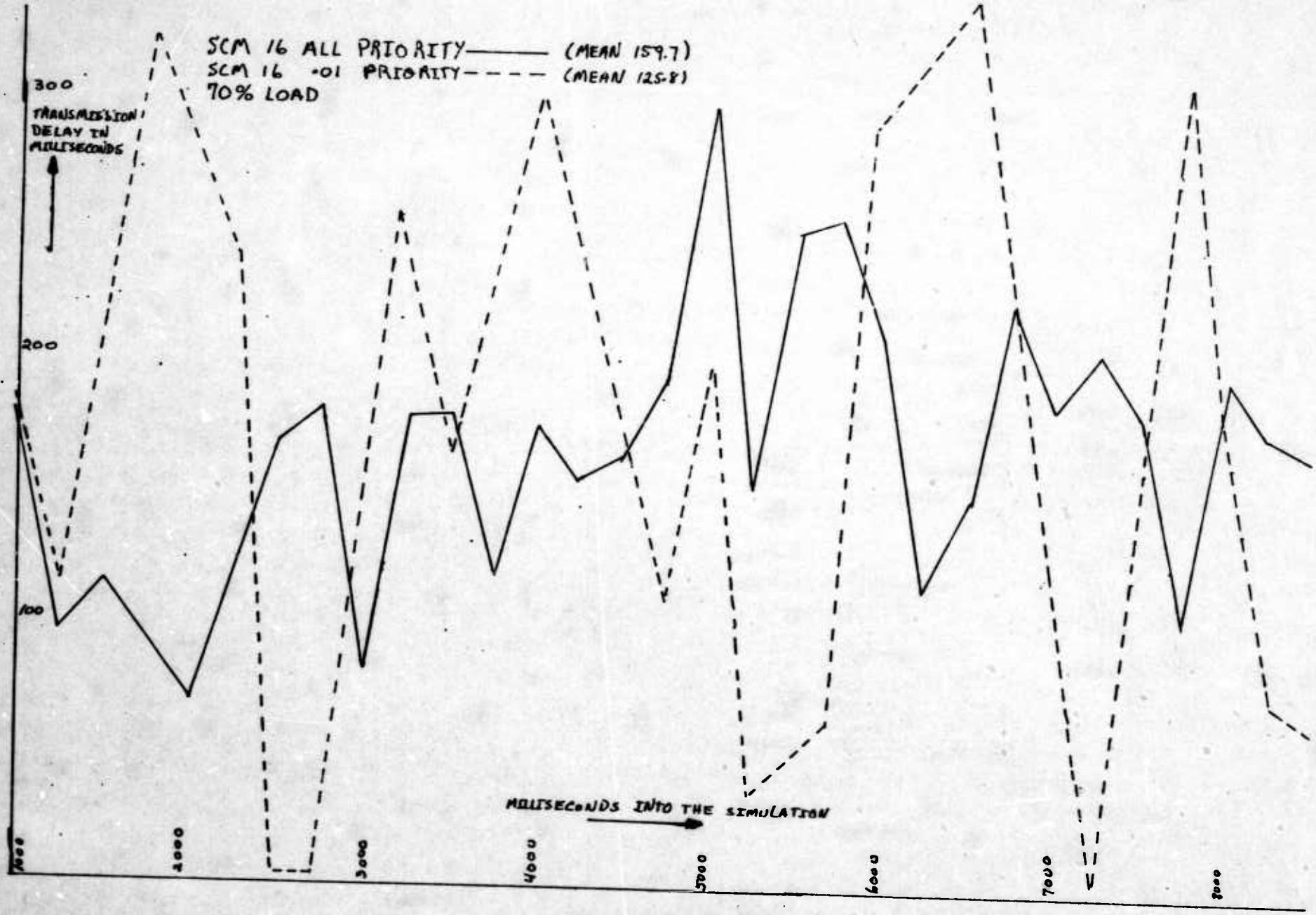


Figure 21 - 7.1% CATI - Effect on CATI

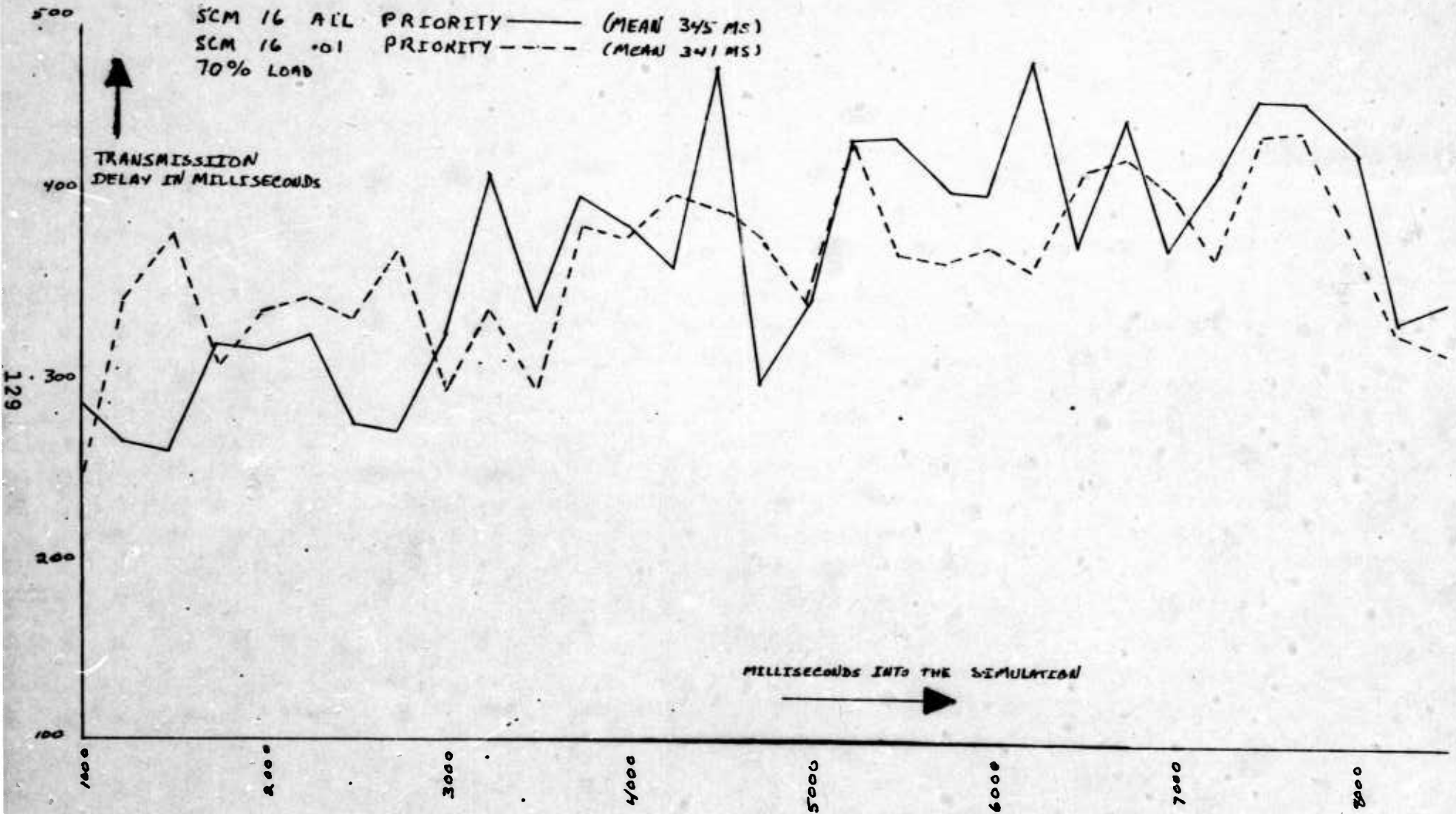


Figure 22 - 7.1% CATI - Effect on CATII

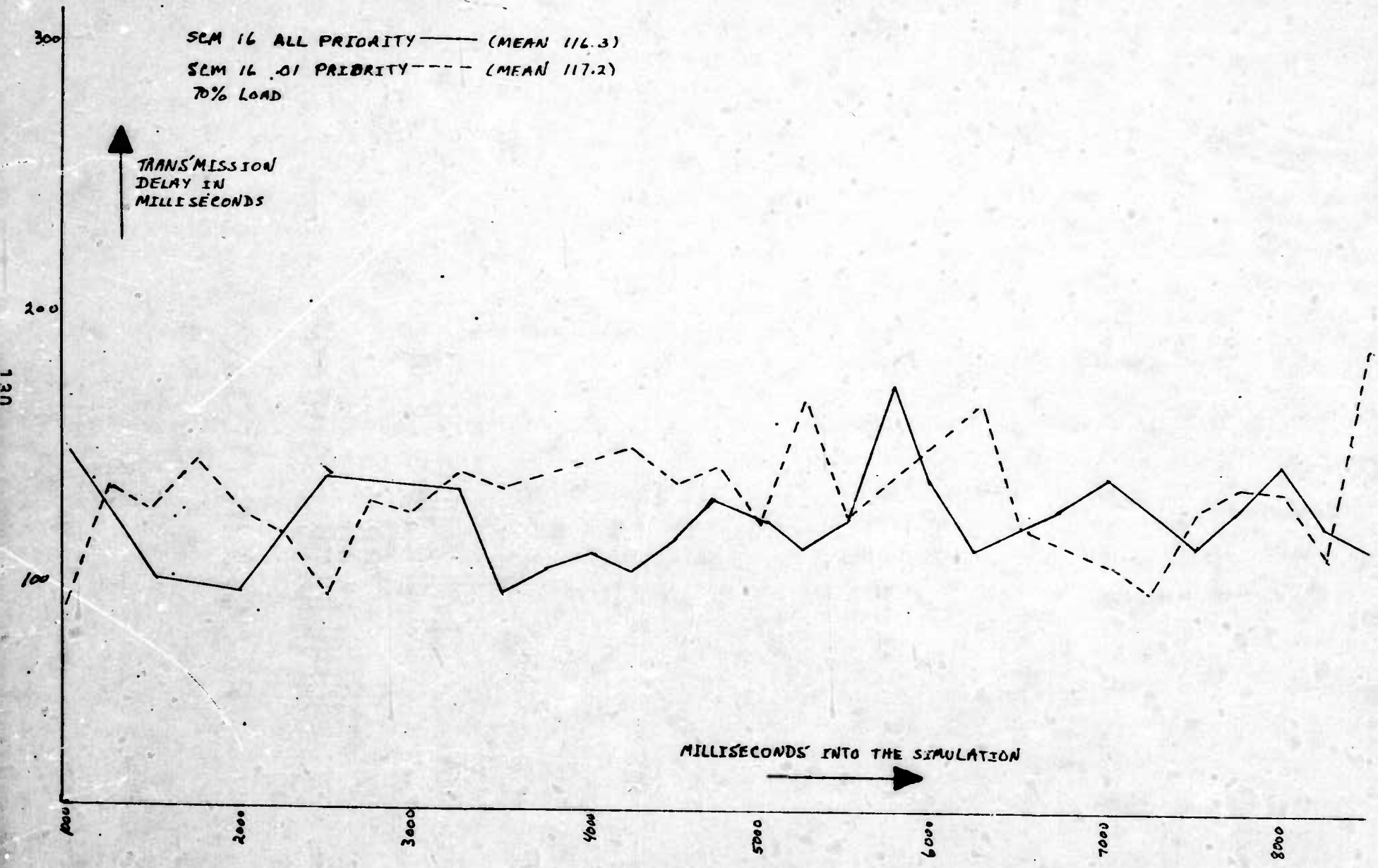


Figure 23 - 7.1% CATI - Effect on Status Packets

priority packets.

The transmission time for each packet was accumulated at the packet's arrival time in the network for graphing purposes. Since the simulation was stopped at 10000 milliseconds (10 seconds) and those packets still in the network were discarded, this caused a sharp drop off in transmission time statistics near the termination of the simulation. Therefore the last 1500 milliseconds of the simulation data is not representative and is not graphed in the figures. For this reason future users may prefer to accumulate transmission times as packets leave the network. This will require a simple change in the SPSS control cards (setting the parameter TIME equal to TIME + TTIME).

A student T-test was performed on the means of each set of graphed data with the null hypothesis that they were equal. This hypothesis was not rejected at the 90% confidence level for Category II packets, although intuitively a reasonable conclusion would be that increasing higher priority workload would increase transmission delays for lower priority workload. The hypothesis was rejected for Category I packets. A visual comparison of the means of the Category I packets shows a 27% increase in transmission time.

Though 93% of the traffic showed no significant change in transmission times, there did appear to be significant changes in flow patterns. Table XXVI lists the mean times

packets spent in queues at each node. The column containing the mean time spent in output queues demonstrates that flows have changed. Four means went up from the 1.1% pri column tc the 7.1% pri column, and four means went down. A student T-test was performed on each pair of output queue means. Six of the eight tests rejected the null hypothesis that the means were equal at the 98% confidence level. One of the remaining two rejected the null hypothesis at the 95% confidence level and the other one (Andrews) showed no significant change. Therefore a reasonable assumption is that flow patterns did indeed change.

TABLE XXVI

MEAN TIMES PER PSN

Node	Mean Times In					
	Input Queues		Output Queues		Transmission	
	1.1% Pri	7.1% Pri	1.1% Pri	7.1% Pri	1.1% Pri	7.1% Pri
Albany	1.366	2.695	112.517	137.986	34.07	36.67
Andrews	2.582	2.565	94.740	93.997	23.212	23.479
Fort Dietrick	6.448	5.852	44.228	53.793	51.569	54.322
Gentile	3.797	3.357	117.588	91.595	40.839	36.668
Hancock	1.276	1.234	83.601	49.439	27.634	27.042
McClellan	.812	1.034	21.846	34.118	25.634	27.536
Norton	1.248	1.103	64.470	100.912	42.000	40.336
Tinker	2.071	1.823	245.341	220.827	40.852	38.154

Since the queueing discipline in AUTODIN II is based on

priority, with first-in first-out within priority a reasonable assumption is that significant increases in priority load on the network will increase the queue waiting, and therefore time in the network, for lower priority workload. Since this did not happen, and since there were significant changes in flow patterns through the network, a conclusion is suggested. That conclusion is that the routing algorithm is reacting to network traffic flow conditions and, furthermore, it is reacting in a positive manner.

Normal Traffic Load Increases

The analysis in Chapter 7 examined the network when it was loaded at 20% of network capacity and when it was loaded at 40% of network capacity. This section will examine the network when it is loaded at 60%, 70%, and 80% of network capacity.

No status packets are generated at 20% loading. In other words, no Switch Control Module output lines became congested, nor did any input queues become saturated. The routing algorithm operates as though it were a fixed routing algorithm. However at 40% loading 273 status packets moved in a nine second period and the algorithm no longer acted as though it were a fixed routing algorithm. As the load was increased the status packets generated also increased, as displayed in Table XXVII.

TABLE XXVII

STATUS PACKET FLOW

% of Capacity	Number of Status Packets	% Increase
20%	0	-----
40%	273	-----
60%	403	48%
70%	464	15%
80%	551	19%

But even with these increases in status packets, the generation of status packets for network changes only does not approach the generation of status packets on a 100 millisecond periodic basis. The status packet figures in Table XXVII are for a nine second period. The periodic generation of status packets during that same period would place 4,410 status packets on the network. This is almost ten times the flow of status packets at an 80% load level. In fact, the simulation only generated 4,523 data packets at an 80% load level. This would indicate that periodic generation of status packets either should not be included in the algorithm, or that the period between generations should be considerably longer than 100 milliseconds.

The graph in Figure 24 presents the effects of increasing workload on status packet delays. This graph does not display significant difference to the eye, but a student T-test was used to compare the 60% and 80% means to the 70% mean,

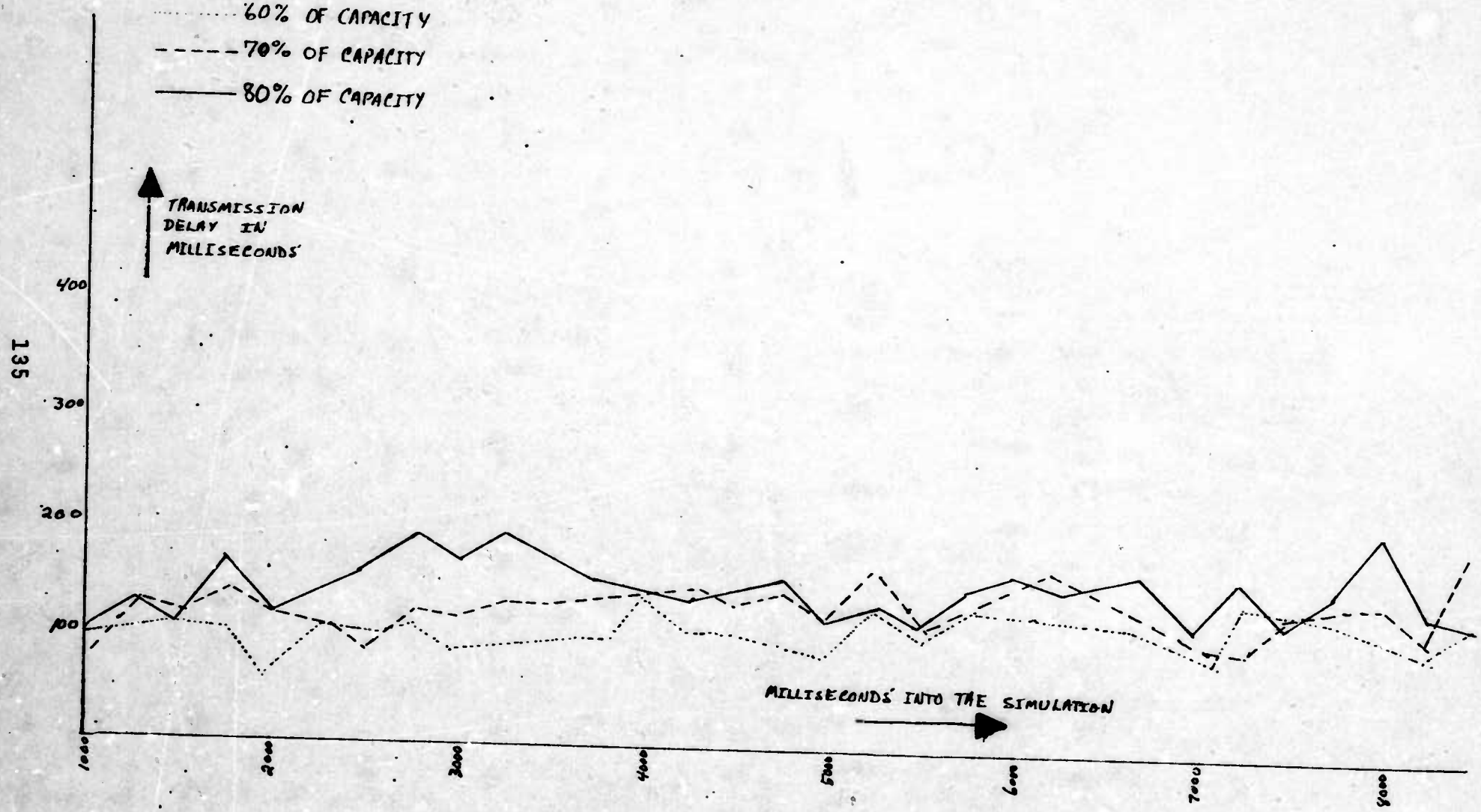


Figure 24 - Network Status Packet Flow Graph

with a null hypothesis that the differences between them were statistically insignificant. The T-test rejected the null hypothesis at a 98% confidence level. Therefore there are significant differences between the delay times for status packets at different loading percents.

The same is not true for Category I data packets. Though the graph in Figure 25 seems to show more differences than the graph for status packets, the student T-test would not reject the null hypothesis that their means were equal at any confidence level (in the T distribution table, anyway). The increases in network load did not appear to effect Category I transmission times. This may be due to the small percentage of total traffic which Category I packets represents.

The effect of the increases in traffic load on Category II traffic is clearly evidenced by the graph in Figure 26, and the student T-test confirmed that there was significant differences between the means of Category II packet transmission time delays at 60%, 70% and 80% network capacity loadings. Furthermore, the 80% simulation test may not have reached a stable condition during the 10000 millisecond run. The peaks toward the end of the graph were pushing beyond earlier peaks. The drop off caused by accumulating network time at the packet's arrival is clearly evidenced in the behavior of the 80% line.

The Q-GERT simulation data provides more evidence that

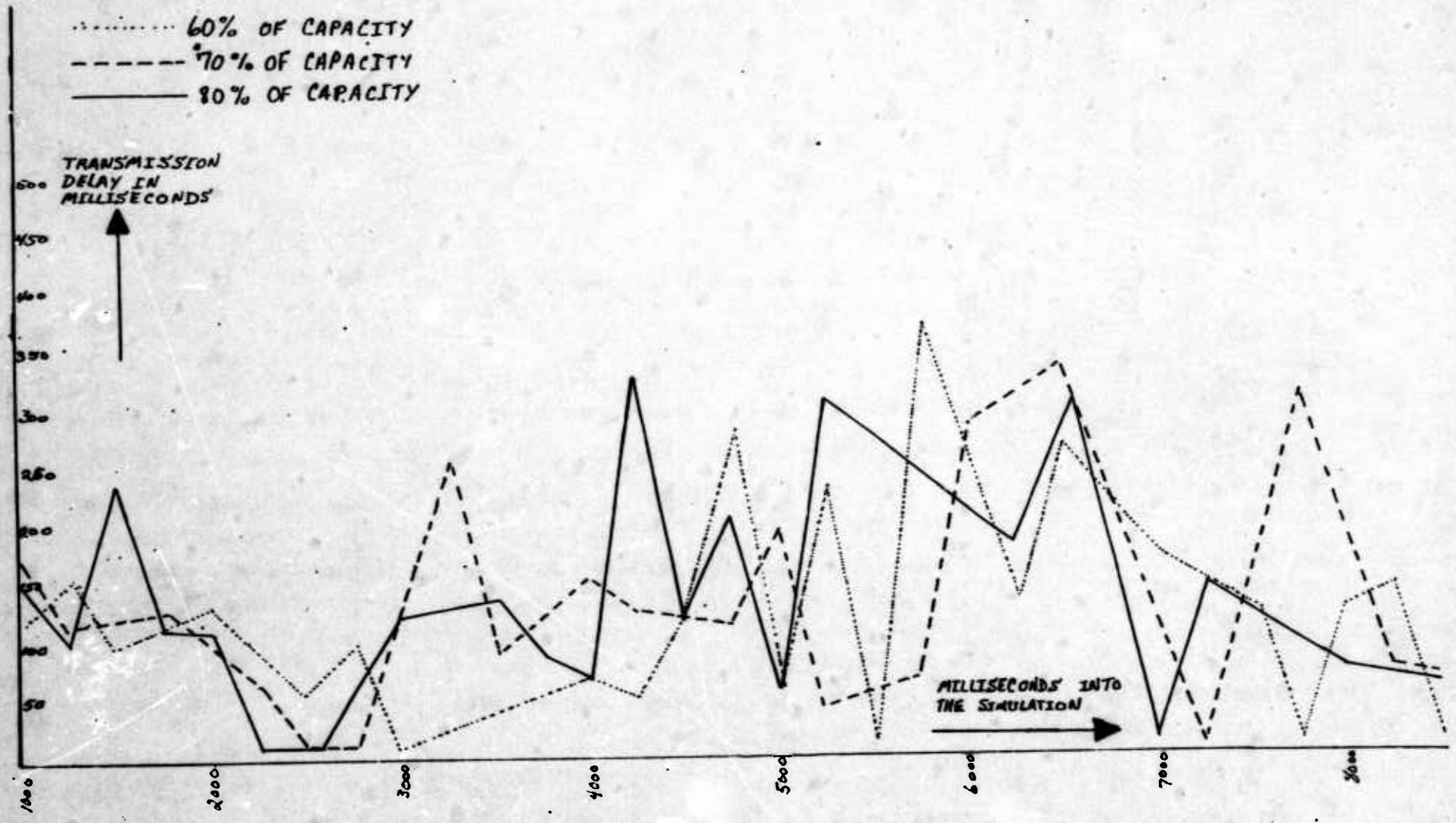


Figure 25 - Category I Flow Graph

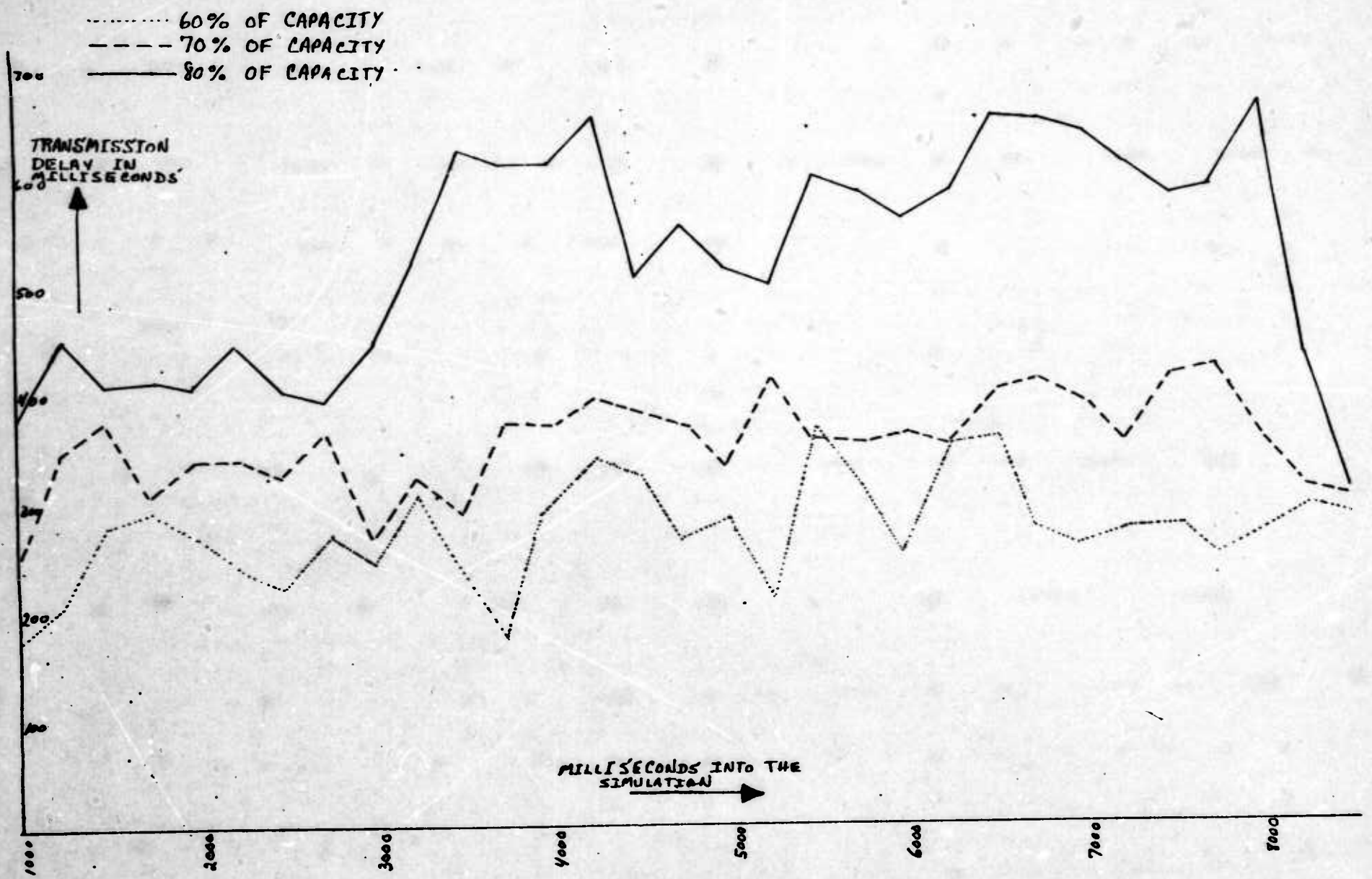


Figure 26 - Category II Flow Graph

the 80% run may not have stabilized. Many of the Q-nodes representing output queues were either the same as, or close to, the maximum number in that queue when the simulation ceased. This indicates a possible build up in those queues. Also, server utilization was quite high (again from Q-GERT data) with many servers above 90%.

The Q-GERT data for the 70% simulation had much lower queue occupancy figures, although several servers were above 90% utilized, and the 60% simulation had very little average back up.

Conclusions

Periodic generation of status packets should not be a part of the AUTODIN II routing algorithm. Further, loading the network beyond 70% could cause queue back up problems. However, moderate increases in Category I traffic do not seem to effect the network.

By changing the Q-GERT control cards the network could be initialized to the termination conditions of the 80% simulation and the simulation rerun to determine when and with what queue occupancy conditions the simulation stabilizes. This is a possible future experiment requirement. There are also many other possibilities, such as adding and/or subtracting Switch Control Modules to/from Packet Switching Nodes, changing the line connections between Switch

Control Modules, and increasing the line connections between Switch Control Modules.

The analysis of data in Chapters 7 and 8 is only a small sample of possible uses of the model. Future users should be limited only by their own imaginations and sponsors requirements or questions.

Bibliography

1. Western Union Telegraph Co., AUTODIN II Design Executive Summary, Defense Communications Agency Contract No. 200-c-637, Government Systems Division, McLean, Virginia, May, 1978.

2. Systems Control, Inc., AUTODIN II Network Routine - Final Technical Report, Contract No. WDL-SC-606040-EC, Annex 1, Palo Alto, CA., September. 1977.

3. Rockwell International, DCA Switching System Simulation and HOL Study Program - Final Report, Volume I, Contract DCA100-78-C-0013, Newport Beach, CA., July, 1979.

4. Whittenton, John A., An Analysis of A Routing Algorithm for AUTODIN II. MS thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December, 1979.

5. Western Union Telegraph Company, AUTODIN II/ Phase I, Defense Communication Agency Contract No. 200-C-637, Part V Design (Technical) Specification - Support I Network Design, Government Systems Division, McLean, Virginia, April, 1976.

6. Western Union Telegraph Company, AUTODIN II - Network Routing, Defense Communication Agency Contract No. 200-C-637, Government Systems Division, McLean, Virginia, November, 1978.

7. Systems Control, Inc., AUTODIN II Network Routing Hierarchical Routing Design, Volume I, Contract No. DCA 100-78-C-0013, Newport Beach, CA., July, 1978.

8. Pritsker, A. Alan B., Modeling and Analysis Using Q-GERT Networks, New York: Halsted Press, 1979.

9. Nie, Norman H., Hull, C. Hadley, Jenkens, Jean G. Stembrenner, Karin, and Bent, Dale H., Statistical Package for the Social Sciences, McGraw-Hill, Inc., 1975.

10. Nie, Norman H. and Hull, C. Hadley, SPSS Update, McGraw-Hill, Inc., 1979.

11. System Control, Inc., AUTODIN II Network Routing - Update of Network Performance Analysis, Contract No. WDL-SC-606280-EP, Annex 2, Palo Alto, CA., May, 1978.

12. Wannacott, Thomas H. and Wannacott, Ronald J., Introductory Statistics, John Wiley and Sons, Inc., New York, 1969.

APPENDIX A:

AUTODIN II USER'S MANUAL

Contents

List of Figures

List of Tables

USERS MANUAL

CHAPTER A-1

CONFIGURING THE AUTODIN II SIMULATION MODEL

Introduction

The AUTODIN II simulation model provides a means for testing conditions which may not normally exist, but which could occur, against different network configurations to find the network configuration which is most responsive. Thus network configurations and conditions can be tested before hardware is modified. However, a body of basic knowledge is required concerning the network to be modeled in order to configure and exercise the model. Chapter One will trace an example from conception of the network to the point of providing the necessary formatted information to the AUTODIN II simulation. Chapter Two will provide the formatted input data definition. Chapter Three will describe information which the model will provide as output. The necessary job and task control parameters required to complete a computer run will be described in Chapter Four.

The information required from the user to configure and

exercise the model falls naturally into three areas. The first of these areas is information about the physical configuration of the network to be modeled. The second area is information describing traffic load expected over the network. The final area is information concerning traffic control parameters. A section of this chapter will be devoted to each of these areas. Some of this information is static and remains constant throughout a single computer run. Other portions of the information can be changed dynamically during the computer run to simulate changes in traffic load, loss of transmission lines, loss of AUTODIN II nodes, and other occurrences which can happen dynamically. A section concerning information which can be changed dynamically will close this chapter.

Model Configuration

The example which will be used to demonstrate the information necessary to the AUTODIN II simulation model is the eight node AUTODIN II network proposed by Defense Communication Agency (DCA). The maximum number of nodes for the AUTODIN II simulation model is eight, although less nodes can be modeled.

The first concern of the simulation user is the number of network nodes to be simulated. Assume that the model designer has decided to model eight nodes (Figure A-1), located at Albany, Andrews, Hancock, Fort Dietrick, Gentile,

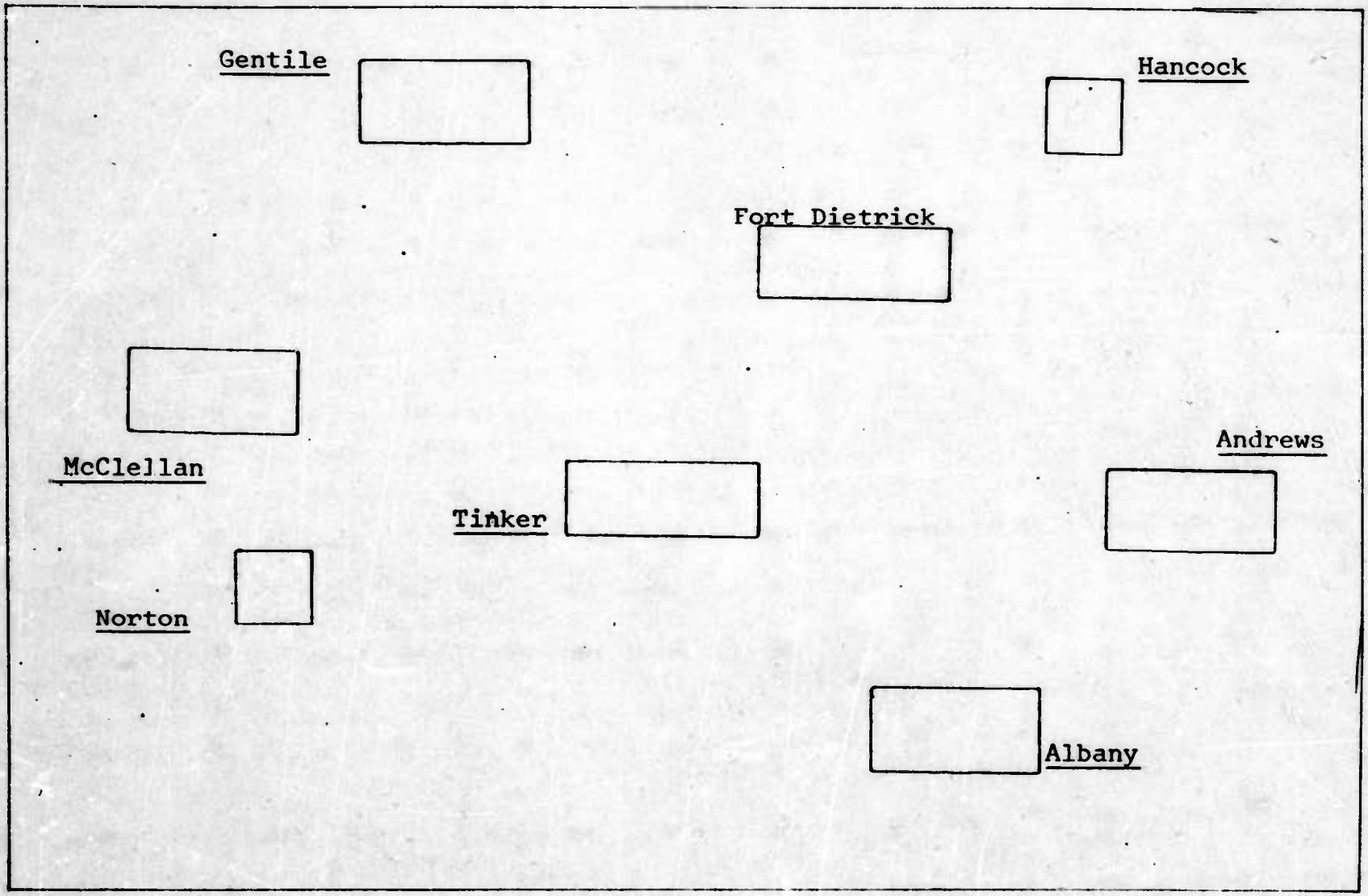


Figure A-1. Selected Packet Switching Nodes

Tinker, McClellan, and Norton. The nodes must then be numbered consecutively from one to the maximum number of nodes modeled (Figure A-2). These numbers will be placed in the user input record field labeled INODE.

Each Packet Switching Node (PSN) is composed of one or more PDP 11/70 computers which perform the process of routing AUTODIN II packets through the network. These computers are called Switch Control Modules (SCMs). The simulation limit is twenty-five (or less) SCMs for the entire network, and five SCMs in each Packet Switching Node. Again assume the model designer, in establishing the information to configure the network, has determined the number of SCMs he wants at each node. To determine this he has considered the traffic load, both coming into the network at that node, and flowing to and through that node from other nodes. More SCMs allow more transmission line connections and more traffic, but, of course, there is a cost constraint. The example decisions are in Table A-I.

TABLE A-I

SWITCH CONTROL MODULES/PSN

City	Packet Switching Node Number	Number of Switch Control Modules
Albany	1	2
Andrews	2	2
Fort Dietrick	3	3
Gentile	4	2
Hancock	5	1
McClellan	6	2

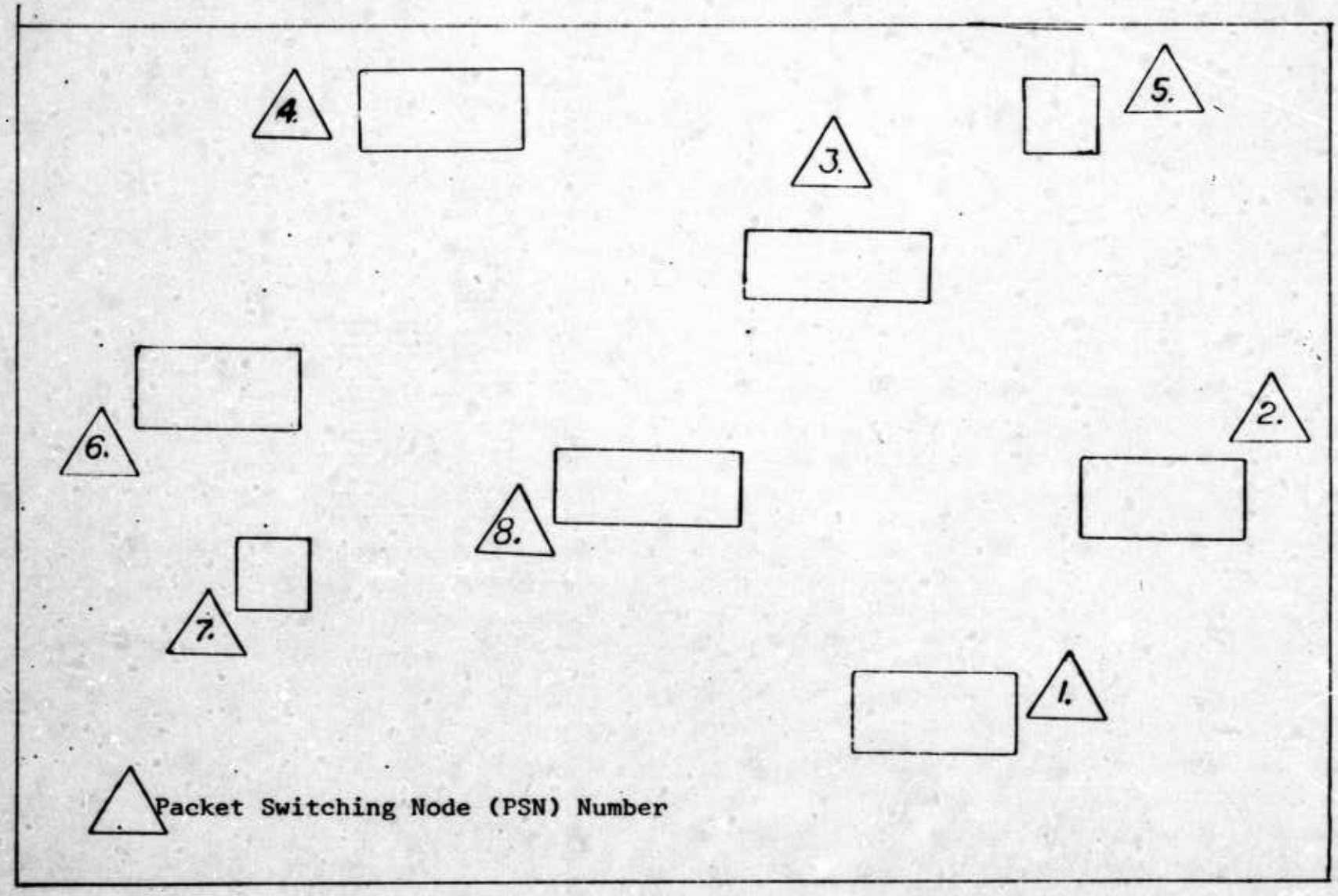


Figure A-2. Packet Switching Node Numbering

Norton	7	1
Tinker	8	3

Total Switch Control Modules 16

The Switch Control Modules must be numbered consecutively within the total network. Therefore the Switch Control Modules for the example will range from one to sixteen (Figure A-3). The numbering can be arbitrary as long as consecutive numbers are used. These numbers will be placed in the user input record field labeled ISCM when the record is established. The ISCM and INODE fields appear in all user input records. All records contain information at the Switch Control Module level. Therefore the Packet Switching Node Number will be repeated for each Switch Control Module which is contained in it.

Each SCM is connected to other SCMs by transmission lines. The simulation limits are from one to five transmission lines per SCM. Therefore the line limit for the network is 125 (or less). The model designer must determine how SCMs should be connected together by transmission lines. A definition is appropriate here. "Trunk" and "line" are used interchangeably for purposes of this simulation and a line is a single connection (two way) between two Switch Control Modules. A link consists of <u>all</u> lines (or trunks) connecting two Packet Switching Nodes. Figure A-4 illustrates the line connections for our example. Line connectivity is established by entries into the ICONSCM fields of user input record type 001. There are five of these fields,

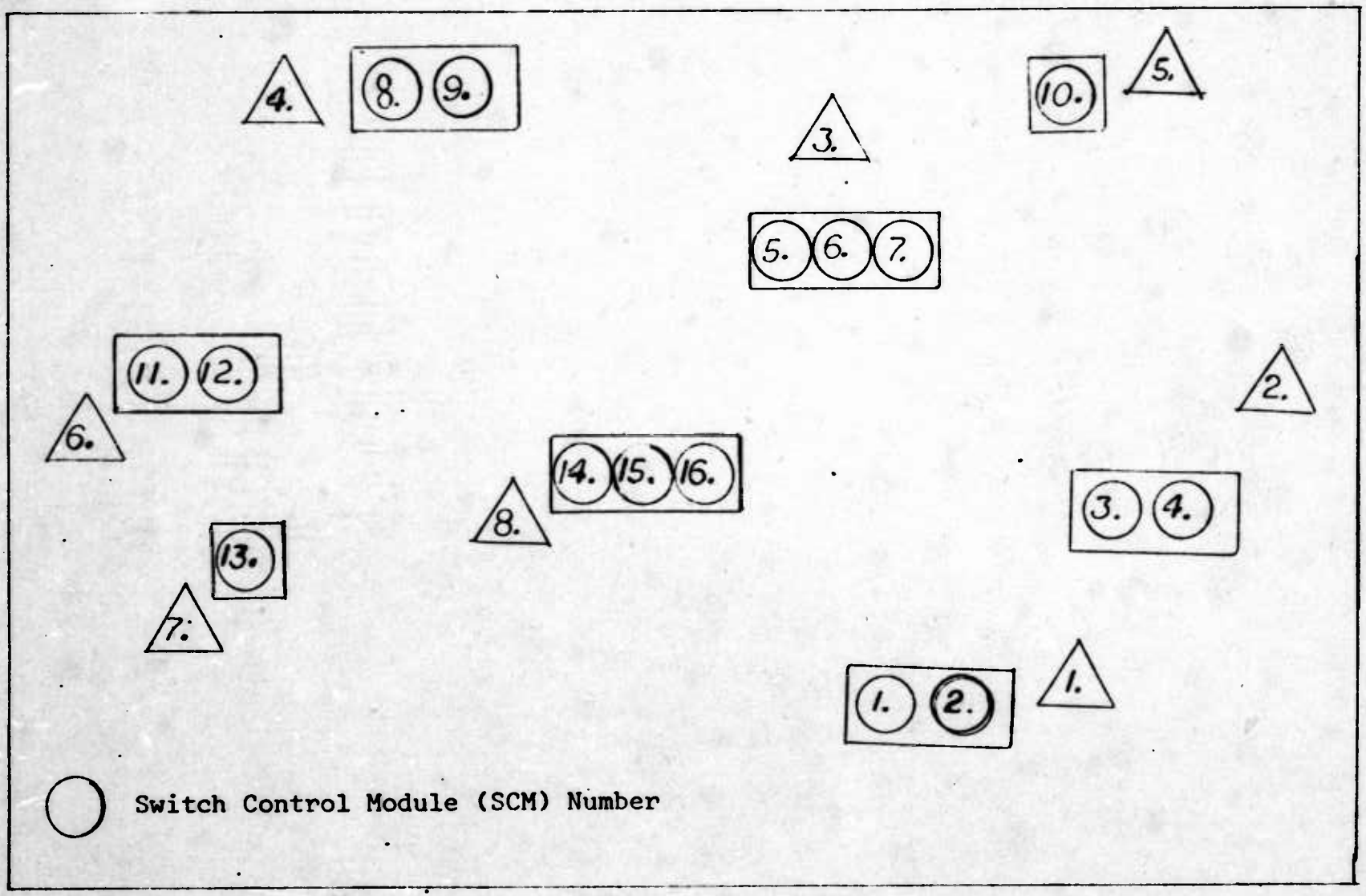


Figure A-3. Switch Control Module Numbering

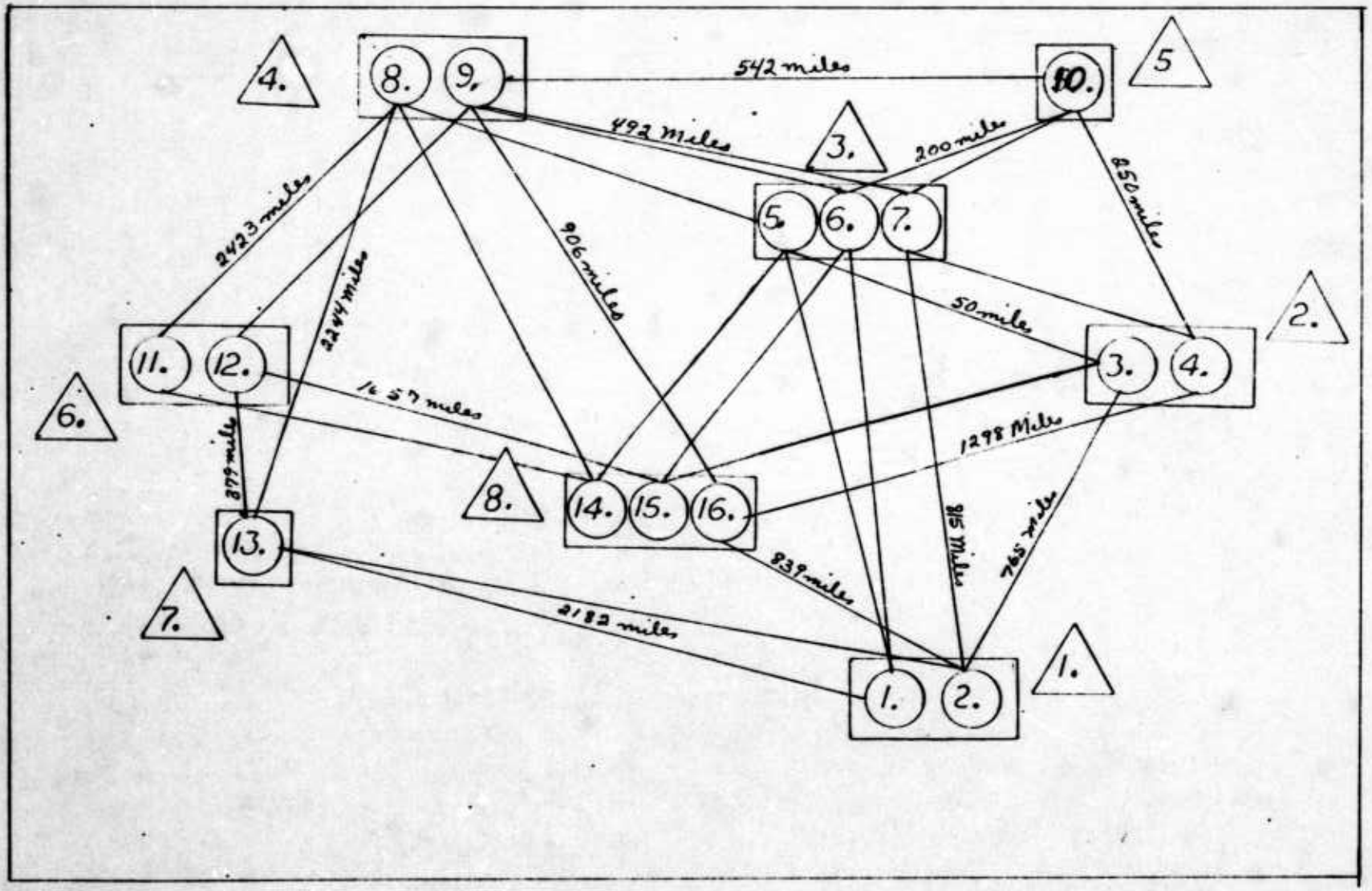


Figure A-4. Transmission Line Connectivity

one for each possible line (entries not used may be left blank). The entries are the SCM numbers of the Switch Control Module connected by transmission lines to this Switch Control Module (as represented by the SCM number in ISCM of the type 001 record).

The final information the model designer must provide to complete configuring the network is a mileage figure for each of the line connections as presented in Table A-II, and Figure A-4. Line mileage is established in the simulation network by entries into the LINDLA fields of user input record type 001. There are five of these fields; each one corresponds positionally to an ICONSCM field. If a line connection has been established in the ICONSCM field, the corresponding mileage should be entered in the LINDLA field. Otherwise, the LINDLA field should be left blank.

Traffic Load

Once the network is configured, the traffic the network will carry must be characterized. Since host computers and terminals are connected to a Packet Switching Node at the Switch Control Module level, traffic load is developed at that level. The model designer must provide traffic load arriving from outside the network at each SCM; this traffic load will be provided with units in millions of characters per hour (Figure A-5). It should not contain any character counts for network protocal, but should only count alphabetic/

TABLE A-II

LINE MILEAGE TABLE

PSN Locations	Node	Node	SCM	SCM	Mileage
Albany-Norton	1	7	1	13	2182
Albany-Fort Dietrick	1	3	1	5	815
Albany-Fort Dietrick	1	3	1	6	815
Albany-Norton	1	7	2	13	2182
Albany-Tinker	1	8	2	16	839
Albany-Fort Dietrick	1	3	2	7	815
Albany-Andrews	1	2	2	3	765
Andrews-Tinker	2	8	3	15	1298
Andrews-Fort Dietrick	2	3	3	5	50
Andrews-Tinker	2	8	4	16	1298
Andrews-Fort Dietrick	2	3	4	7	50
Andrews-Hancock	2	5	4	10	250
Fort Dietrick-Tinker	3	14	5	14	1298
Fort Dietrick-Gentile	3	8	5	8	492
Fort Dietrick-Gentile	3	8	6	9	492
Fort Dietrick-Hancock	3	5	6	10	200
Fort Dietrick-Tinker	3	8	6	15	1298
Fort Dietrick-Gentile	3	8	7	9	492
Fort Dietrick-Hancock	3	5	7	10	200
Gentile-McClellan	4	6	8	11	2423
Gentile-Norton	4	7	8	13	2244
Gentile-Tinker	4	8	8	14	906
Gentile-McClellan	4	6	9	12	2423
Gentile-Tinker	4	8	9	16	906
Gentile-Hancock	4	5	9	10	542
McClellan-Tinker	6	8	11	14	1657
McClellan-Tinker	6	8	12	15	1657
McClellan-Norton	6	7	12	13	379

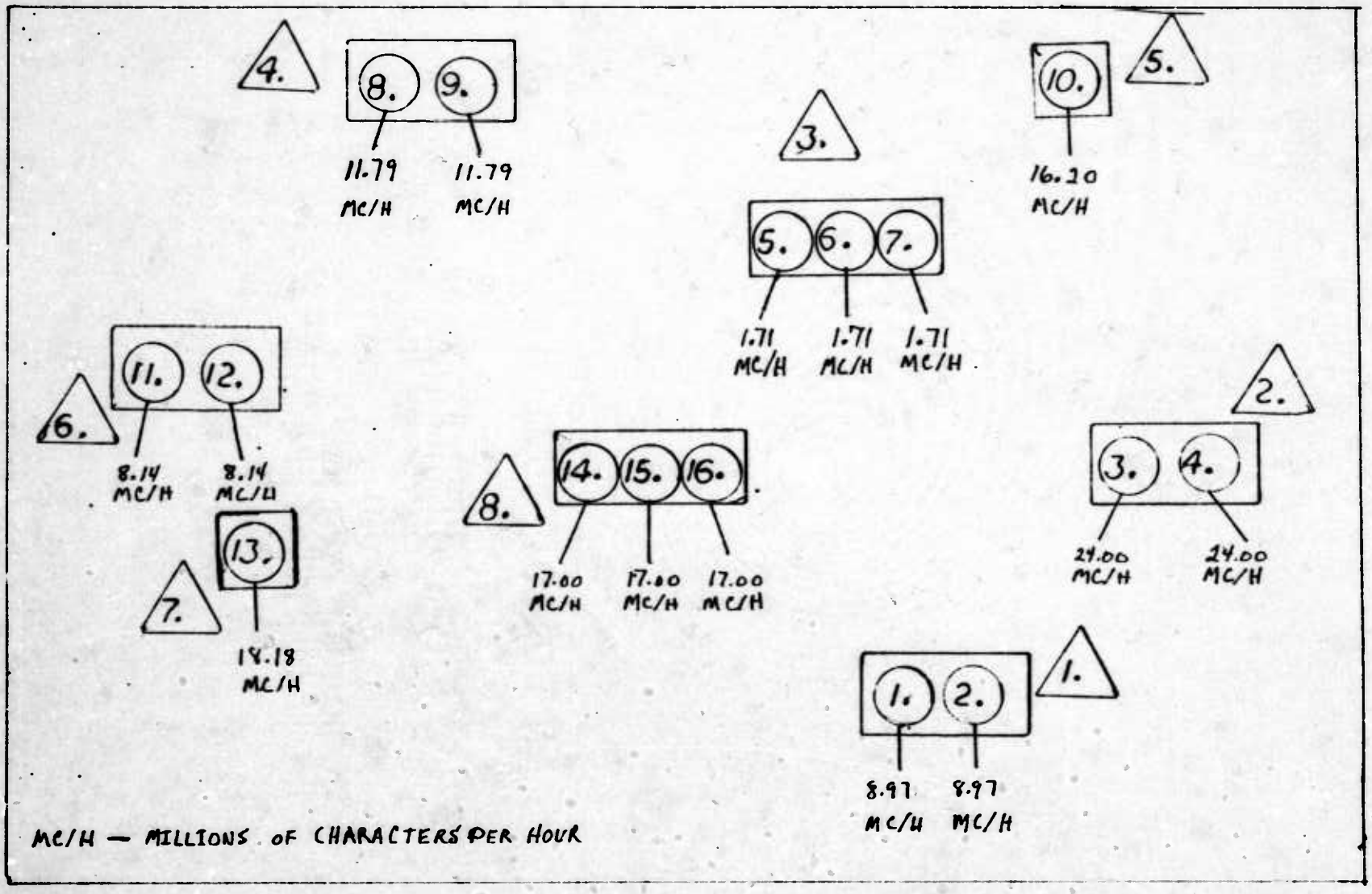


Figure A-5 Input Traffic Flow

numeric data from outside the network. Programming within the model converts the millions of characters per hour to an arrival rate for packets in milliseconds, but to accomplish this the model designer must also provide the percentage of all packets which are short packets (75 characters or 600 bits). This percentage is entered in the user input record type 001 in the field labeled SHRTPKP. The remaining packets will be long packets (588 characters or 4704 bits long). Short packets, in general, are produced by terminals; long packets by host computers.

The model designer may want to gather statistics on a segment (or even two portions) of traffic load. Therefore two fields are provided in the input user type 001 record for traffic load. One is labeled GNMCHRS. If only one total count is provided for traffic load it should be entered in this field. The other field is labeled SPMCHRS. When both fields have counts (in millions of characters per hour) entered into them a percentage of the packets generated, corresponding to each field's percentage of the total traffic, is uniquely identified as from that source and statistics may be accumulated concerning both or either segment of traffic load. However, only long packets are identified as originating from SPMCHRS.

Packets carry a precedence (or priority) level. There are four levels, Categories I through IV. Since precedence determines processing priority, the model designer must de-

termine what percentage of AUTODIN II packets will be assigned each precedence level. Table A-III and Table A-IV detail the decisions made concerning source traffic load for the example network.

Another aspect of traffic load which must be provided is a means of determining the destination of each packet. The model designer must provide two percentage distributions to accomplish this purpose. Each percentage distribution is entered at the Switch Control Module level. All traffic leaving the SCM must be directed to one of eight Packet Switching Nodes. The first percentage distribution determines the percentage of the packets originating at the SCM which are directed to each of the Packet Switching Nodes. These percentages are entered in the input record type 002 in the fields labeled NODTRAP, NODTRP2,···, NODTRP8. These percentages must add to 100%. Table A-V details the decisions made for the example network.

The second percentage distribution determines the percentage of packets from other Packet Switching Nodes arriving at this Packet Switching Node which have this SCM as their destination. Figure A-6 illustrates how these percentages are related and Table A-VI records the percentages for the example network. These percentages are entered in user input record type 003 in the fields labeled SCMTRAP, SCMTRP2,···, SCMTRP8.

TABLE A-III

SHORT PACKET % TABLE

SCM Number	GNMCHRS Millions of	SPMCHRS Characters/Hour	Short Packet percentage
1	8.97	0	.38
2	8.97	0	.38
3	24.00	0	.47
4	24.00	0	.47
5	1.71	0	.25
6	1.71	0	.25
7	1.71	0	.25
8	11.79	0	.49
9	11.79	0	.49
10	16.20	0	.30
11	8.14	0	.47
12	8.14	0	.47
13	18.18	0	.24
14	17.00	0	.27
15	17.00	0	.27
16	17.00	0	.27

TABLE A-IV

PRECEDENCE % TABLE

SCM Number	Precedence CATI	Precedence CATII	Precedence CATIII	Precedence CATIV
1	.01	.99	.00	.00
2	.01	.99	.00	.00
3	.01	.99	.00	.00
4	.01	.99	.00	.00
5	.01	.99	.00	.00
6	.01	.99	.00	.00
7	.01	.99	.00	.00
8	.01	.99	.00	.00
9	.01	.99	.00	.00
10	.01	.99	.00	.00
11	.01	.99	.00	.00
12	.01	.99	.00	.00
13	.01	.99	.00	.00
14	.01	.99	.00	.00
15	.01	.99	.00	.00
16	.01	.99	.00	.00

TABLE A-V

SCM to PSN PERCENTAGES

Nodes	1	2	3	4	5	6	7	8
SCMs								
1	.14	.22	.02	.07	.06	.04	.09	.36
2	.14	.22	.02	.07	.06	.04	.09	.36
3	.12	.43	.04	.07	.12	.06	.06	.10
4	.12	.43	.04	.07	.12	.06	.06	.10
5	.05	.43	.01	.08	.06	.14	.01	.22
6	.05	.43	.01	.08	.06	.14	.01	.22
7	.05	.43	.01	.08	.06	.14	.01	.22
8	.09	.18	.02	.30	.06	.12	.07	.16
9	.09	.18	.02	.30	.06	.12	.07	.16
10	.06	.38	.03	.07	.18	.06	.06	.16
11	.06	.14	.04	.06	.05	.33	.09	.23
12	.06	.14	.04	.06	.05	.33	.09	.23
13	.15	.25	.01	.07	.11	.16	.12	.13
14	.14	.13	.02	.14	.08	.14	.07	.28
15	.14	.13	.02	.14	.08	.14	.07	.28
16	.14	.13	.02	.14	.08	.14	.07	.28

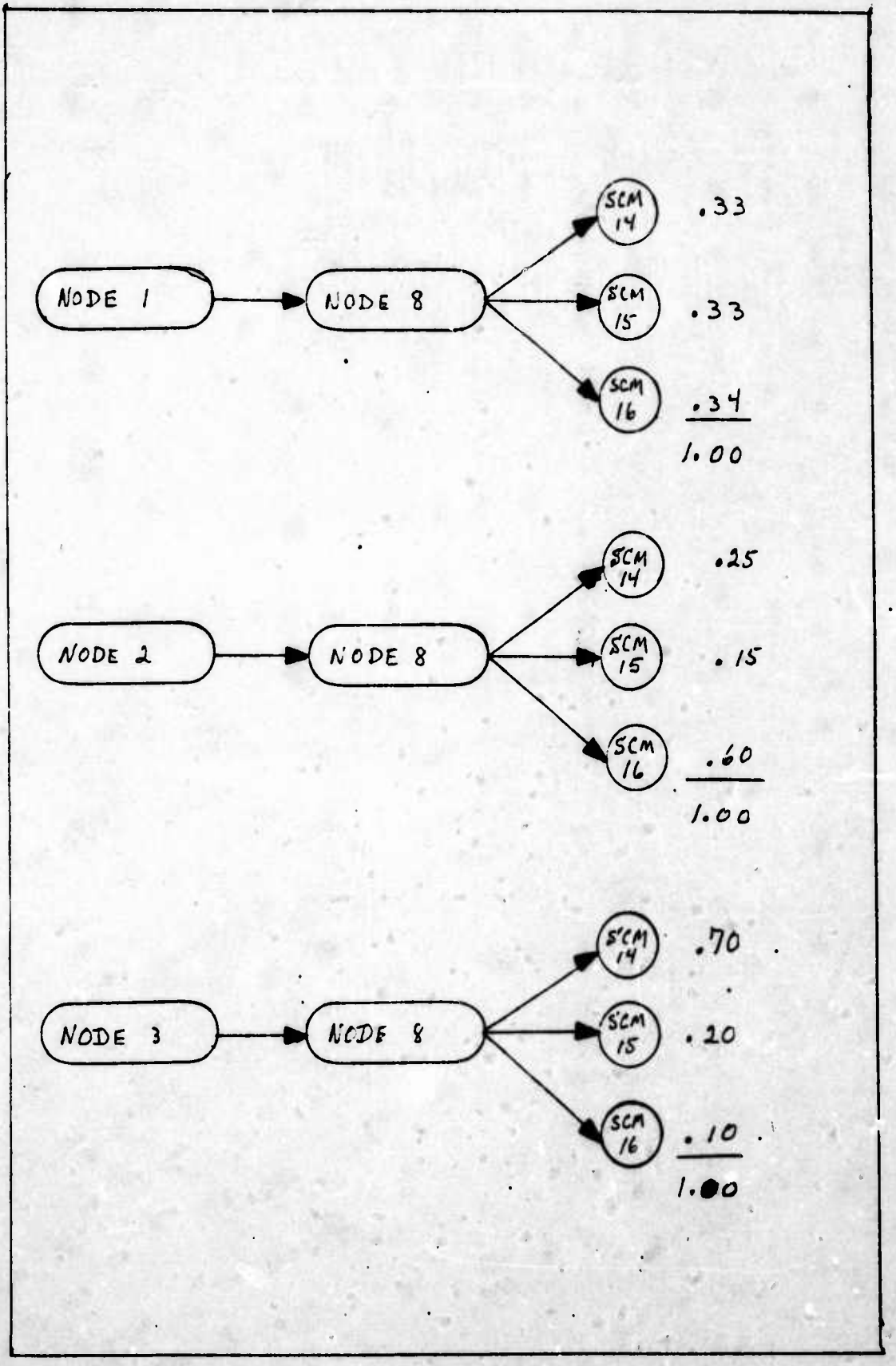


Figure A-6 Percentage Distribution for Destination SCM's

TABLE A-VI

PSN to SCM PERCENTAGES

Nodes	1	2	3	4	5	6	7	8
SCMs								
1	.50	.50	.50	.50	.50	.50	.50	.50
2	.50	.50	.50	.50	.50	.50	.50	.50
3	.50	.50	.50	.50	.50	.50	.50	.50
4	.50	.50	.50	.50	.50	.50	.50	.50
5	.33	.33	.33	.33	.33	.33	.33	.33
6	.33	.33	.33	.33	.33	.33	.33	.33
7	.34	.34	.34	.34	.34	.34	.34	.34
8	.50	.50	.50	.50	.50	.50	.50	.50
9	.50	.50	.50	.50	.50	.50	.50	.50
10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
11	.50	.50	.50	.50	.50	.50	.50	.50
12	.50	.50	.50	.50	.50	.50	.50	.50
13	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
14	.33	.33	.33	.33	.33	.33	.33	.33
15	.33	.33	.33	.33	.33	.33	.33	.33
16	.34	.34	.34	.34	.34	.34	.34	.34

Traffic Control

The remaining information the model designed must supply consists of queue levels which cause changes in the way the routing algorithm routes packets. There are up to six queues in each simulation Switch Control Module. There is a queue for incoming traffic and five (or less) queues for outgoing traffic, one for each transmission line. A saturation level is provided for each SCM in a Packet Switching Node. If the combined input queues exceed the combined saturation levels for a Packet Switching Node, the node is considered saturated and the routing algorithm will weight that as a factor in routing traffic through that node. The saturation level is entered in the user input record type 004 in the field labeled SATLEVL.

A congestion level is provided for each transmission line leaving an SCM. If the queue for that line exceeds its congestion level, that line is considered congested and the routing algorithm will weight that as a factor in routing packets over that transmission line. The congestion levels are entered in the user input record type 004 in the fields labeled CNGSTLV, CHGSTL2,···, CNGSTL5. Table A-VII records the saturation and congestion levels for the example network.

Dynamic Changes

Dynamic changes can be made to the network configuration and to the traffic load. Lines can be dropped and

TABLE A-VII

SATURATION AND CONGESTION LEVELS

SCM Number	Saturation Level	Congestion Levels 1	2	3	4	5	Number of Lines
1	9	7	7	7			3
2	9	7	7	7	7		4
3	9	7	7	7			3
4	9	7	7	7			3
5	6	7	7	7	7		4
6	7	7	7	7	7		4
7	7	7	7	7	7		4
8	9	7	7	7	7		4
9	9	7	7	7	7	7	5
10	7	7	7	7	7		3
11	9	7	7				2
12	9	7	7	7			3
13	7	7	7	7	7		4
14	6	7	7	7			3
15	7	7	7	7			3
16	7	7	7	7			3

brought back up. Source input can be set to a very low decimal point figure to simulate no input. The percentage of short packets (and therefore long packets) can be changed during the simulation process and so can the percentage of packets in each precedence level.

Changing a line's status is accomplished by placing either a minus one (to drop the line) or a plus one (to bring the line up) in the user input record type 900 in one (or more) of the fields labeled LNSTAT1,···, LNSTAT5, and by placing a time delay in milliseconds in the corresponding field labeled LNCHTM1,···, LNCHTM5 in the same record. This will cause the line to be dropped (or brought back up) at that time in the simulation.

The quantities and percentages supplied in GNMCHRS, SPMCHRS, SHRTPKP, and the four PRECDNC fields in the input type 001 record can be changed by entering a quantity (or percentage) in a like labeled field in the input type 910 record and by entering a time delay in milliseconds in the field labeled TIME in the same record. If any field is left blank in the 910 record, no change will take place for that particular field in the corresponding quantity or percentage. It will remain as it was entered in the type 001 record. A change in any field that effects the arrival rate of packets will cause that arrival rate to be recalculated.

As an example of the uses of the 900 and 910 records, a

node may be eliminated from the network dynamically by taking all its lines down, and by setting its source characters near to zero (if set to zero the field will act as if it were blank and no change will occur).

The use of user input to configure, load and control the network provides flexibility in using the model. The data definition necessary to use this flexibility is described in the next chapter.

CHAPTER A-2

DEFINITION OF USER INPUT

Input Data Records

The user's input data is keypunched onto six types of 80 column records or cards. The cards are numbered 001, 002, 003, 004, 900 and 910. The 001, 002, 003 and 004 cards are required for each SCM defined in the user's network configuration. The 900 and 910 cards are optional cards used for dynamic changes during a simulation run. The 001 card is used to define the user's network configuration, traffic originating at each SCM and percentages of each type of packets. The 002 card defines the distribution of destination nodes for packets originating at the specified node. The 003 card defines, for each node, the distribution of terminating packets to their destination SCM. The 004 card defines the saturation and congestion levels for each SCM. The 900 card allows the user to make dynamic changes in the network configuration during the simulation run. The 910 card allows the user to make dynamic changes in the network traffic and types of packets. A detailed explanation of each card type and the required user input is provided in this chapter. For ease of describing the input and output data, the proposed AUTODIN II

network defined in Chapter A-1 will be followed as an example through the next two chapters.

Input Record 001

Since the simulation is defined at the lowest level of packet switching, the user must define all input at that level. That level is the Switch Control Module (SCM).

The user should first define the network to be simulated as shown in Figure A-4. The PSNs are uniquely identified by integers 1 through NPSN (NPSN is less than or equal to 8), and the SCMs are uniquely identified by integers 1 through NSCM (NSCM is less than or equal to 25). Next the trunks, or lines, connected to each SCM are defined. Each SCM may be connected to a maximum of five other SCMs outside its own PSN. The SCMs within a PSN are assumed to be interconnected to each other through the Parallel Communication Link (PCL). The user is required to use the PSN and SCM integer identifier on the input cards. To define the outside SCM connected to a particular SCM, the user inputs the outside SCM integer identifier (as described in the following sections). Blank or zero implies a trunk is not connected.

Once the network configuration is defined, traffic data concerning message arrivals, packet priorities and packet destinations must be defined for each SCM. Two types of input traffic can be defined: General traffic (GNMCHRS) and Special traffic (SPMCHRS). Traffic units are in millions

of characters per hour. The sum of GNMCHRS and SPMCHRS must be greater than zero. General traffic is defined as the normal network traffic originating at this SCM. Special traffic is defined as an additional or unique traffic load which is to be tracked through the network. Special traffic is bulk traffic, ie, traffic that consists of large quantities of data, usually from a host, and is divided into large packets to be delivered to a destination PSN. Special traffic allows the user to analyze the effects of special or additional discrete traffic loads entering the AUTODIN II network. General and special traffic are uniquely identified on output data files for future user analysis.

The proportion of short and long packets in the network is defined by the user. The user specifies the percentages of short packets (SHRTPKT) in the network. SHRTPKT must be less than or equal to one. Long packets are determined by the program with the following equation:

$$SHRTPKT + LNGPKT = 1.0$$

AUTODIN II packets can be assigned one of four priorities, or precedence levels, depending on the proportion of traffic of CATI, CATII, CATIII and CATIV messages. The user can define the percentage of each of the levels of priority (PRECDNC) of packets as they enter the network. The assigned priority follows the packet from source to destination PSN and is available to the user in the output data.

The user's input must meet the following requirement:

$$0 \leq PRECDNCi \leq 1.0$$

$$\sum_i PRECDNCi = 1.0$$

Propagation time is determined by the distance, in miles, between the nodes. Since each user's input record is at the SCM level, the user must provide the distance between a particular SCM and the SCMs connected to it. The propagation time over any line is computed by the program by adding the time required to enter the packet into the trunk, for a particular packet type, and the distance in miles multiplied by the average propagation time for the network, ten microseconds per mile (Ref 7;V-B-33). The time to enter a packet into the trunk is defined internally in the simulation program. The actual propagation time is computed in the simulation program by the following equations:

1. For status packets

Time = 10 milliseconds + mileage / 10 miles per microsecond.

2. For short packets

Time = 20.1 milliseconds + mileage / 10 miles per microsecond.

3. For long packets

Time = 93.4 milliseconds + mileage / 10 miles per micro-

second.

The network configuration and traffic data defined in this section may now be punched on cards as defined in Table A-VIII.

The 001 cards for the network as defined in Capt. John Whittenton's thesis (Ref 4:5) will now be developed.

The first three fields of the card are self explanatory. General traffic (GNMCHRS) for each PSN was assigned the same quantities as the Systems Control Incorporation specification (Ref. 2:80). No data was available for proportioning the PSN traffic to the SCM level. Therefore, the PSN traffic will be divided evenly between the SCMs within each PSN.

To illustrate, Albany's host and terminal send traffic is 39868 BPS. Converting bits to characters (eight bits per character), seconds to hours and dividing by the number of SCMs in the Albany PSN (2), yields 8.97 million characters per hour. The GNMCHRS for each SCM is provided in Table A-IX

Next, the percent of packets originating from each PSN, to be assigned as short packets, is computed. From the data in Table IV of Capt. Whittenton's thesis (Ref 4:83), the proportion of short packets, per PSN, is obtained by dividing the number of short packets per second by the total number of packets per second. No data was available as to the proportion each SCM, within a single PSN, should be assigned. Therefore, the SCMs were assigned the same propor-

TABLE A-VIII

CARD TYPE 001 FORMAT

FORTRAN Label	Card Columns	Contents
ISCNI	1-2	SCM Identification Number
INODE	3-4	PSN Identification Number
INRSCM	5-6	Number of SCMs in this PSN
GNMCHRS	7-16	The normal, or general, traffic rate originating at this SCM, in millions of characters per hour
SHRTPKT	17-19	the percentage of total packets, originating at this SCM, to be identified as a short packet
ICONSCM	20-21 22-23 24-25 26-27 28-29	Other SCMs connected by trunk to this SCM.
SPMCHRS	30-38	Special, or identified, traffic originating at this SCM, in millions of characters per hour
LINDLA	39-43 44-48 49-53 54-58 59-63	the number of miles between this SCM and other connected SCMs (the mileage must be in the same order as the connected SCMs in ICONSCM described above).
PRECDNC	64-66 67-69 70-72 73-75	The proportion of packets to be assigned to each precedence level (highest level to lowest level, respectively).
CRDTYPE	78-80	Type of Card (001)

tions as their respective PSN.

The SCM to SCM connections are taken directly from Figure A-4. For example, SCM 2 (of the Albany PSN) is connected to four other SCMs: SCM 13 (Norton PSN), SCM 16 (Tinker PSN), SCM 7 (Fort Dietrick PSN) and SCM 3 (Andrews PSN). The first four values of ICONSCM (Table A-IX) will contain the connected SCMs: 13,16;7 and 3. The fifth value of ICONSCM will be blank, signifying no connection.

No special traffic (SPMCHRS) is identified in the example. Therefore, SPMCHRS is zero for all SCMs. However, suppose the user desired to track and analyze a specific or added workload on the network. The user would assign the expected traffic from this workload to SPMCHRS for each SCM. The output data (see Chapter A-3) would have a unique identifier for all packets generated from this special workload. An analysis could then be performed.

The estimated mileage between connected SCMs, or between PSNs, was taken from a national mileage map. The mileage to each connected SCM was input into LINDLA in the same order as the connected SCMs (ICONSCM) were entered on card 001.

In Capt. John Whittenton's thesis, precedence levels are established at only two levels, CATI as high priority and CATII,CATIII and CATIV as low (or none) priority. CATI makes up 1% of the total packets and CATII, CATIII and CATIV make up 99% of the total packets. Therefore, the first precedence level (in the 001 card) is .01, implying 1% of the

TABLE A-IX

EXAMPLE NETWORK STATISTICS

SCM	GNMCHRS	SHRTPKT	ICONSCM	PRCDNCi
1	8.97	.38	3,5,6	.01,.99,0,0
2	8.97	.38	13,16,7,3	.01,.99,0,0
3	24.00	.47	2,15,5	.01,.99,0,0
4	24.00	.47	16,7,10	.01,.99,0,0
5	1.71	.25	3.1.14.8	.01,.99,0,0
6	1.71	.25	9,10,1,15	.01,.99,0,0
7	1.71	.25	9,10,4,2	.01,.99,0,0
8	11.79	.49	11,13,14,5	.01,.99,0,0
9	11.79	.49	6,7,10,12,16	.01,.99,0,0
10	16.20	.30	4,6,7,9	.01,.99,0,0
11	8.14	.47	8,14	.01,.99,0,0
12	8.14	.47	9,13,15	.01,.99,0,0
13	18.18	.24	1,2,8,12	.01,.99,0,0
14	17.00	.27	5,8,11	.01,.99,0,0
15	17.00	.27	3.6.12	.01,.99,0,0
16	17.00	.27	2,4,9	.01,.99,0,0

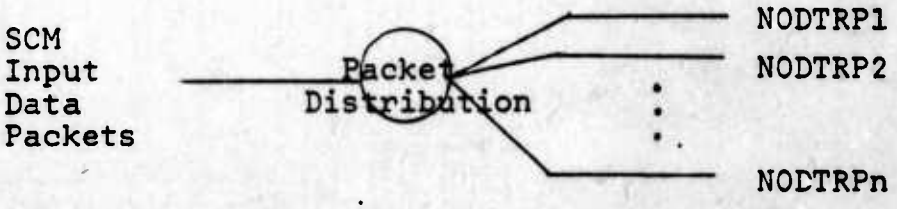
packets will be priority one; the second precedence level is .99, implying 99% of the packets will be priority two; and the third and fourth levels of priority are zero, implying no packets are assigned priorities three and four.

The 001 cards for the example described above are listed in Table A-X.

Input Record 002

The 002 card allows the user to describe a destination distribution for packets originating at each of the SCMs. The 003 card, as described in the next section, allows the user to describe local SCM delivery distribution for packets originating at each of the PSNs.

To develop the data for the 002 card, the user must know what percentages of traffic, originating at each SCM, is to be delivered to each of the PSNs in the network, (including the originating PSN itself). NODTRPi is defined as the proportion of packets destined for the ith node, as shown in Figure A-7.



Distribution of Input Packets

Figure A-7

TABLE A-X

EXAMPLE 001 CARDS

```
 1 1 2        8.97.3813 5 6          2182. 645. 645.   0.    0..01.99      001
 2 1 2        8.97.381316 7 3        2182. 839. 839. 600.    0..01.99      001
 3 2 2       22.35.47 215 5           600.1298.  50    0.    0..01.99      001
 4 2 2       22.35.4716 710          1298.  50. 196.   0.    0..01.99      001
 5 3 3        1.58.25 3 114 8          50. 645.1298. 392.    0..01.99      001
 6 3 3        1.58.25 910 115         392. 250. 645.1298.    0..01.99      001
 7 3 3        1.58.25 910 4 2         392. 250.  50. 645.    0..01.99      001
 8 4 2       10.98.49111314 5        2423.2244. 906. 392.    0..01.99      001
 9 4 2       10.98.491216 6 710      2423. 906. 392. 392. 542..01.99       001
10 5 1       15.09.30 9 6 7 4         542. 250. 250. 196.    0..01.99      001
11 6 2        7.58.47 814            2423.1657.   0.   0.    0..01.99      001
12 6 2        7.58.47 91513          2423.1657. 379.   0.    0..01.99      001
13 7 1       16.93.24 1 2 812        2182.2182.2244. 379.    0..01.99      001
14 8 3       15.84.2711 8 5          1657. 906.1298.   0.    0..01.99      001
15 8 3       15.84.2712 6 3          1657.1298.1298.   0.    0..01.99      001
16 8 3       15.84.27 9 4 2           906.1298. 839.   0.    0..01.99      001
```

NODTRPi has the following restrictions:

$$0 \leq \text{NODTRPi} \leq 1.0$$

$$\sum_i \text{NODTRPi} = 1.0$$

The 002 card allows the user to define a wide range of traffic distribution. The data describing the distribution must be punched in the format described in Table A-XI

The 002 cards for the network described in Capt. Whittenton's thesis (Ref 4:88), will now be described. Since the example network was not developed to the SCM level, the distribution of packets to destination node will be assumed the same for all SCMs within the same PSN.

The expected exchange of packets between nodes is given in Ref 17:V-B-131. from which the distribution of data packets for each SCM was assigned and punched on the 002 cards. The 002 cards for each SCM are listed in Table A-XII.

Input Record 003

The 003 card allows the user to describe the local delivery of packets by the SCMs. When packets reach their destination PSN, the determination must be made of which SCM can deliver the packets to the host or terminal. The distribution, defined on the 003 cards, is used by the simulation program to determine the percent of packets to be delivered by each SCM within each node. In addition, the user determines the proportion of packets each SCM can de-

TABLE A-XI

CARD TYPE 002 FORMAT

FORTRAN Label	CARD Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
NODTRP1	44-48	Proportion of traffic originating at this SCM, destined for PSN 1 through 8.
NODTRP2	49-53	
NODTRP3	54-58	
NODTRP4	59-63	
NODTRP5	64-66	
NODTRP6	67-69	
NODTRP7	70-72	
NODTRP8	73-75	
CRDTYPE	78-80	Type of card (002)

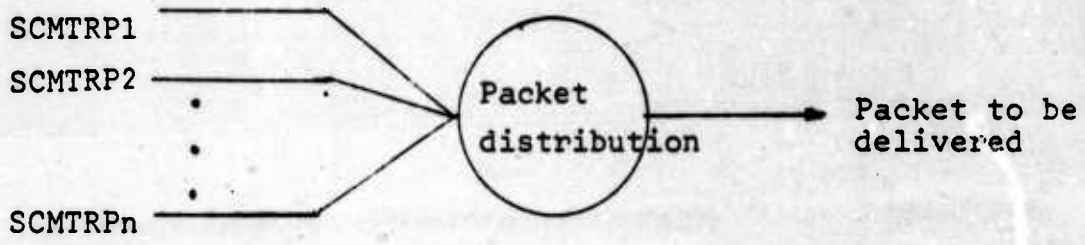
TABLE A-XII

EXAMPLE 002 CARDS

1	1		.14	.22	.02	.07.06.04.09.36	002
2	1		.14	.22	.02	.07.06.04.09.36	002
3	2		.12	.43	.04	.07.12.06.06.10	002
4	2		.12	.43	.04	.07.12.06.06.10	002
5	3	3	.05	.43	.01	.08.06.14.01.22	002
6	3	3	.05	.43	.01	.08.06.14.01.22	002
7	3	3	.05	.43	.01	.08.06.14.01.22	002
8	4		.09	.18	.02	.30.06.12.07.16	002
9	4		.09	.18	.02	.30.06.12.07.16	002
10	5		.06	.38	.03	.07.18.06.06.16	002
11	6		.06	.14	.04	.06.05.33.10.22	002
12	6		.06	.14	.04	.06.05.33.10.22	002
13	7		.15	.25	.01	.07.11.16.12.13	002
14	8		.14	.13	.02	.14.08.14.07.28	002
15	8		.14	.13	.02	.14.08.14.07.28	002
16	8		.14	.13	.02	.14.08.14.07.28	002

liver based on each packet's source PSN. That is, the user may specify that a particular SCM in a node is to deliver more packets than another SCM in the same node, and also specify a particular SCM to deliver a larger proportion of traffic from one node than from another node.

Let SCMTRP(i,j) be the proportion of traffic from node (i) to SCM (j). Figure A-8 describes the user input required for SCM (j).



Distribution of Packets at a node to be delivered by a particular SCM

Figure A-8

The user input must meet the following conditions for j:

$$0 \leq SCMTRP(i,j) \leq 1.0$$

$$\sum_i SCMTRP(i,j) = 1.0$$

The format of the 003 card is described in Table A-XIII.

Again, returning to the network being developed from Capt. Whittenton's thesis, no delivery data is known below the PSN level. Actually, the delivery data will not be known until the AUTODIN II users (and their hosts and terminals and expected traffic loads) are defined. For this example, assume each SCM within the same node delivers data packets in the same proportions. For example, if a node contains two SCMs, each SCM will deliver 50% of the traffic destined for that node. The appropriate 003 cards are shown in Table A-XIV.

Input Record 004

The 004 card allows the user to define the saturation and congestion levels for each SCM. The AUTODIN II routing algorithm uses these levels to determine if packets should be routed through or around a particular node. The PSN saturation level is a level which is compared to PSN input queue backup. When input queue backup exceeds this level, all PSN transmission links are flagged as saturated. The PSN queue backup is found by summing all its SCM input queues. The PSN saturation level is computed by summing all its SCM saturation levels. If a PSN saturation level is exceeded, all nodes are notified by network generated status packets. Packets, not destined for a saturated node, are routed around the node, i.e., another path is determined.

The link congestion level is a level which is compared to

TABLE A-XIII

CARD TYPE 003 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
SCMTRP1	44-48	Proportion of traffic from PSN1
SCMTRP2	49-53	to be locally delivered by
SCMTRP3	54-58	this particular SCM.
SCMTRP4	59-63	
SCMTRP5	64-66	
SCMTRP6	67-69	
SCMTRP7	70-72	
SCMTRP8	73-75	
CRDTYPE	78-80	Type of card (003)

TABLE A-XIV

EXAMPLE 003 CARDS

1	1	.50	.50	.50	.50.50.50.50.50	003
2	1	.50	.50	.50	.50.50.50.50.50	003
3	1	.50	.50	.50	.50.50.50.50.50	003
4	2	.50	.50	.50	.50.50.50.50.50	003
5	3	.33	.33	.33	.33.33.33.33.33	003
6	3	.33	.33	.33	.33.33.33.33.33	003
7	3	.34	.34	.34	.34.34.34.34.34	003
8	4	.50	.50	.50	.50.50.50.50.50	003
9	4	.50	.50	.50	.50.50.50.50.50	003
10	5	1.0	1.0	1.0	1.01.01.01.01.0	003
11	6	.50	.50	.50	.50.50.50.50.50	003
12	6	.50	.50	.50	.50.50.50.50.50	003
13	7	1.0	1.0	1.0	1.01.01.01.01.0	003
14	8	.33	.33	.33	.33.33.33.33.33	003
15	8	.33	.33	.33	.33.33.33.33.33	003
16	8	.34	.34	.34	.34.34.34.34.34	003

the link queue backup. When the link queue backup exceeds this level the link is flagged as congested. The PSN congestion level is determined from the average of the SCM congestion levels of each trunk making up a link (a trunk is a two way communication line, where as a link is a logical description of all trunks connecting two links). The PSN queue backup is found by computing the average of the output (trunk) queues making up a particular link. If a link is declared congested, all nodes in the network are notified through the generation of network status packets. Packets are, if possible, routed around the congested link or links.

The user must define the saturation level for each SCM and the congestion level for each trunk of an SCM. There are no limits for these values. However, saturation and congestion levels less than or equal to zero are meaningless. Also, saturation and congestion levels that are too high will not allow for a dynamic routing algorithm. That is, no matter how backed up the queues get, unless the queues exceed saturation or congestion levels the routing algorithm will not change its routing paths.

The format for the 004 card is described in Table A-XVI

The saturation and congestion levels used in Capt. Whittenton's thesis (Ref 4:34), are shown in Table A-XV.

TABLE A-XV

SATURATION AND CONGESTION LEVELS

TABLE A-XVI

CARD TYPE 004 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
SATLEVEL	7-15	Saturation level for this SCM (Node saturation level is determined by summing SATLEVELs of all SCMs within the node).
CNGST1 CNGST2 CNGST3 CNGST4 CNGST5	39-43 44-48 49-53 54-58 59-63	Congestion level for the link corresponding to ICONSCM on the 001 card.
CRDTYPE	78-80	Type of card (004).

Saturation Threshold (1, 2 and 3 SCMs)	7,18,20
Congestion Threshold per trunk	7

These values were divided among the SCMs as shown in the listing of 004 cards for each SCM in Table XVII.

Input Record 900

This card allows the user to simulate a network failure. A line, link, SCM or PSN can be disconnected or connected at a predetermined time during the simulation. As with the previous cards, each card provides information for only one SCM. Also, for each disconnect or connect action a separate card is required. Therefore, two cards are required to bring down an SCM at a time T1 and bring it up again at a time T2; one specifying all lines in the SCM to be brought down at time T1, and the second specifying all lines to be brought up at time T2. To disconnect (or connect) a link may require more than one card because the lines of two or more SCMs within the same PSN may make up that link. To disconnect (or connect) a PSN all lines of all SCMs within the PSN must be disconnected (or connected).

The format of the 900 card is shown in Table A-XVIII. The lines (LNSTATi) on the 900 card reference the lines connecting ISCM to other SCMs in the same sequential order as input on the 001 card. LNSTAT may be coded as 1, -1, 0 or blank, with the following definitions:

TABLE A-XVII

EXAMPLE 004 CARDS

1	1	9	7	7	7	7	7	004
2	1	9	7	7	7	7	7	004
3	2	9	7	7	7	7	7	004
4	2	9	7	7	7	7	7	004
5	3	7	7	7	7	7	7	004
6	3	7	7	7	7	7	7	004
7	3	6	7	7	7	7	7	004
8	4	9	7	7	7	7	7	004
9	4	9	7	7	7	7	7	004
10	5	7	7	7	7	7	7	004
11	6	9	7	7	7	7	7	004
12	6	9	7	7	7	7	7	004
13	7	7	7	7	7	7	7	004
14	8	7	7	7	7	7	7	004
15	8	7	7	7	7	7	7	004
16	8	6	7	7	7	7	7	004

TABLE XVIII

CARD TYPE 900 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
LNSTAT1 LNSTAT2 LNSTAT3 LNSTAT4 LNSTAT5	20-21 22-23 24-25 26-27 28-29	Line Status - LNSTAT(i) must correspond in order with ICONSCM on the 001 card.
LNCHTM1 LNCHTM2 LNCHTM3 LNCHTM4 LNCHTM5	38-43 44-48 49-53 54-58 59-63	the time (in milliseconds) into the simulation run to change the lines status as indicated by the LNSTAT(i) code.
CRDTYPE	78-80	Card type (900).

1. LNSTAT = 1, implies the line is to be connected at time LNCHTM.
2. LNSTAT = -1, implies the line is to be disconnected at time LNCHTM.
3. LNSTAT = 0 or blank, implies no change in the line (LNCHTM is not edited).

LNCHTM is the time (in milliseconds) into the simulation run that LNSTAT is to occur. For LNCHTM to be meaningful, it must be greater than zero and less than the maximum simulation run time.

If the user desires to bring down the Albany (PSN 1) to Norton (PSN 7) link at time 350 milliseconds into the simulation and to bring the link back up at time 500 milliseconds into the simulation, four cards would be required to perform the following actions (reference Figure A-4).

1. Bring down line 1 of SCM 1 of PSN 1 at time 350.
2. Bring down line 1 of SCM 2 of PSN 1 at time 350.
3. Bring up line 1 of SCM 1 of PSN 1 at time 500.
4. Bring up line 1 of SCM 2 of PSN 1 at time 500.

The cards for the above actions would appear as follows:

```
1 1    -1      350              900
2 1    -1      350              900
1 1     1      500              900
2 1     1      500              900
```

An alternate method of doing the same thing would be to disconnect and connect the link at the Norton (PSN 7) end, which has only one SCM. Then the above four actions could be reduced to two actions:

1. Bring down line 1 and line 2 of SCM 13 of PSN 7 at time 350.
2. Bring up line 1 and 2 of SCM 13 of PSN 7 at time 500.

The cards for these actions would appear as follows:

```
13  7    -1 -1    350 350        900
13  7     1  1    500 500        900
```

Input Record 910

This card allows the user to make certain changes in the traffic scenario during a simulation run. For any SCM the following parameters can be redefined at any specified time:

1. General traffic (GNMCHRS).
2. Percentage of short (and long) packets (SHRTPKT).
3. Special traffic (SPMCHRS).
4. Percentage of each precedence (priority) level.

The user's input for the GNMCHRS, SHRTPKT, SPMCHRS and PRECDNC levels must satisfy the same limits imposed on the input variables of the 001 card. In addition, if GNMCHRS,

SHRTPKT or SPMCHRS are zero or blank, no change will occur; that is, these values must be greater than .0001, for a change to occur. Also, the sum of the precedence levels must equal 1.0 for a change to occur.

Allowing these values to change during a simulation run will provide the user with data to analyze the results of:

1. Spikes in traffic at one or more SCMs or PSNs,
2. Increases or decreases in regular or bulk traffic.
3. Changes in priority levels (CATI, CATII, CATIII and CATIV messages).

Once a parameter has been assigned a new value, it will remain at that value unless another card is input to change it again.

The format of the 910 card is given in Table A-XIX.

Cintinuing with the example, if the user wants to investigate the results of a 150 millisecond, 200% spike in General traffic at the Andrews node (node 1), two actions would take place (assume the spike is to begin at 200 milliseconds into the simulation and end at 350 milliseconds):

1. Double the General traffic parameter (GNMCHRS) for SCM 1 and SCM 2 of PSN 1 at time 200.
2. Reinput the original GNMCHRS for SCM 1 and SCM 2 of PSN 1 at time 350.

The cards for the above actions would appear as follows:

```
1  1    17.94          350                910
```

TABLE A-XIX

CARD TYPE 910 FORMAT

FORTRAN Label	Card Columns	Contents
ISCM	1-2	SCM identification number
INODE	3-4	PSN identification number
GNMCHRS	7-16	General traffic originating at this SCM (in millions of characters per hour).
SHRTPKT	17-19	Percentage of short packets originating at this SCM.
SPMCHRS	30-38	Special, or identified, traffic originating at this SCM (in millions of characters per hour).
TIME	39-43	The number of milliseconds into the simulation the action will take place.
PRECDNC	64-66 67-69 70-72 73-75	Four precedence (priority) levels containing the probability of traffic for each SCM.
CRDTYPE	78-80	910

2	1	17.94	350	910
1	1	8.97	500	910
2	1	8.97	500	910

This chapter has discussed how to code the input data for the simulation program. Card types 001, 002, 003 and 004 are used to define the user's network configuration and traffic. Card types 900 and 910 are used to define dynamic changes during the simulation run. Chapter A-3 describes the output files (TAPE1, TAPE2, TAPE3 and TAPE4) generated during the simulation. TAPE4 will be the file of most concern to the user for normal network analyses. Unique, or special, analyses may require changes or reformating of the output files. For this purpose, each of the output files are defined in Chapter A-3.

CHAPTER A-3

DEFINITION OF DATA OUTPUT

Introduction

The AUTODIN II network simulation output listings will be described in this chapter by presenting an explanation of the four disk files used in the program: TAPE1, TAPE2 TAPE3 and TAPE4. Figure A-9 provides an overview of the complete simulation and analysis program. TAPE1 contains a listing of all the Q-GERT output data. TAPE2 and TAPE3 are data files used in reducing and reformating the data records from TAPE1. TAPE4 is the data file used for statistical analysis of the network simulation. Unless the user requires additional or unique analysis of a network, TAPE4 will provide sufficient data, in a usable format, for most statistical analyses. This chapter will provide the user with a description of the output data and sufficient information so that the user may make changes in the data reduction programs.

TAPE1

The TAPE1 disk file contains all output records from the Q-GERT modeling and analysis program. It may be considered

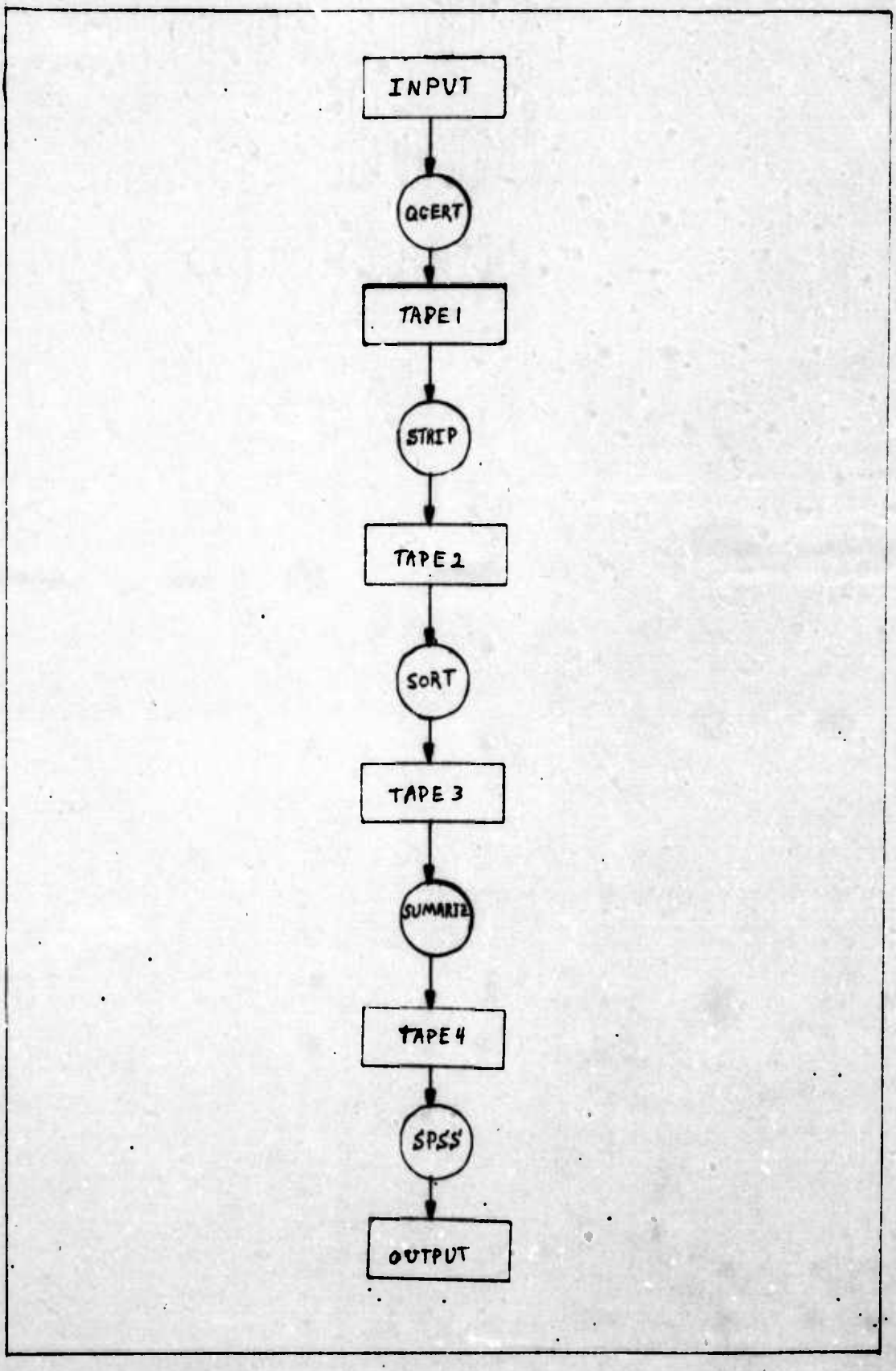


Figure A-9. Simulation Program Overview

to consist of two parts: the Q-GERT summary data and the trace data. The summary data is provided for each simulation run by the Q-GERT program. For an explanation of the summary data, the user should study Pritsker's text (Ref 8:52). The trace data is a chronological listing of all packet activities from the time a packet enters the network to the time it is discarded or delivered from the network. It is important to note the quantity of data that can be generated in the trace data. For the example being followed in this manual, a 45,000 millisecond simulation at 20% of the network capacity produced over 60,000 output lines on TAPE1 (due primarily to the trace data). However, the trace data is very valuable in providing detailed data for the AUTODIN II network analysis. As each packet traverses through the simulated network, each packet's activities are recorded. These activities include: when a packet entered the network, when it entered and left each SCM's input queue, when it entered and left the trunk queue, time spent on the trunk line, and when and where the packet was discarded or delivered. For each activity, values for the following data name variables are written onto TAPE1 (Ref 8 for definitions).

1. Q-GERT START NODE
2. Q-GERT ENDNODE
3. ACTIVITY NUMBER
4. START TIME (milliseconds)

5. END TIME (milliseconds)

6. ACTIVITY NUMBER

7. MARK TIME (milliseconds)

8. TRANSACTION NUMBER

9. PACKET ATTRIBUTES(10):

    1. Packet type: status packet, short data packet or long data packet.

    2. Priority level: system packet, CATI, CATII, CATIII or CATIV.

    3. Source SCM

    4. Destination SCM

    5. Departure SCM (SCM from which the packet will depart this node).

    6. Departure line (line or trunk from which the packet will leave the departure SCM to reach the next SCM).

    7. Processing Time

    8. SCM the packet is presently visiting.

    9. Time, in milliseconds, for the packet to be loaded into the output line or transmitted to the destination SCM.

    10. Code to identify the packets source workload: General or Specific.

The STRIP program (Appendix B) reads TAPE1 and writes the Q-GERT summary data onto the output file for printing and writes the selected packet activities from the Q-GERT trace data onto TAPE2. TAPE1 is then returned.

TAPE2

TAPE2 contains the packet activities that were initiated

at the following Q-GERT nodes:

1. Q-GERT node 8, the beginning of the processor activity.

2. Q-GERT node 10, the beginning of the activity to discard packets or to send packets to another SCM within the same PSN.

3. Q-GERT nodes 16 through 20, the beginning of the activity to transmit packets to another SCM within another PSN.

4. Q-GERT node 21, the beginning of the activity to deliver packets to a host or terminal.

Records containing information about the AUTODIN II network configuration (input by the user on the 001 card) are also selected to provide a cross reference between PSNs and SCMs required in the SUMARIZ program (discussed later).

These selected packet activities contain the SCMs and PSNs that each packet visited in the network. They will also allow for computing the times each packet spent in the input and output queues, the time for SCM processing, whether or not a packet was placed on the PCL, and the time required for loading and transmission on the output trunks. Each packet activity record is in a reduced format as shown in Table A-XX.

TABLE A-XX

TAPE2 RECORD FORMATS

Q-GERT Name	Card Columns
STARTNODE	1-9
STARTTIME	10-18

ENDTIME	19-27
MARKTIME	28-37
TRANSACTION NO.	38-42
ATTRIBUTE(1)	43-50
ATTRIBUTE(2)	51-57
ATTRIBUTE(3)	58-64
ATTRIBUTE(4)	65-71
ATTRIBUTE(5)	72-78
ATTRIBUTE(6)	79-85
ATTRIBUTE(7)	86-92
ATTRIBUTE(8)	93-99
ATTRIBUTE(9)	100-106
ATTRIBUTE(10)	107-113

TAPE2 is then sorted by the CDC 6600 SORTMRG program (see Appendix B) to group the packet activities. The sorted records are placed on the TAPE3 disk file. TAPE2 is then returned.

TAPE3

The TAPE3 file contains the sorted TAPE2 records. Each packet's activities are grouped together and all packet groups are sorted by MARKTIME, that is, the time the packet entered into the network. TAPE3 is in input file for the SUMARIZ program. SUMARIZ reduces the packet groups into a single record of five lines or cards, and writes the records to TAPE4. TAPE3 is then returned.

TAPE4

TAPE4 contains the AUTODIN II network data records to be used by the user for analysis of the AUTODIN II network. Each record provides information about a single packet's flow

through the simulated AUTODIN II network environment. Each record takes five lines or cards and contains: general information about the packet (when it entered the network and the total time in the network) on card 1; and, detailed information about the packet (when it entered the network and the event activities in each PSN it visited) on cards 2 through 5. Fields for all eight PSNs were included in each record (even though a packet did not visit all eight PSNs), to allow data processing by SPSS (to be discussed in the next section). The data on TAPE4 can easily be punched on cards, written to magnetic tape, or placed on permanent file. Also, if the user does not require the detailed information on cards 2 through 5, those cards can be stripped off by the SPSS (or other analysis) program or not written on TAPE4 by a minor change in the SUMARIZ program. TAPE4 record formats are defined in Table A-XXI.

TABLE A-XXI

TAPE4 RECORD FORMATS

CARD 1

SPSS Label	Card Columns	Contents
TIME	1-10	The time (milliseconds) into the simulation run a packet entered its source PSN. This time is the same as MARKTIME in the Q-GERT text.
ATT1	11-14	The type of packet: 1= status

		packet, 2= short packet and 3= long packet.
ATT2	15-18	Packet priority: 5= system priority (for status packets), 10= priority 1 (CATI), 20= priority 2 (CATII), 30= priority 3 (CATIII) and 40= priority 4 (CATIV).
ATT3	19-22	The source SCM, that is, the SCM a packet first entered, and at which time MARKTIME was initiated.
ATT4	23-26	The destination SCM, that is, the SCM a packet last visited before being routed to its destination host or terminal.
SYS	27-30	The type of workload traffic this packet was generated from: 1.0= general traffic and 2.0= special traffic.
HOP	31-34	The number of "hops" a packet takes between its source PSN and destination PSN (hop>0).
TTIME	35-44	The total time (in milliseconds) a packet spent in the network from the time it entered its source PSN to the time it left its destination PSN.

CARD 2

SPSS Label	Card Columns	Contents
X01	1-10	Time the packet spent in the input queue of PSN1.
X02	11-20	Time the packet spent in the output queue of PSN 1.
X03	21-30	Transmission time required for a packet to be placed on the trunk at PSN 1, and transmitted to the input queue of the next PSN.

X04	31-34	The sequential order a packet visited PSN1. For example, if the packet's source PSN was PSN1, the sequence is 1.0; or, if PSN 1 was the second PSN visited, the sequence is 2.0. If the PSN was not visited, the sequence is 0.0.
X05	35-44	Time the packet spent in the input queue of PSN2.
X02	11-20	Time the packet spent in the output queue of PSN2.
X07	55-64	Transmission time required for a packet to be placed on the trunk at PSN2, and transmitted to the input queue of the next PSN.
X08	65-68	The sequential order a packet visited PSN2.

CARD 3

The same as card 2, except use PSN3 and PSN4 and variables X09 through X16.

CARD 4

The same as card 2, except use PSN5 and PSN6 and variables X17 through X24.

CARD 5

The same as card 2, except use PSN7 and PSN8 and variables X25 through X32.

For an example analysis of the data on TAPE4, a listing of SPSS control cards is given in Appendix B. Users may prefer to use other statistical analysis packages or to develop their own analysis program. For an explanation of the SPSS cards, the user should study Ref 9 and 10.

The following section describes a portion of the analysis performed by the SPSS program in this thesis.

SPSS Analysis

This section explains the SPSS program (see Appendix B) used in the example simulation followed in this manual. It is not the intent to redefine the SPSS input and control cards, except as they relate to the analysis methodology. The user should reference the SPSS texts for an explanation of SPSS input cards.

The SPSS VAR LABELS and VALUE LABELS cards, shown in the SPSS listing in Appendix B, redefine the input variables for clarity in the SPSS output products.

The SPSS cards in the program listing (Appendix B) is only one example of the various SPSS analyses available. The simulation was run for 10,500 milliseconds with a dynamic change in network traffic. The SPSS cards for that simulation will be described. The user should realize that these SPSS cards may not be appropriate for other simulations. The time and type of dynamic network configuration and traffic changes will be of primary concern in developing the SPSS

analysis.

To allow the simulation to stabilize, no data was selected from TAPE4 for the first 1000 milliseconds of the simulation. Also, since the simulation time was 10,500 milliseconds, no data was selected after 10,000 milliseconds. This eliminated incomplete data records due to packets not yet reaching their destination node. The following card provided this capability:

```
SELECT IF (TIME GE 1000 AND TIME LE 10000)
```

An overall analysis was desired for two variables: the number of hops required for a packet to reach its destination node (HOP), and the packet delay time (TTIME). The following SPSS card provided the mean, variance, minimum and maximum values, number of cases and other statistical values for the variables:

```
CONDESCRIPTIVE HOP,TTIME
```

The simulation run was set up to dynamically increase general traffic at two levels in the following time intervals:

1. Normal traffic between 1000 and 3000 milliseconds.
2. 130% traffic increase between 3000 and 7000 milliseconds.
3. 150% traffic increase between 7000 and 10,000 milliseconds.

Therefore, the effects of each of these changes were to be investigated. It was decided to develop a table at 250 millisecond intervals and investigate the packet delay times in each of these intervals. Furthermore, each interval was to be divided into partitions for priority levels and types of packets (status, short and long packets). This was accomplished by the following SPSS cards:

```
*COMPUTE TIME=250*TRUNC(TIME/250)

BREAKDOWN TABLES=TTIME BY TIME BY ATT2 BY ATT1
```

For each variable within each interval the following statistics were printed: sum, mean, standard deviation, variance and quantity of cases. Graphs can be developed directly from these tables to display the results.

An analysis of the input queues, output queues and transmission times for each PSN was desired. As noted in the previous section, fields are provided on each input record (TAPE4) for all eight PSNs, even though a packet visits only a few of the PSNs in the network. The PSNs that were not visited contain a zero in the hop sequence (X04 for node 1) variable. For a given PSN, if a record contains a hop sequence of zero, that record should be omitted from the statistics of that particular PSN. The following SPSS cards provide an analysis of the Albany PSN (node 1):

```
*SELECT IF (X04 GT 0)
```

CONDESCRIPTIVE X01 TO X03

The X01 to X03 variables are defined by VAR LABELS card as the input queue time, output queue time and transmission time, respectively. Similar cards would be developed for the other PSNs.

To investigate the effects of the increase in traffic on the input queues, output queues and transmission times, a combination of the cards discussed above will provide the appropriate tables. The interval size is 1000 milliseconds. The following SPSS cards are the cards used for the Albany PSN (node 1):

```
*SELECT IF (X04 GT 0)
*COMPUTE TIME=1000*(TIME/1000)
BREAKDOWN TABLES=X01 TO X03 BY TIME
```

SPSS cards for other nodes would be similar.

It was necessary to perform a statistical test on the traffic variations. Since three variations of traffic were to be tested, the Analysis of Variance (ANOVA) test was used. The design was to test the hypothesis that no difference exists between the average delay times of each of the three traffic levels, versus the null hypothesis that a difference does exist. The SPSS ANOVA test provides a complete printout of results (Ref 9). It is only necessary to define the three traffic levels for the SPSS ANOVA. This was accomplished by defining a new variable, TRAF,

such that: TRAF = 1 for the normal traffic (1000 to 3000 millisecond interval), TRAF = 2 for the 130% traffic increase (3000 to 7000 millisecond interval), and TRAF = 3 for the 150% traffic increase (7000 to 10,000 millisecond interval). The following SPSS cards provide the ANOVA of traffic levels:

```
*COMPUTE TRAF=1
*IF (TIME GE 3000) TRAF=2
*IF (TIME GE 7000) TRAF=3
ANOVA TTIME BY TRAF(1,3)/
```

ANOVA provides for multi variable tests. The user may desire to test other conditions at the same time, such as priority levels or influence of special workloads. The simulation run should be set up to vary these parameters and then they could be tested similar to the discussion just completed.

The possible variations of simulation runs and analyses are too numerous to list. Each variation depends on the objective of the analyst. It is hoped that the discussion of SPSS has provided an insight into the possible ways to analyze the user's network problems.

CHAPTER A-4

AUTODIN II
CONTROL LANGUAGE REQUIREMENTS

Introduction

Chapter A-1 of this manual described the gathering of data for the AUTODIN II model. Chapter A-2 described formatting that data for use by the model. Chapter A-3 described the format and analysis of output from the model through the use of SPSS. This is all necessary information for running the AUTODIN II model.

But not all the information needed to complete a computer run of the AUTODIN II simulation is included in the first three chapters. Another segment of information is necessary to the process. This segment includes the job control language, the necessary Q-GERT control cards, the SORTMRG control cards, and a very basic set of SPSS control cards. This information is tailored for the CDC Cyber series of computers and is presented in the next several sections.

Job Control Language

The job control language (JCL) described here assumes

that the Q-GERT simulation language package and the SPSS package are available as linked object code rather than as source code. The JCL follows:

```
JOB,CM233000,IO750,T750.    T790515,HENDERSON,4331
ATTACH,QGERT,XLRGQGERTSLGO,ID=T800679.
FTN,L=TEMP,SYSEDIT,B=USER.
COPYL,QGERT,USER,LGO.
LDSET,PRESET=ZERO
LGO,,TAPE1.
REWIND,LGO.
REWIND,TAPE1.
FTN.
LGO.
REWIND,TAPE2.
FILE,TAPE2,FO=SQ,BT=C,FL=150,RT=Z.
FILE,TAPE3.FO=SQ,BT=C,FL=150,RT=Z.
SORTMRG.
REWIND,TAPE3.
REWIND,LGO.
FTN.
LGO.
REWIND,TAPE4.
ATTACH,SPSS,ID=AFIT.
SPSS,D=TAPE4.
'eor'
Source code for subroutines UI, UF, US, UO
'eor'
Q-GERT control cards
User parameter cards
'eor'
Source code for STRIP
'eor'
SORTMRG control cards
'eor'
Source code for SUMARIZ
'eor'
SPSS control cards
'eof'
```

Q-GERT Control Cards

The first set of control cards which follow are for one Switch Control Module (SCM) only. After that set, two sets

of the control cards necessary to add Switch Control Modules to the model will be presented.

```
GEN,DIN2SIM,TRAFINC,12,7,1980,0,50,20000,10500,1,E,0,10,1,1,
                                                    ,,,,26,20*
SOU,2,0,1*  CLOCK FOR ALL SCMS
VAS,2,1,UF,0*
ACT,2,2,CO,12.5*
ACT,2,3,US,0*
SIN,3/CLOCK,1,1,D,I*
REG,25,1,1*
ACT,25,26,US,30*
SIN,26/UPDNLINE,1,1,D,I*
DEF,1*  SUBNETWORK FOR SCM #1
SOU,5,0,1,(50)10*    TAC SOURCE NODE
VAS,5,1,UF,1,(50)20*
ACT,5,5,EX,1,(50)30*
ACT,5,7,(50)40*
REG,6,1,1,(50)50*
QUE,7/INPUTBUF,(6)S/2,(50)70*  SCM INPUT QUEUE
ACT,7,8,US,1,7/PROCESSOR,(50)80*
REG,8,1,1,F,(50)90*
ACT,8,9,UF,101,(8)1,A5.LT.-.10,(50)100*
ACT,8,10,(8)2,A6.LE.0.0,(50)110*  LOCAL DELIVERY
ACT,8,11,(8)3,A6.LE.1.,(50)120*   LINE 1
ACT,8,12,(8)4,A6.LE.2.,(50)130*   LINE 2
ACT,8,13,(8)5,A6.LE.3.,(50)140*   LINE 3
ACT,8,14,(8)6,A6.LE.4.,(50)150*   LINE 4
ACT,8,15,(8)7,A6.LE.5.,(50)160*   LINE 5
REG,9,1,1,(50)170*  MAY GO TO NODE 6 OR 8 OF ANY SCM
ACT,9,21,(50)180*
SIN,21/DISCARD,1,1,D,I,(50)190*
SIN,10/LOCALDELIVERY,1,1,D,I,(50)200*
QUE,11/OUTBUFF1,(6)S/2,(50)210*  OUTPUT BUFFER FOR LINE 1
ACT,11,16,UF,102,(50)112*
REG,16,1,1,(50)130*
QUE,12/OUTBUFF2,(6)S/2,(50)240* OUTPUT BUFFER FOR LINE 2
ACT,12,17,UF,102,(50)150*
REG,17,1,1,(50)160*
QUE,13/OUTBUFF3,(6)S/2,(50)270* OUTPUT BUFFER FOR LINE 3
ACT,13,18,UF,102,(50)280*
REG,18,1,1,(50)290*
QUE,14/OUTBUFF4,(6)S/2,(50)300* OUTPUT BUFFER FOR LINE 4
ACT,14,19,UF,102,(50)310*
REG,19,1,1,(50)320*
QUE,15/OUTBUFF5,(6)S/2,(50)330* OUTPUT BUFFER FOR LINE 5
ACT,15,20,UF,102,(50)340*
REG,20,1,1,(50)350*
ESN*
```

If this were the complete set of Q-GERT control cards a FIN* card would follow the ESN*. Control cards for two more Switch Control Modules will be added to the set, however.

```
DUP,2,E*  SUBNETWORK FOR SCM 2
REP,20,VAS,5,1,UF,2*
REP,30,ACT,5,5,EX,2*
REP,80,ACT,7,8,US,2,*
ESN*
DUP,3,E*  SUBNETWORK FOR SCM 3
REP,20,VAS,5,1,UF,3*
REP,30,ACT,5,5,EX,3*
REP,80,ACT,7,8,US,3*
ESN*
FIN*
```

Switch Control Module (SCM) subnetworks can be added up to a total of 25 SCMs, with FIN* signalling the end of the Q-GERT control cards. The number of SCM subnetworks must match the number of Switch Control Modules specified in the user parameter cards. This is important.

SORTMRG Control Cards

The CDC Cyber SORTMRG control cards required to sort the output of STRIP are:

```
SORT
FILE,INPUT TAPE2(CR),OUTPUT TAPE3(CR)
FIELD,SRT3(1,5,DISPLAY),SRT2(18,9,DISPLAY),SRT1(27,16,
                                                DISPLAY)
KEY,SRT1(A,COBOL6),SRT2(A,COBOL6),SRT3(A,COBOL6)
END
```

SPSS Control Cards

The SPSS control cards can be tailored to meet the analy-

sis requirements of the user. The set presented here is very basic and only presented as an example.

```
RUN NAME          AUTODIN II ANALYSIS
VARIABLE LIST     TIME,ATT1 TO ATT4,SYS,HOP,TTIME,
                  X01 TO X32
INPUT FORMAT      FIXED(F10.2,6F4.0,F10.2/2(3F10.2,F4.0)/
                  2(3F10.2,F4.0)/2(3F10.2,F4.0)/2(F10.2,F4.0))
INPUT MEDIAN      DISK
VAR LABELS        TIME,MARKTIME/ATT1,TYPE PF PACKET/
                  ATT2,PRIORITY/ATT3,SOURCE NODE/
                  ATT4,DESTINATION NODE/
                  TTIME,TOTAL TIME IN NETWORK/
                  X01,NODE 1 INPUT QUEUE/
                  X02,NODE 1 OUTPUT QUEUE/
                  X03,NODE 1 TRANSMISSION TIME/
                  X04,NODE 1 HOP SEQUENCE/
                  X05,NODE 2 INPUT QUEUE/
                  X06,NODE 2 OUTPUT QUEUE/
                  X07,NODE 2 TRANSMISSION TIME/
                  X08,NODE 2 HOP SEQUENCE/
                  X09,NODE 3 INPUT QUEUE/
                  X10,NODE 3 OUTPUT QUEUE/
                  X11,NODE 3 TRANSMISSION TIME/
                  X12,NODE 3 HOP SEQUENCE/
                  X13,NODE 4 INPUT QUEUE/
                  X14,NODE 4 OUTPUT QUEUE/
                  X15,NODE 4 TRANSMISSION TIME/
                  X16,NODE 4 HOP SEQUENCE/
                  X17,NODE 5 INPUT QUEUE/
                  X18,NODE 5 OUTPUT QUEUE/
                  X19,NODE 5 TRANSMISSION TIME/
                  X20,NODE 5 HOP SEQUENCE/
                  X21,NODE 6 INPUT QUEUE/
                  X22,NODE 6 OUTPUT QUEUE/
                  X23,NODE 6 TRANSMISSION TIME/
                  X24,NODE 6 HOP SEQUENCE/
                  X25,NODE 7 INPUT QUEUE/
                  X26,NODE 7 OUTPUT QUEUE/
                  X27,NODE 7 TRANSMISSION TIME/
                  X28,NODE 7 HOP SEQUENCE/
                  X29,NODE 8 INPUT QUEUE/
                  X30,NODE 8 OUTPUT QUEUE/
                  X31,NODE 8 TRANSMISSION TIME/
                  X32,NODE 8 HOP SEQUENCE/
VALUE LABELS      ATT1(1)STATUS PACKET(2)SHORT PACKET
                  (3)LONG PACKET/ATT2(5)NETWORK PRIORITY
                  (10)CAT I(20)CAT II(30)CAT III(40)CAT IV/
                  ATT3 TO ATT4(1)ALBANY(2)ANDREWS
                  (3)FT. DETRICK(4)GENTILE(5)HANCOCK
                  (6)MCCLELLAN(7)NORTON(8)TINKER
```

```
SELECT IF        (TIME GE 1000 AND TIME LT 10000)
CONDESCRIPTIVE   HOP,TTIME
*COMPUTE         TIME 250.*TRUNC(TIME/250.)
BREAKDOWN        TABLES TTIME BY TIME BY ATT2 BY ATT1/
                        TTIME BY ATT2
*SELECT IF       (X04 GT 0)
CONDESCRIPTIVE   X01 TO X03
*SELECT IF       (X08 GT 0)
CONDESCRIPTIVE   X05 TO X07
*SELECT IF       (X12 GT 0)
CONDESCRIPTIVE   X09 TO X11
*SELECT IF       (X16 GT 0)
CONDESCRIPTIVE   X13 TO X15
*SELECT IF       (X20 GT 0)
CONDESCRIPTIVE   X17 TO X19
*SELECT IF       (X24 GT 0)
CONDESCRIPTIVE   X21 TO X23
*SELECT IF       (X28 GT 0)
CONDESCRIPTIVE   X25 TO X27
*SELECT IF       (X32 GT 0)
CONDESCRIPTIVE   X29 TO X31
FINISH
```

Summary

The information supplied in this user's manual, with the noted source code and linked object code, will provide capability to run the AUTODIN II simulation model on a CDC Cyber computer with sufficient resources. Adaptation to other computers will require modification to these instructions.

APPENDIX B:

SOURCE LISTING FOR SIMULATION ANALYSIS

```
XXX,STCSA,CM235000,IC750,T750.   ID-NUMBER,NAME,LOCATION
MAP,PAF T.
LIMIT,7500.
ATTACH,OGERT,XLRGOGERTSLGO,ID=T869673.
FTN,L=TEMP,SYSEDIT,B=USER.
COPYL,OGERT,USER,LGO.
LDSET,PRESET=ZERO.
LGO,,T PE1.
REWIND,LGO.
REWIND.TAPE1.
MAP,PA T.
FTN,OL
LGO.
REWIND TAPE2.
RETURN.TAPE1.
FILE,T PE3,FO=SO,BT=C,FL=150,RT=Z.
FILE,T PE2,FO=SO,BT=C,FL=150,RT=Z.
SORTMRG.
REWIND.TAPE3.
REWIND,LGO.
MAP,PA T.
FTN.
LGO.
REWIND,TAPE5.
ROUTE,TAPE5,TID=BB,ST=CSB,FID=JH1,DC=PR.
REWIND,TAPE3.
REWIND.TAPE4.
ATTACH SPSS,ID=AFIT.
SPSS,D=TAPE4.
EXIT,S
DMP,10 250,121000.
7/8/9 OR "EOR
C
C***** ******* ************************************************ *****
C
C     SUBROUTINE UI READS ALL USER INPUT DATA AND INITIALIZES THE
C     THE VARIABLE AND ARRAYS USED BY THE QGERT SIMULATION
```

```
C   PROGRAM.  UI ALSO PERFORMS TWO OF THE BACKGROUND
C   FUNCTIONS  (INITIALIZE-D AND CONNECTIVITY) REQUIRED
C   IN MINHOP.
C
C*****
C
      SUBROUTINE UI
      COMMON/DVAR/NDE,NETBU(500),NREL(500),NRELP(500),NREL2(500),
     1     NRUN,NRUNS,NIC(500),PARAM(100,4),TBEG,TNOW
      COMMON /PARM/ ISTRM,JTRIB(6),NPRMS,IPAR(100),SCALE,IISED(10),SSEED
     X     (10)
      COMMON/UCOM2/LD(8,8),D(8,8,8),NC(8,8),LIST(8,8),HOP(8,8),
     X     ROUTE(8,8),SROUTE(8,8),LINKS(8,8,20),DISTRIB(25,25)
      COMMON /UCOM1/ NSCM, NETDATA(25,16),NODESCM(8,6),NPACK,
     X      DATANET(25,14),LZONE(5),NODE,KSCM,KTICK
      COMMON /UCOM3/ TROUTE(8,8,2), ICLOCK, ATT(10),NRNODES,MXSCMND
      DIMENSION NRSCM(8,2),DISTANC(8,8,8),SCMEDIT(25,16)
      REAL LINCLA(5),LD(8,8),PRECONC(4),LINKDIS(8,8),MSPRHR
      INTEGER TSCM,INODE,INRSCM,ICONSCM(5),CRDTYPE
      LOGICAL PADDATA.
      EQUIVALENCE (LD(1,1),LINKDIS(1,1)),(D(1,1,1),DISTANC(1,1,1))
      DATA NRNODES/0/,ISTICK/0/,NSCM/0/,MXSCMND/5/,CONGSTD/1.01/,CLEAR/1
     X     .01/
      DATA MSPRHR/3.6/,UTSPRSP/600/,ISUPP/8/,INFINIT/1000/
      DATA LINKDIS/64*1000./,NODESCM/48*1000/,HOP/64*1000/,LIST/64*1000/
      DATA NC/64*1000/,DISTANC/512*1000/,NETDATA/400*1000/,DATANET/350*1
     X     000./,LINKS/1280*0/,UPDNNOD/25/,NRSCM/16*0/
      DATA ICLOCK/0/
      PADDATA=.FALSE.
      DO 1000 I=1,8
           NODESCM(I,6)=0
           DO 1000 J=1,8
                DO 1000 K=1,20
1000            LINKS(I,J,K)=0
      DO 1010 I = 1, 25
      NETDATA(I,7) = 7
           DO 1010 J = 1, 5
```

```
1019        NETDATA(I,J+7) = 7                                                  100420
1020  READ (5,93) ISCM,INODE,INRSCM,GNMCHRS,SHRTPKP,(ICONSCM(I),I=1,5),         100430
     X      SPMCHRS,(LINDLA(J),J=1,5),(PRECDNC(L),L=1,4),CRDTYPE               100440
   90 FORMAT(I2,I2,I2,F10.0,F3.0,5I2,F9.0,5F5.0,4F3.0,I5)                      100450
C                                                                             100460
      IF (EOF(5)) 1280,1030                                                   100470
 1030 WRITE (6,96) ISCM,INODE,INRSCM,GNMCHRS,SHRTPKP,(ICONSCM(I),I=1,5),      100480
     X      SPMCHRS,(LINDLA(J),J=1,5),(PRECDNC(L),L=1,4),CRDTYPE               100490
   96 FORMAT(1H ,3I2,F10.10,F6.3,5I2,F17.9,5F8.5,4F6.3,I5)                    100500
      IF ((INODE.LT.1).OR.(INODE.GT.25)) 1270,1040                            100510
 1040 IF (CRDTYPE.EQ.002) 1160,1050                                           100520
 1050 IF (CRDTYPE.EQ.003) 1180,1060                                           100530
 1060 IF (CRDTYPE.EQ.4) 1200,1070                                             100540
 1070 IF (CRDTYPE.EQ.999) GO TO 1220                                          100550
      IF (CRDTYPE.EQ.911) GO TO 1250                                          100560
      IF (CRDTYPE.EQ.001) GO TO 1080                                          100570
      WRITE (6,97)                                                            100580
   97 FORMAT(" INVALID CARD TYPE - PRECEDING CARD.")                          100590
      BADDATA = .TRUE.                                                        100600
      GO TO 1220                                                              100610
 1080 IF ((ISCM.LT.1).OR.(ISCM.GT.25)) 1250,1090                              100620
 1090 IF (NRSCM(INODE,1).EQ.0) NRSCM(INODE,1) =INRSCM                         100630
      NRSCM(INODE,2) = NRSCM(INODE,2) + 1                                     100640
      IF ((NRSCM(INODE,1).NE.INRSCM).OR.(NRSCM(INODE,2).GT.INRSCM))           100650
     X 1460,1110                                                              100660
 1110 NETDATA(ISCM,1) = INODE                                                 100670
      IF (NRNODES.LT.INODE) NRNODES = INODE                                   100680
      IF (NSCM.LT.ISCM) NSCM = ISCM                                           100690
      DATANET(ISCM,1) = GNMCHRS                                               100700
      DO 1120 I=1,5                                                           100710
          NEXTSCM = ICONSCM(I)                                                100720
C  IS ISCM CONNECTED TO ANOTHER SCM                                           100730
          IF (NEXTSCM.LE.0) GO TO 1110                                        100740
          ISCMQ = 15 + I                                                      100750
          ISCMQ = NODCV(ISCMQ,ISCM)                                           100760
          NEXTSCM = NODCV(6,NEXTSCM)                                          100770
          NETEU(ISCMQ) = NEXTSCM                                              100780
```

```
1110        NETDATA(ISCM,I+1) = ICONSCM(I)
1120        DATANET(ISCM,I+1) = LINDLA(I)
      DO 1130  I = 1, 4
1130        DATANET(ISCM,I+6) = PRECONC(I)
      L = 1
      DO 1140  I = 1, MXSCMNO
1140        IF  (NODESCM(INODE,I).NE.1000) L = L + 1
      NODESCM(INODE,L) = ISCM
      NODESCM(INODE,6)=NODESCM(INODE,6)+1
      DATANET(ISCM,11) = SHRTPKP
1150  DATANET(ISCM,12) = SPMCHRS
      TOTPKHR= ((GNMCHRS+SPMCHRS)*8)/((4704*(1.-SHRTPKP))+(600*SHRTPKP))
      DATANET(ISCM,13)=(SPMCHRS*8)/(((1.-SHRTPKP)*TOTPKHR)*4704)
      TPAR(ISCM) = 1
      PARAM(ISCM,2) = 0.0
      PARAM(ISCM,3) = 10000.
      PARAM(ISCM,1) = MSPRHR/TOTPKHR
      GO TO 1020
1160  TOTALPR = 0.0
      DO 1170  I = 1, 4
          SCMEDIT(ISCM,I) = LINDLA(I+1)
          SCMEDIT(ISCM,I+4) = PRECONC(I)
1170  TOTALPR = SCMEDIT(ISCM,I) + SCMEDIT(ISCM,I+4) + TOTALPR
      IF (ABS(TOTALPR - 1.0).LT.0.001)  GO TO 1020
      WRITE (6,94) TOTALPR
   94 FORMAT(21H TOTAL DISTRIBUTION, ,F5.3,21H DOES NOT EQUAL 100%.)
      ADDATA = .TRUE.
      GO TO 1020
1180  DO 1190  I =1, 4
          SCMEDIT(ISCM,I+8) = LINDLA(I+1)
1190      SCMEDIT(ISCM,I+12) = PRECONC(I)
      GO TO 1020
1210  IF (GNMCHRS.NE.0) NETDATA(ISCM,7) = GNMCHRS
      DO 1210  I = 1, 5
1210      IF  (LINDLA(I).NE.0) NETDATA(ISCM,I+7) = LINDLA(I)
      GO TO 1020
1220  ATT(3) = ISCM
```

```
      ATT(4) = 900.                                                     101150
      DO 1240 I = 1, 5                                                  101170
           IF (ICONSCM(I).EQ.0) 1240,1230                               101180
 1230      ATT(1) = ICONSCM(I)                                          101190
           ATT(2) = I                                                   101200
           CALL PTIN(UPDNNOD,LINDLA(I),TNOW,ATT)                        101210
 1240      CONTINUE                                                     101220
      GO TO 1320                                                        101230
 1250 ATT(3) = ISCM                                                     101240
      ATT(4) = 910.                                                     101250
      ATT(5) = GNMCHRS                                                  101260
      ATT(5) = SPMCHRS                                                  101270
       TT(2) = SHRTPKP                                                  101280
       TT(1) = 0.0                                                      101290
       TT(7) = PRECDNC(1)                                               101300
      ATT(8) = PRECDNC(2)                                               101310
       TT(9) = PRECDNC(3)                                               101320
      ATT(10) = PRECDNC(4)                                              101330
      IF (LINDLA(1).GT.0.000001) CALL PTIN(UPDNNOD,LINDLA(1),TNOW,ATT)  101340
      GO TO 1320                                                        101350
 1260 WRITE (6,91) ISCM                                                 101360
   91 FORMAT (11H BAD SCM = ,I3)                                        101370
      BADDATA = .TRUE.                                                  101380
      GO TO 1320                                                        101390
 1270 WRITE (6,92) INODE                                                101400
   92 FORMAT (12H BAD NODE = ,I2)                                       101410
      BADDATA = .TRUE.                                                  101420
      GO TO 1320                                                        101430
C  INITIALIZE A (LINK X LINE) ARRAY FOR EACH NODE.  ALSO SIGNIFY        101440
C  CONNECTIONS IN THE LINK DISTANCE TABLE BY A 1.                       101450
 1290 DO 1320 I = 1, NRNODES                                            101460
           DO 1320 J = 1, NRNODES                                       101470
                TOTALPR = 0                                             101480
                DO 1290 K = 1, MXSCMND                                  101490
                IF (NODESCM(J,K).NE.INFINIT) 1290,1300                  101500
 1290           TOTALPR = SCMEDIT(NODESCM(J,K),I+8) + TOTALPR           101510
 1300           IF (ABS(TOTALPR-1).GT.0.001) 1310,1320                  101520
```

```
1310            WRITE (6,95) I, J, (NODESCM(J,L),L= 1, 5)                    101530
  95 FORMAT(5H NODE ,I2,9H TO NODE ,I2,494 SCMS OUTPUT % DO NOT EQUAL 1      101540
     X 0% (CARR TYPE 103).,/10H SCMS ARE ,5I5)                               101550
            BADDATA = .TRUE.                                                 101560
1320  CONTINUE                                                               101570
      DO 9210 I=1,8                                                          101580
 9210       WRITE(6,92?0)I,I,(NODESCM(I,J),J=1,6)                          101590
 9211       FORMAT(19X," NODESCM",8X,I3,40X,I2,1X,5(3X,I7),/84X,I7)        101600
      WRITE(6,9210)((SCMEDIT(I,J),J=1,7),I=1,7)                              101610
9210  FORMAT(" SCMEDIT ",7F5.3)                                              101620
      IF(BADDATA) 1330,1340                                                  101630
 1330 WRITE(6,9100)                                                          101640
 9100 FORMAT(" BAD DATA IN INPUT - RUN TERMINATED")                          101650
      CALL SYSTEM(52," CHECK DATA")                                          101660
      CALL DUMP(NSCM,NRSCM,0)                                                101670
 1340 DO 1360 I = 1, NSCM                                                    101680
           DO 1360 J = 1, NRNODES                                            101690
                DO 1350 K = 1, MXSCMND                                       101700
                  IF (NODESCM(J,K).NE.1000)  1350,1360                       101710
 1350 DISTRIB(I,NODESCM(J,K)) = SCMEDIT(NODESCM(J,K),(NETDATA(I,1)+8)) *     101720
     X                  SCMEDIT(I,J)                                        101730
 1360                  CONTINUE                                              101740
      WRITE (6,9600)((DISTRIB(I,J),J=1,15),I=1,16)                           101750
      WRITE (6,9610)((SCMEDIT(I,J),J=1,13),I=1,16)                           101760
      WRITE (6,9620)((PARAM(I,J),J=1,3),I=1,16)                              101770
 9600 FORMAT(" DIST ",5F8.3)                                                 101780
 9610 FORMAT(" SCMEDIT ",13F9.4)                                             101790
 9620 FORMAT(" PARAM ",3F9.4)                                                101800
      WRITE(6,9220)((NETDATA(I,J),J=1,10),I=1,7)                             101810
 9220 FORMAT(" NETDATA ",10I6)                                               101820
      DO 1390 I = 1, NSCM                                                    101830
           DO 1390 J = 2, 6                                                  101840
C NETDATA(I,J)  ARE THE SCMS FOR THIS NODE.                                  101850
                L = NETDATA(I,J)                                             101860
C  IF NOT CONNECTED THE INPUT SCM WILL BE ZERO.                              101870
                IF (L.EQ.0) GO TO 1390                                       101880
C  NETDATA(L,1) IS THE DESTINATION NODE THIS NODE IS CONNECTED WITH.          101890
```

```
C  NETDATA(I,1) IS THIS NODE.                                                    101900
      LINKS(NETDATA(I,1),NETDATA(L,1),1) = LINKS(NETDATA(I,1),NETDATA           101910
     X           (L,1),1) + 1                                                   101920
C  M IS THE LINK NUMBER PLUS ONE.                                               101930
           M = LINKS(NETDATA(I,1),NETDATA(L,1),1) + 1                           101940
C  (((J-1) * 100 + I) * 1000) + N * 100 + L ENCODES THE LINE NUMBER             101950
C  AND SCM FOR THIS NODE IN THE TOP THREE POSITIONS OF A FIELD AND THE          101960
C  LINE NUMBER AND SCM OF THE CONNECTED NODE IN THE BOTTOM THREE                101970
C  POSITIONS OF THE SAME FIELD.                                                 101980
C                                                                               101990
           DO 1370 N = 2, 6                                                     102000
              IF (I.EQ.NETDATA(L,N)) 1380,1370                                  102010
 1370         CONTINUE                                                         102020
 1380 LINKS(NETDATA(I,1),NETDATA(L,1),M) = (((J-1)*100+I)*1000)+(              102030
     X        N-1)*100+L                                                       102040
           LINKDIS(NETDATA(I,1),NETDATA(L,1)) =1.                             102050
 1390      CONTINUE                                                           102060
      DO 1400 I = 1,NRNODES                                                   102070
         DO 1400 J = 1,NRNODES                                                102080
      IF (LINKDIS(I,J).NE.LINKDIS(J,I)) CALL DUMP (LD(1,1),LD(8,8),2)         102090
 1400      CONTINUE                                                           102100
      WRITE(5,9300)((LINKDIS(I,J),J=1,8),I=1,8)                               102110
9300  FORMAT(" LD ",8F8.2)                                                    102120
      DO 9305 I=1,8                                                           102130
9305     WRITE(5,9310)((I,(LINKS(I,J,K),K=1,12)),J=1,5)                       102140
9310     FORMAT(" NODE=",I4,"LINKS",12I8)                                     102150
      DO 1430 I = 1,NRNODES                                                   102160
         L = 0                                                                102170
         DO 1420 J = 1,NRNODES                                                102180
            IF (LINKDIS(I,J).EQ.1) 1410,1420                                  102190
 1410       L= L + 1                                                         102200
            NC(I,L) = J                                                       102210
 1420       CONTINUE                                                         102220
 1430 CONTINUE                                                                102230
9320  FORMAT(" NC ",8I6)                                                      102240
      WRITE(5,9320)((NC(I,J),J=1,8),I=1,8)                                    102250
      WRITE (6,9325)((DATANET(I,J),J=1,14),I=1,16)                            102260
```

```
9325 FORMAT(" DATANET ",13F7.2,F11.7)                                  102270
     WRITE (6,9326)                                                    102280
9326 FORMAT(" NODE   SCM  NODCV  NFTBU")                               102290
     DO 1440 J=1,4                                                     102300
          DO 1440 I = 5,21                                             102310
               NCV = NODCV(I,J)                                        102320
               NBU = NFTBU(NCV)                                        102330
1440 WRITE(6,98) I,J,NCV,NBU                                           102340
  98 FORMAT(I5,I6,2I7)                                                 102350
     NODE = 1                                                          102360
     CALL MINHOP                                                       102370
     DO 1451 I = 2, NRNODES                                            102380
          NODE = I                                                     102390
          CALL RTABLE                                                  102400
1451      CONTINUE                                                    102410
     RETURN                                                            102420
1450 WRITE (6,93) INRSCM, NRSCM(INODE,1), INRSCM, NRSCM(INODE,2)       102430
  93 FORMAT (29H INPUT NUMBER SCMS, ,I2,32H SHOULD EQUAL HELD NUMBER   102440
    X CMS, ,I2,/,23H OR INPUT NUMBER SCMS, ,I2,38H SHOULD BE GREATER   102450
    XTHAN SCMS COUNTED, ,I2,1H.)                                       102460
     BADDATA = .TRUE.                                                  102470
      GO TO 1020                                                       102480
     END                                                               102490
C                                                                      102500
C***** ****** ********************************************** ********* 102510
C                                                                      102520
C   FUNCTION UF COMPUTES THE ATTRIBUTES ASSIGNED TO EACH OF            102530
C   THE TRANSACTION TYPES.  UF ALSO CONTAINS THE ENTRY POINT           102540
C   TO MINHOP.                                                         102550
C                                                                      102560
C***** ******* ********************************************** ******* 102570
C                                                                      102580
     FUNCTION UF(ICODE)                                                102590
     COMMON/VAR/NDE,NFTBU(500),NREL(500),NRELP(500),NREL2(500),        102600
    1     NRUN,NRUNS,NTC(500),PARM(100,6),TBEG,TNOW                    102610
     COMMON /PARM/ ISTRM,JTRIB(6),NPRMS,IPAR(100),SCALE,IISED(10),SSEED 102620
    X     (10)                                                         102630
```

```
      COMMON /UCOM1/ NSCM, NETDATA(25,16),NODESCM(8,6),NPACK,
     X       DATANET(25,14),LZONE(8),NODE,KSCM,KTICK
      COMMON /UCOM2/LD(8,8),D(8,8,8),NC(8,8),LIST(8,8),HOP(8,8),
     X      ROUTE(8,8),SROUTE(8,8),LINKS(8,8,2),DISTRIB(25,25)
      COMMON /UCOM3/ TROUTE(8,8,2), ICLOCK, ATT(10),NRNODES,MXSCMND
      DIMENSION DISTANC(8,8,8)
      REAL LINKDIS(8,8),LD(8,8)
      EQUIVALENCE (NRNODES,NNODES)
      EQUIVALENCE (LD(1,1),LINKDIS(1,1)),(D(1,1,1),DISTANC(1,1,1))
      DIMENSION CP(25),VAL(25),IQ(25),CP1(4),VAL1(4)
      INTEGER SCM,TESTQ,TESTSCM,OLDSCM,OLDNODE
      REAL LOCAL(8),BUFF(8),LINES(8)
      DATA LOCAL/8*0./,BUFF/8*0./,LINES/8*0./,NLINES/0/
      DATA BNOM/7./,CNOM/7./,INFI/1000/,TROUTE/128*0./
      DATA INFINIT/1000/,CLEAR/1.00/
C
C***** ****** ********************************************** ****
C
C
C   IF ICODE IS EQUAL TO ZERO, TRANSACTION ATTRIBUTES FOR THE
C   FTICK OR STICK CLOCK SIGNALS ARE ASSIGNED. IF ICODE IS AN
C   SCM IDENTIFIER, THAT IS, IF SCM EQUALS 1, 2, ...,25, TRANS-
C   ACTION ATTRIBUTES FOR THE SIMULATED DATA PACKETS AND STATUS
C   PACKETS ARE ASSIGNED.  IF ICODE EQUAL 11 THE REQUIRED DE-
C   IVERY TO THE NEXT SCM OR HOST/TERMINAL IS DETERMINED. IF
C   ICODE EQUAL 102 THE TRANSMISSION TIME FOR THE PACKET IS
C   DETERMINED.
C
C***** ****** ********************************************** ****
C
      IF=0.0
      CALL GETATT(ATT)
      IF(ICODE.LE.0)GO TO 2000
      IF(ICODE.LE.25)GO TO 2020
      ICODE=ICODE-100
      GO TO (2080,2110)ICODE
C     COMPUTE ATTRIBUTES FOR FTICK/STICK CLOCK SIGNAL TRANSACTION
 2000 CONTINUE
```

```
C      ATT1 IS PACKET TYPE,STATUS=1                                          103010
       ATT(1)=1.0                                                            103020
C      ATT2 IS PRIORTY,STATUS=10                                             103030
       ATT(2) = 3.0                                                          103040
C      ATT3 IS TEMPORARILY A CYCLE OF 25 TO VARY TICK/STICK                  103050
C      CLOCK SIGNALS AMONG THE SCMS.                                         103060
       ICLOCK=ICLOCK+1                                                       103070
       IF (ICLOCK.GT.8) ICLOCK = 1                                           103080
       ATT(3)=ICLOCK                                                         103090
C      THE OTHER ATT ARE NOT DEFINED YET.                                    103100
C      PRINT QUANTITIES IN Q NODES FOR ICLOCK = 1                            103110
       DO 2010 I=1,NSCM                                                      103120
            NNOD=NODCV(7,I)                                                  103130
 2010       IQ(I)=XNINQ(NNOD)                                               103140
       WRITE(6,9930)TNOW,(IQ(I),I=1,NSCM)                                    103150
 9930  FORMAT(" ******* INPUT QUE AT ",F7.2,20I5)                            103160
       CALL PUTATT(ATT)                                                      103170
       RETURN                                                                103180
C      COMPUTE ATTRIBUTES FOR DATA PACKETS.                                  103190
 2020  CONTINUE                                                              103200
C      INITIALIZE VARIABLES AND ARRAYS:                                      103210
C         PLONG IS THE PROBABILITY OF A LONG PACKET                          103220
       SCM=ICODE                                                             103230
       NODE=NETDATA(SCM,1)                                                   103240
       PLONG = 1.00 - DATANET(SCM,11)                                        103250
       DUMY = 0.0                                                            103260
       DO 2030 I=1,NSCM                                                      103270
            VAL(I)=I                                                         103280
            DUMY = DUMY + DISTRIB(SCM,I)                                     103290
 2030       CP(I) = DUMY                                                    103300
       IF (ABS(DUMY-1.0).LT.0.0001) GO TO 2040                               103310
       CP(NSCM) = 1.0                                                        103320
       WRITE (6,9440) (CP(I),I=1,NSCM)                                       103330
 9440  FORMAT(" CP = ",8F8.4)                                                103340
 2040  DUMY = 0.0                                                            103350
       DO 2050 I = 1, 4                                                      103360
            VAL1(I) = (I*10)                                                 103370
```

```
            DUMY = DUMY + DATANET(SCM,I+5)                                  103380
            CP1(I)=DUMY                                                     103390
 2059       CONTINUE                                                        103400
      IF (ABS(DUMY-1.0).LT.0.0001) GO TO 2060                               103410
      CP1(4) = 1.0                                                          103420
      WRITE (6,9410) (CP1(I),I=1,4)                                         103430
 9410 FORMAT(" CP1 = ",4F8.4)                                               103440
C     ATT1 IS THE PACKET TYPE                                               103450
C         SHORT DATA PACKET=2                                               103460
C         LONG DATA PACKET =3                                               103470
 2060 RF=2.0                                                                103480
      ATT(1) = 1.0                                                          103490
       N=DRAND(1)                                                           103500
      IF (RN.GT.PLONG) GO TO 2070                                           103510
      RF = 3.0                                                              103520
       N = DRAND(2)                                                         103530
C     ATT10 IS TYPE  OF PACKET WORKLOAD: 1 IS GENERAL, 2 IS SPECIAL.        103540
       F (RN.LE.DATANET(SCM,13))  ATT(10) = 2.0                             103550
 2070 CONTINUE                                                              103560
C      TT2 IS  PACKET PRIORTY,PRECEDENCE1 IS 10,2 IS 20, 3 IS 30, 4 IS 40   103570
      ATT(2) = DPROB(CP1,VAL1,4,2)                                          103580
C     ATT3 IS  THE SOURCE SCM                                               103590
      ATT(3)=SCM                                                            103600
C      TT4 IS THE DESTINATION SCM (REF. PAGE 251 OF PRITSKER)               103610
      ATT(4)=DPROB(CP,VAL,NSCM,2)                                           103620
C     OTHER ATT ARE NOT YET DEFINED                                         103630
      CALL PUTATT(ATT)                                                      103640
      RETURN                                                                103650
C     FOR PACKETS TO BE DELIVERED TO ANOTHER SCM IN THE SAME                103660
C     NODE, COMPUTE THE SWITCHING NETWORK                                   103670
 2080 CONTINUE                                                              103680
C     COMPUTE  PRESENT SCM(OLDSCM) AND NEXTSCM (THIS NODE) AND              103690
C     CHANGE NEGATIVE ATT5 TO POSITIVE.                                     103700
      OLDSCM=ATT(5)                                                         103710
      OLDNODE=NETDATA(OLDSCM,1)                                             103720
      OLDSCM=NCOCV(9,OLDSCM)                                                103730
       EXTSCM=-ATT(5)                                                       103740
```

```
      ATT(5)=NEXTSCM                                                    103750
      ALL PUTATT(ATT)                                                   103760
C     F LINE =-1.0, PACKETS ARE TO BE DISCARDED AT THIS SCM             103770
      F (ATT(6).GE.-1.0) GO TO 2090                                     103780
      ETBU(OLDSCM)=OLDSCM                                               103790
      ETURN                                                             103800
C     IF PACKET IS A STATUS PACKET DELIVER TO SCM INPUT                 103810
C     UFFER, NODE 5                                                     103820
 2090 F((ATT(1).GE.2.0).OR.(NETDATA(ATT(4),1).NE.OLDNODE))GO TO 2100    103830
      EXTSCM=NODCV(6,NEXTSCM)                                           103840
      ETBU(OLDSCM)=NEXTSCM                                              103850
      ETURN                                                             103860
C     IF PACKET IS A DATA PACKET, DELIVER TO SCM OUTPUT                 103870
C     UFFER, NODE 8                                                     103880
 2100 EXTSCM=NODCV(8,NEXTSCM)                                           103890
      ETBU(OLDSCM)=NEXTSCM                                              103900
      ETURN                                                             103910
C     COMPUTE TIME FOR THE PACKET TO BE TRANSMITTED                     103920
C     TO THE NEXT NODE.                                                 103930
 2110 ONTINUE                                                           103940
      F=ATT(9)                                                          103950
      ETURN                                                             103960
      NTRY UPDAT                                                        103970
      RITE (6,9100)                                                     103980
 9100  ORMAT(" ENTERED UPDATE")                                         103990
      F (KTTICK.EQ.1)2280,2120                                          104000
 2120 ONTINUE                                                           104010
C                                                                       104020
C  *** ******* ********** ********************************** ********** 104030
C  *                                                          *         104040
C  *  NTRY TO MINHOP                                          *         104050
C  *                                                          *         104060
C  *** ******* ********** ********************************** ********** 104070
C                                                                       104080
      NTRY MINHOP                                                       104090
      RITE (6,9102)                                                     104100
 9102 ORMAT(" ENTERED MINHOP")                                          104110
```

```
      DO 2130 I = 1,NRNODES                                                    104120
           DO 2130 J = 1,8                                                     104130
                HOP(I,J) = INFINIT                                             104140
                LIST(I,J) = INFINIT                                            104150
                     DO 2130 K = 1,NRNODES                                     104160
2130 DISTANC(I,J,K) = INFINIT                                                  104170
      DO 2150 I = 1,NRNODES                                                    104180
           HOP(I,I) = 0                                                        104190
           L = 1                                                               104200
           DO 2150 J = 1,NRNODES                                               104210
      IF ((LINKDIS(I,J).GE.CLEAR).AND.(LINKDIS(I,J).LT.INFINIT))2140,          104220
     X          2150                                                          104230
2140       HOP(I,J) = 1                                                       104240
           LIST(I,L) = J                                                       104250
           DISTANC(I,J,L) = 1                                                  104260
2150       DISTANC(I,I,J) = 0.0                                                104270
2160       IF (LINKDIS(I,J).LT.INFINIT) L = L + 1                              104280
      DO 2270 I = 1, NRNODES                                                   104290
           DO 2250 K = 1, NRNODES                                              104300
           IHOPCNT = 1                                                         104310
2170       IALL = 0                                                            104320
2180       DO 2240 J = 1, NRNODES                                              104330
                IF (DISTANC(I,J,K).EQ.IHOPCNT)2190,2240                        104340
2190            M = 1                                                          104350
2200            J1 = NC(J,M)                                                   104360
                IF (J1.EQ.INFINIT)  GO TO 2230                                 104370
                IF (J1.LT.0) GO TO 2220                                        104380
                IF (DISTANC(I,J1,K).GT.IHOPCNT)2210,2220                       104390
2210            DISTANC(I,J1,K) = IHOPCNT+1                                    104400
                IALL = 1                                                       104410
2220            M = M + 1                                                      104420
                GO TO 2200                                                     104430
2230            IHOPCNT = IHOPCNT + 1                                          104440
2240            CONTINUE                                                       104450
           IF (IALL.NE.0)  GO TO 2170                                          104460
2250       CONTINUE                                                            104470
           DO 2270 J = 1 , NRNODES                                             104480
```

```
                IHOPCNT = INFINIT                                                  104490
                DO 2260 K = 1, NRNODES                                             104500
 2260       IF (IHOPCNT.GT. DISTANC(I,J,K)) IHOPCNT = DISTANC(I,J,K)               104510
            HOP(I,J) = IHOPCNT                                                     104520
            DO 2270 K = 1, NRNODES                                                 104530
 2270   IF (DISTANC(I,J,K).GT.(IHOPCNT+1)) DISTANC(I,J,K) = INFINIT               104540
        WRITE(6,9330)((HOP(I,J),J=1,8),I=1,8)                                      104550
 9330   FORMAT(" HOP ",8F8.2)                                                      104560
        DO 9340 I=1,8                                                              104570
 9340       WRITE(6,9350)((I,(DISTANC(I,J,K),K=1,8)),J=1,8)                        104580
 9350   FORMAT(" ",I4," D",8F8.2)                                                  104590
        WRITE(6,9360)((LIST(I,J),J=1,5),I=1,5)                                     104600
 9360   FORMAT(" LIST ",5I7)                                                       104610
C                                                                                  104620
C***** ****** ********************************************                        104630
C                                                                                  104640
C  ENTRY TO MINHOP                                                                 104650
C                                                                                  104660
C***** ****** ********************************************                        104670
C                                                                                  104680
C     RTABLE BUILDS TWO ROUTING TABLES, (1) SROUTE TABLE                           104690
C     IS USED IF CURRENT NODE IS THE SOURCE NODE OF THE                            104700
C     PACKET AND (2) ROUTE TABLE IS USED IF CURRENT NODE                           104710
C     IS A TANDEM NODE FOR THE PACKET.  DATA FROM THE D                            104720
C     TABLE ,OUTBUFFERS AND THE HOP ARRAY ARE USED TO                              104730
C     BUILD THE TWO ROUTING TABLES.  TWO WEIGHTING FACTORS                         104740
C     ARE INTRODUCED TO BIAS AGAINST CONGESTION (BNOM) AND AGAINST                 104750
C     THE USE EXTRA HOP ROUTES (CNOM).                                             104760
C     INITIALIZES  NLN=NUMBER OF LINKS IN NODE                                     104770
C                  LINES(NLN)=NUMBER OF LINES PER LINK                             104780
C                  BUFF(NLN)=NUMBER OF PACKETS IN TOTAL                            104790
C                            OUTPUT LINES IN EACH LINK                             104800
C     SETUP LINES AND BUFF AND NLINKS FROM LINKS ARRAY                             104810
C     ENTRY RTABLE                                                                 104820
      WRITE (6,90) NODE                                                            104830
   90 FORMAT(" ENTERED RTABLE, NODE = ",I5)                                        104840
 2250 NLINKS = 1                                                                   104850
```

```
      DO 2310 I = 1,NNODES                                              104860
           LINES(I) = 0.0                                               104870
           BUFF(I) = 0.0                                                104880
           LOCAL(I) = 0.0                                               104890
           NLINES=LINKS(NODE,I,1)                                       104900
           IF(NLINES.LE.0)GO TO 2310                                    104910
           NLINKS=NLINKS+1                                              104920
C          SETUP LINES(NLINKS)=NUMBER OF LINES IN NLINK                 104930
           LINES(NLINKS)=NLINES                                         104940
C          SETUP BUFF(NLINKS)=TOTAL PACKETS IN OUTPUT                   104950
C          QUEUE OF EACH LINE                                           104960
           NLINES=NLINES+1                                              104970
           DO 2290 J=2,NLINES                                           104980
C          DROP SCM FROM CODE TO GET LINE NO.                           104990
                LINEQUE=LINKS(NODE,I,J)/10000                           105000
C          DROP LINE FROM CODE TO SCM                                   105010
                LINESCM=LINKS(NODE,I,J)/1000-10*LINEQUE                 105020
C          CONVERT LINEQUE TO OGERT LINEQUE                             105030
                LINEQUE = LINEQUE + 10                                  105040
                LINEQUE=NODCV(LINEQUE,LINESCM)                          105050
 2290           BUFF(NLINKS)=BUFF(NLINKS)+XNING(LINEQUE)                105060
 2310      CONTINUE                                                     105070
           WRITE(6,9005)NODE                                            105080
 9005      FORMAT(" NODE=",I5," LINES AND BUFF=")                       105090
           WRITE(6,9006)(LINES(I),I=1,8),(BUFF(I),I=1,8)                105100
 9006      FORMAT(" ",8F8.1)                                            105110
C          FOLLOWING CODE IS FROM SCI DOCUMENT.                         105120
C          COMPUTE CONGESTION WEIGHT AND HOP WEIGHTS.                   105130
           W1=100.*RNOM                                                 105140
           W2=CNOM                                                      105150
C          COMPUTE WEIGHTED BUFFER LENGTHS PER LINE                     105160
           DO 2320 L=1,NLINKS                                           105170
           IF(LINES(L).GT.0.)GO TO 2310                                 105180
           LOCAL(I) = INFINIT                                           105190
           GO TO 2320                                                   105200
 2311      LOCAL(L)=BUFF(L)/LINES(L)                                    105210
 2320      CONTINUE                                                     105220
```

```
C      COMPUTE ROUTE AND SROUTE ARRAYS                                  105230
       DO 2410 N = 1, NRNODES                                           105240
            ROUTE(NODE,N)=0.                                            105250
            SROUTE(NODE,N)=1.                                           105260
            BEST = INFINIT                                              105270
            SBEST = INFINIT                                             105280
            DO 2410 L=1,NLINKS                                          105290
                 ID = INT(DISTANC(NODE,N,L))                            105300
                 IF((ID.LE.HOP(NODE,N)).AND.(ID.LT.INFI))GO TO 2360     105310
 2330            IF((ID.GT.HOP(NODE,N)).AND.(ID.LT.INFI))GO TO 2340     105320
                 GO TO 2410                                             105330
 2340            STEST = LOCAL(L) + W1 * (DISTANC(NODE,N,L)-ID) + W2    105340
                 IF(STEST.LT.SBEST)GO TO 2350                           105350
                 GO TO 2410                                             105360
 2350            SROUTE(NODE,N)=L                                       105370
                 SBEST=STEST                                            105380
                 GO TO 2410                                             105390
 2360            TEST = LOCAL(L) + W1 * (DISTANC(NODE,N,L)-ID)          105400
                 STEST=TEST                                             105410
                 IF(STEST.LT.SBEST)GO TO 2400                           105420
 2390            IF(TEST.LT.BEST)GO TO 2390                             105430
                 GO TO 2330                                             105440
 2390            ROUTE(NODE,N)=L                                        105450
                 BEST=TEST                                              105460
                 GO TO 2330                                             105470
 2400            SROUTE(NODE,N)=L                                       105480
                 SBEST=STEST                                            105490
                 GO TO 2380                                             105500
 2410            CONTINUE                                               105510
      WRITE (5,9000) NODE                                               105520
 9000 FORMAT(" NODE=",I5," ROUTE AND SROUTE")                           105530
      WRITE (5,9001) (ROUTE(NODE,I),I=1,8),(SROUTE(NODE,I),I=1,8)       105540
 9001 FORMAT(" ",8F8.1)                                                 105550
C                                                                       105560
C***** ********** ************************************** *            105570
C                                                                       105580
C   ENTRY TO TROUTE                                                     105590
```

```
C*****
C
C     SET UP TROUTE(8,8,2) ARRAY FOR ROUTING PACKETS
C     C MINIMUM QUEUE LINES WHERE ELEMENTS IN THE
C     ARRAY ARE (1) THE SCM WHICH IS THE SOURCE SCM
C     FOR THE LINE (TRUNK) IN ELEMENT (2).  THE LINE
C     N ELEMENT (2) HAS MINIMUM QUEUE SIZE OF AL-
C     INES IN A LINK FROM NODE1 TO DODE2.
C     NODE1 IS THE SOURCE OR TANDEM NODE
      ODE1=NODE
C     SCM IS THE NUMBER OS SCMS IN NODE1
      SCM1 = NODESCM(NODE1,6)
C     ODE2 IS THE DESTINATION NODE
      O 2430  NODE2 = 1, NRNODES
          TROUTE(NODE1,NODE2,1)=0.0
          TROUTE(NODE1,NODE2,2) = 0.0
          IF(NODE1.EQ.NODE2)GO TO 2430
          MINQ = INFINIT
C     SCM IS THE NUMBER OF SCMS IN NODE1
          DO 2420 I = 1, NSCM1
              SCM=NODESCM(NODE1,I)
              DO 2420 LINE=1,5
C     HECK FOR A LINE GOING TO NODE2
                  TESTSCM=NETDATA(SCM,LINE+1)
                  IF (TESTSCM.LT.0)  GO TO 2420
                  IF(NETDATA(TESTSCM,1).NE.NODE2)GO TO 2420
C     TEST QUEUE SIZE OF TESTSCM, IF LESS THAN PREVIOUS,
C     ASSIGN TO TROUT AS OPTIONAL SCM.  LINEQ IS THE
C     OGERT NODE NUMBER FOR LINE.
                  LINEQUE = LINE + 10
                  TESTSCM=NODCV(LINEQUE,SCM)
                  TESTQ=XNINQ(TESTSCM)
                  IF(TESTQ.GE.MINQ)GO TO 2420
                  TROUTE(NODE1,NODE2,1)=SCM
                  TROUTE(NODE1,NODE2,2)=LINE
                  MINQ=TESTQ
```

```
 2420                  CONTINUE                                                  105970
 2430             CONTINUE                                                       105980
      WRITE(5,9002)NODE                                                         105990
 9002 FORMAT(" NODE=",I6," TROUTE")                                             106000
      WRITE(5,9003)((TROUTE(NODE,I,J),I=1,8),J=1,2)                             106010
 9003 FORMAT(" ",8F8.1)                                                         106020
      RETURN                                                                    106030
      END                                                                       106040
      SUBROUTINE US(IVAL,TIME)                                                  106050
C     ENTRY INTO SUBROUTINE US OCCURS FOR THE PERPACKET FUNCTION OR             106060
C     FOR THE BACKGROUND FUNCTION.  FOR THE PER PACKET FUNCTION THE             106070
C     SCM NUMBER IS REQUIRED (SCM=1,2,...,25).  FOR THE BACKGROUND              106080
C     FUNCTION (FTICK,STICK,TABTICK OR STATUS PACKET) SCM IS CODED ZERO.         106090
C     THE TIME VARIABLE IS THE PROCESSOR TIME TO EXECUTE THE PER                106100
C     PACKET OR BACKGROUND FUNCTIONS.                                           106110
      COMMON/SVAR/NODE,NFTBU(500),NREL(500),NRELP(500),NREL2(500),              106120
     1     NRUN,NRUNS,NTC(500),PARAM(100,4),TREG,TNOW                           106130
      COMMON /PARM/ ISTRM,JTRIB(5),NPRMS,IPAR(100),SCALE,IISED(10),SSEED        106140
     X     (10)                                                                 106150
      COMMON /UCOM1/ NSCM, NETDATA(25,16),NODESCM(8,6),NPACK,                   106160
     X      DATANET(25,14),LZONE(8),NODE,KSCM,KTTICK                           106170
      COMMON /UCOM2/LD(8,8),D(8,8,8),NC(8,8),LIST(8,8),HOP(8,8),                106180
     X     ROUTE(8,8),SROUTE(8,8),LINKS(8,3,20),DISTRIB(25,25)                  106190
      COMMON /UCOM3/ TROUTE(8,8,2), ICLOCK, ATT(10),NRNODES,MXSCMND             106200
      DIMENSION DISTANC(8,8,8)                                                  106210
      EQUIVALENCE (LD(1,1),LINKDIS(1,1)),(D(1,1,1),DISTANC(1,1,1))              106220
      REAL LINKDIS(8,8),MSPRHR,LD(8,8)                                          106230
      LOGICAL LOCHNGE,LINESDN                                                   106240
      INTEGER SCM                                                               106250
      EQUIVALENCE (NRNODES,NNODES)                                              106260
      DATA ISTICK/0/,CONGSTD/1.01/,CLEAR/1.00/,INFINIT/1000/                    106270
      DATA MSPRHR/3.6/                                                          106280
      WRITE (6,9300) IVAL                                                       106290
 9300 FORMAT(" ENTERED US, SCM = ",I2)                                          106300
      CALL GETATT(ATT)                                                          106310
      SCM = IVAL                                                                106320
      XSCM = IVAL                                                               106330
```

```
C                                                                       106340
C     IF IVAL (KSCM) IS EQUAL TO ZERO, PERFORM MESGEN.                  106350
C     IF KSCM IS AN SCM IDENTIFIER, COMPUTE PROCESSOR                   106360
C     TIME AND ENTER THE PAKROUTE MODULE.  IF KSCM                      106370
C     EQUAL 30, PERFORM DYNAMIC CHANGES IN THE AUTODIN                  106380
C     II TRAFFIC OR CONFIGURATION.                                      106390
C                                                                       106400
      IF (KSCM.LE.0)3100,3000                                           106410
 3000 IF(KSCM.LE.25) 3010,3020                                          106420
 3010 ODE = NETDATA(SCM,1)                                              106430
C     COMPUTE PROCESSOR TIME FOR PRIORTY AND NON PRIORTY PACKETS        106440
      TIME = 5.0                                                        106450
      IF(ATT(2).LT.10.1)TIME = 3.0                                      106460
      CALL  PAKROUT                                                     106470
      RETURN                                                            106480
 3020 IF (KSCM.EQ.30) GO TO 3030                                        106490
      CALL DUMP(98,2770038,1)                                           106500
 3030 CALL GETATT(ATT)                                                  106510
      IF (ATT(4).GT.999.) GO TO 3060                                    106520
      ISUBONE = IABS(NETDATA(ATT(3),ATT(2)+1))                          106530
      LINES=LINKS(NETDATA(ATT(3),1),NETDATA(ISUBONE,1),1) + 1           106540
      DO 3040 I = 2, NLINES                                             106550
           LNCNCT = LINKS(NETDATA(ATT(3),1),NETDATA(ISUBONE,1),I)       106560
           LINEHR = LNCNCT/100000                                       106570
           ISCMHR = (LNCNCT-(LINEHR*100000))/1000                       106580
           LINETHR = (LNCNCT-((LINEHR*100000)+(ISCMHR*1000)))/100       106590
      SCMTHR = LNCNCT-((LINEHR*100000)+(ISCMHR*1000)+(LINETHR*100))     106600
      IF ((LINEHR.EQ.IFIX(ATT(2))).AND.(ISCMHR.EQ.IFIX(ATT(3)))) GO TO  106610
     X     3050                                                         106620
 3040      CONTINUE                                                     106630
 3050 NETDATA(ISCMHR,LINEHR+1)=ATT(1)*IABS(NETDATA(ISCMHR,LINEHR+1))    106640
      NETDATA(ISCMTHR,LINETHR+1)=ATT(1)*IABS(NETDATA(ISCMTHR,LINETHR+1)) 106650
      RETURN                                                            106660
 3060 SCM = ATT(3)                                                      106670
      IF (ATT(5).GT.0.0100000001) DATANET(SCM,1) = ATT(5)               106680
      IF (ATT(2).GT.0.001) DATANET(KSCM,11) = ATT(2)                    106690
      IF (ATT(6).GT.0.0100000001) DATANET(KSCM,12) = ATT(5)             106700
```

```
      TOTPKHR= 8.*(DATANET(KSCM,12)+DATANET(KSCM,1))/(47*4.*(1.-
     XDATANET(KSCM,11))+(600.*DATANET(KSCM,11)))
      DATANET(KSCM,13)=(DATANET(KSCM,12)*8.)/((1.-DATANET(KSCM,11))
     X TOTPKHR*4794.)
      PARAM(KSCM,1)=MSPRHR/TOTPKHR
      ATT(2) = 0.0
      DO 3070 I = 7, 10
 3070    ATT (2) = ATT(I) + ATT(2)
      IF ((ABS(1.0-ATT(2))).GT.0.0001) GO TO 3090
      DO 3080 I = 7, 10
 3080    DATANET(KSCM,I) = ATT(I)
 3090 RETURN
C
C     ****************************************************************
C     *                                                              *
C     *  ENTRY TO MESSAGE GENERATION (MESGEN)                        *
C     *                                                              *
C     ****************************************************************
C
 3100 IF ((ICLOCK.EQ.1).AND.(ISTICK.EQ.4)) ISTICK = 0
      IF (ICLOCK.EQ.1) ISTICK = ISTICK +1
      WRITE (6,9101) ICLOCK,ISTICK
 9101 FORMAT(" ENTERED US MESGEN, ICLOCK = ",I2,", ISTICK = ",I2)
      NODE = ICLOCK
      NODE = ICLOCK
      IF (NODESCM(NODE,1).EQ.INFINIT) RETURN
      KSCM = N(DESCM(NODE,1)
      LDCHNGE = .FALSE.
      IF (ISTICK.EQ.4) 3120,3330
C
C     ****************************************************************
C     *                                                              *
C     *  EVERY 400 MILLISECONDS "GOOD NEWS" MUST BE CHECKED.         *
C     *                                                              *
C     ****************************************************************
C
 3120 WRITE (6,9110)
```

```
9110 FORMAT(" ENTERED GOODNEWS")                                        107080
     DO 3270 I = 1, NRNODES                                             107090
          NLINES = LINKS(INODE,I,1) + 1                                 107100
     IF ((LINKDIS(INODE,I).LT.0).AND.(LINKS(INODE,I,1).NE.3))  3130,    107110
    X      3230                                                         107120
3130      DO 3220 J = 2, NLINES                                         107130
               LINEHR = LINKS(INODE,I,J) / 100000                       107140
          ISCMHR = (LINKS(INODE,I,J) - (LINEHR*100000)) / 1000          107150
     LINETHR = (LINKS(INODE,I,J)-((LINEHR*100000)+(ISCMHR*1000)))/100   107160
     ISCMTHR = (LINKS(INODE,I,J)-((LINEHR*100000)+(ISCMHR*1000) +       107170
    X           (LINETHR*100)))                                         107180
               IF (NETDATA(ISCMHR,LINEHR+1).GT.0) 3140, 3160            107190
3140           IF (NETDATA(ISCMTHR,LINETHR+1).GT.0) 3150, 3150          107200
3150      NETDATA(ISCMHR,LINEHR+1) = - NETDATA(ISCMHR,LINEHR+1)         107210
               GO TO 3220                                               107220
3160           IF (NETDATA(ISCMTHR,LINETHR+1).GT.0) 3170, 3220          107230
3170 NETDATA(ISCMTHR,LINETHR+1) = - NETDATA(ISCMTHR,LINETHR+1)          107240
               GO TO 3220                                               107250
3180           LDCHNGE = .TRUE.                                         107260
               LINKDIS(INODE,I) = CONGSTD                               107270
               DO 3210 K = 1, NRNODES                                   107280
                    IF (I.NE.IABS(NC(INODE,K))) 3190, 3210              107290
3190                IF (NC(INODE,K).EQ.INFINIT) 3200, 3220              107300
3200                CONTINUE                                            107310
3210           NC(INODE,K) = IABS(NC(INODE,K))                          107320
3220           CONTINUE                                                 107330
3230      IF (LINKDIS(INODE,I).GT.1.0001) 3240, 3270                    107340
3240      IQUEAMT = 0                                                   107350
          ICNGAMT = 0                                                   107360
          DO 3250 J = 2, NLINES                                         107370
               LINEHR = LINKS(INODE,I,J) / 100000                       107380
          ISCMHR = (LINKS(INODE,I,J) - (LINEHR*100000)) / 1000          107390
               K = NODEV(LINEHR+10,ISCMHR)                              107400
               IQUEAMT = IQUEAMT + XNINO(K)                             107410
3250           ICNGAMT = ICNGAMT + NETDATA(ISCMHR,LINEHR+7)             107420
     IF ((IQUEAMT/(NLINES-1)).LT.(ICNGAMT/(NLINES-1))) 3260, 3270       107430
3260      LDCHNGE = .TRUE.                                              107440
```

```
            LINKDIS(INODE,I) = CLEAR                                          107450
3270        CONTINUE                                                          107460
3280  IQUEAMT = 0                                                             107470
      ISATAMT = 0                                                             107480
      DO 3300 I = 1, MXSCMND                                                  107490
            IF (NODESCM(INODE,I).EQ.1000) 3310,3290                           107500
3290        J = NODECV(7,NODESCM(INODE,I))                                    107510
            IQUEAMT = IQUEAMT + XNING(J)                                      107520
3300  ISATAMT = ISATAMT + NETDATA(NODESCM(INODE,I),7)                         107530
3310  IF ((IQUEAMT.LT.ISATAMT).AND.(LZONE(INODE).EQ.1)) 3320, 3330            107540
3320  LZONE(INODE) = 0                                                        107550
      DCHNGE = .TRUE.                                                         107560
C                                                                             107570
C***********************************************************************     107580
C                                                                     *       107590
C   VERY LOW MILLISECONDS "BAD NEWS" MUST BE CHECKED.                 *       107600
C                                                                     *       107610
C***********************************************************************     107620
C                                                                             107630
3330  WRITE(5,9120)                                                           107640
9120  FORMAT(" ENTERED BADNEWS")                                              107650
      DO 3500 I = 1, NRNODES                                                  107660
            NLINES = LINKS(INODE,I,1) + 1                                     107670
            LINESDN=.TRUE.                                                    107680
      IF ((LINKDIS(INODE,I).LT.1000).AND.(LINKDIS(INODE,I).GT.0)) 3340,       107690
     X      3460                                                             107700
3340        DO 3400 J = 2, NLINES                                             107710
                  LINEHR = LINKS(INODE,I,J) / 100000                          107720
            ISCMHR = (LINKS(INODE,I,J) - (LINEHR*100000)) / 1000              107730
      LINETHR = (LINKS(INODE,I,J)-((LINEHR*100000)+(ISCMHR*1000)))/100        107740
       SCMTHR = (LINKS(INODE,I,J)-((LINEHR*100000)+(ISCMHR*1000)+             107750
     X            (LINETHR*100)))                                            107760
                  WRITE (5,9130) ISCMHR,LINEHR,ISCMTHR,LINETHR                107770
9130  FORMAT(" ISCMHR ",I2," LINEHR ",I2," ISCMTHR ",I2," LINETHR ",I2)       107780
                  IF (NETDATA(ISCMHR,LINEHR+1).LT.0)  3350, 3370              107790
3350              IF (NETDATA(ISCMTHR,LINETHR+1).LT.0) 3400, 3360             107800
3360  NETDATA(ISCMTHR,LINETHR+1) = - NETDATA(ISCMTHR,LINETHR+1)               107810
```

```
                GO TO 3400                                                      107820
3370            IF (NETDATA(ISCMTHR,LINEHR+1).LT.0) 3380, 3390                  107830
3380        NETDATA(ISCMHR,LINEHR+1) = - NETDATA(ISCMHR,LINEHR+1)               107840
                GO TO 3400                                                      107850
3390            LINESON = .FALSE.                                               107860
3400            CONTINUE                                                        107870
            IF (LINESON) 3410, 3460                                             107880
3410        LINKDIS(INODE,I) = - LINKDIS(INODE,I)                               107890
            DO 3430 J = 1, NRNODES                                              107900
                IF (I.NE.NC(INODE,J)) 3420, 3440                                107910
3420            IF (NC(INODE,J).NE.INFINIT) 3430, 3450                          107920
3430            CONTINUE                                                        107930
3440        NC(INODE,J) = - NC(INODE,J)                                         107940
3450        LOCHNGE = .TRUE.                                                    107950
3460    IF ((LINKDIS(INODE,I).GT.0).AND.(LINKDIS(INODE,I).LT.1.001))            107960
     X      3470, 3500                                                         107970
3470        IQUEAMT = 0                                                         107980
            ICNGAMT = 0                                                         107990
            DO 3480 J = 2, NLINES                                               108000
                LINEHR = LINKS(INODE,I,J) / 100000                              108010
            ISCMHR = (LINKS(INODE,I,J) - (LINEHR*100000)) / 1000                108020
                K = NODCV(LINEHR+10,ISCMHR)                                     108030
                IQUEAMT = IQUEAMT + XNINQ(K)                                    108040
3480            ICNGAMT = ICNGAMT + NETDATA(ISCMHR,LINEHR+7)                    108050
    IF ((IQUEAMT/(NLINES-1)).GE.(ICNGAMT/(NLINES-1))) 3490, 3500                108060
3490        LOCHNGE = .TRUE.                                                    108070
            LINKDIS(INODE,I) = CONGSTD                                          108080
3500        CONTINUE                                                            108090
3510    IQUEAMT = 0                                                             108100
        ISATAMT = 0                                                             108110
        DO 3530 I = 1, MXSCMND                                                  108120
            IF (NODESCM(INODE,I).EQ.100) 3530, 3520                             108130
3520        J = NODCV(7,NODESCM(INODE,I))                                       108140
            IQUEAMT = IQUEAMT + XNINQ(J)                                        108150
            ISATAMT = ISATAMT + NETDATA(NODESCM(INODE,I),7)                     108160
3530        CONTINUE                                                            108170
    IF ((IQUEAMT.GE.ISATAMT).AND.(LZONE(INODE).EQ.0)) 3540, 3550                108180
```

```
3560 ! ZONE(INODE) = 1                                                  108190
     ! DCHNGE = .TRUE.                                                 108200
3570 ! F (LZONE(INODE).EQ.1) 3560, 3590                                108210
3580 ! O 3570 I = 1, NRNODES                                           108220
3570 ! F ((LINKDIS(INODE,I).GT.0).AND.(LINKDIS(INODE,I).LT.1.0001))    108230
     X       LINKDIS(INODE,I) = CONGSTD                                108240
3590 ! F (.NOT.LDCHNGE) 3640, 3595                                     108250
3595 ! TICK = 0                                                        108260
C                                                                      108270
C***** ********** ********** ********** ********** **********        108280
C                                                                      108290
C   ENTRY TO UPDATE                                                    108300
C                                                                      108310
C***** ********** ********** ********** ********** **********        108320
C                                                                      108330
      CALL UPDAT                                                       108340
      NEXTNOD = NODCV(8,KSCM)                                          108350
      DO 3630 I = 1, NRNODES                                           108360
         IF (I.NE.INODE) 3600, 3630                                    108370
3600     CALL GETAT(ATT)                                               108380
         ATT(1) = 1.                                                   108390
         ATT(2) = 5.                                                   108400
         ATT(3) = KSCM                                                 108410
         ATT(4) = NODESCM(I,1)                                         108420
         ATT(7) = 0                                                    108430
         CALL PUTAT(ATT)                                               108440
         CALL PAKROUT                                                  108450
         CALL GETAT(ATT)                                               108460
C     ADD TIME TO PROCESSOR IF BUSY, ELSE STORE IN ATT7                108470
         NACT=NATCV(7,KSCM)                                            108480
         NNOD=NODCV(7,KSCM)                                            108490
         IF(ISTUS(NNOD,NACT).GT.0)GO TO 3610                           108500
         ATT(7)=5.0                                                    108510
         GO TO 3620                                                    108520
3610     CALL XTEND(NACT,5.0)                                          108530
         WRITE(6,9139)TNOW,KSCM                                        108540
9139     FORMAT(" XTEND CALLED AT",F8.2,I5)                            108550
```

```
3629        CONTINUE                                                          108550
            CALL PTIN(NEXTNOD,I,TNOW,ATT)                                     108560
3630        CONTINUE                                                          108580
      WRITE (6,9112) NEXTNOD,I,TNOW,(ATT(I),I=1,10)                           108590
9112  FORMAT(" IN US, PTIN ",2I4,11F6.2)                                      108600
      WRITE (6,9140) ((LINKDIS(I,J),J=1,8),I=1,8)                             108610
9140  FORMAT(" LD ",8F8.2)                                                    108620
      WRITE (6,9150) ((NC(I,J),J=1,8),I=1,8)                                  108630
9150  FORMAT(" NC ",8I6)                                                      108640
      RETURN                                                                  108650
3640  KTTICK = 1                                                              108660
3650  CALL UPDAT                                                              108670
      RETURN                                                                  108680
      END                                                                     108690
C                                                                             108700
C     SUBROUTINE UO CONTAINS THE PAKROUTE MODULE                              108710
C                                                                             108720
      SUBROUTINE UO                                                           108730
      COMMON/COMVAR/NDE,NETBU(500),NREL(500),NRELP(500),NREL2(500),           108740
     1     NRUN,NRUNS,NTC(500),PARAM(10,4),TBEG,TNOW                        108750
      COMMON /PARM/ ISTRM,ITRIB(5),NPRMS,IPAR(100),SCALE,IISED(10),SSEED      108760
     X     (10)                                                               108770
      COMMON/UCOM2/LD(8,8),D(8,8,8),NC(8,8),LIST(8,8),HOP(8,8),               108780
     X     ROUTE(8,8),SROUTE(8,8),LINKS(8,8,20),DISTRIB(25,25)                108790
      COMMON /LCOM1/ NSCM, NETDATA(25,15),NODESCM(5,6),NPACK,                 108800
     X      DATANET(25,14),LZONE(8),NODE,KSCM,KTTICK                          108810
      COMMON /LCOM3/ TROUTE(8,8,2), ICLOCK, ATT(10),NRNODES,MXSCMND           108820
      DIMENSION LINKDIS(8,8),DISTANC(8,8,8)                                   108830
      EQUIVALENCE (LD(1,1),LINKDIS(1,1)),(D(1,1,1),DISTANC(1,1,1))            108840
      EQUIVALENCE (NRNODES,NNODES)                                            108850
      INTEGER SCM,ONODE,SNODE                                                 108860
      REAL LINKDIS(8,8),LD(8,8)                                               108870
      REAL LOCAL(8),BUFF(8),LINES(8)                                          108880
      DATA LOCAL/8*0./,BUFF/8*0./,LINES/8*0./,NLINES/0/                       108890
      DATA BNOW/7./,CNOW/7./,INFI/1000/,TROUTE/128*0./                        108900
      WRITE(6,9000)                                                           108910
9000  FORMAT(" XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX")                  108920
```

```
C     THE ABOVE WRITE FLAGS THE END OF THE TRACE PRINT ?????
      RETURN
C
C*****
C
C     ENTRY TO PAKROUTE
C
C*****
C
      ENTRY PAKROUT
      WRITE (6,9110) KSCM,NODE
 9110 FORMAT(" ENTERED PAKROUT, KSCM = ",I5," NODE = ",I5)
      SCM=KSCM
C     PACKROUTE
C     PER PACKET FUNCTION
C     REFERENCE RED NO. 2,P. A-5
C     INITIALIZE
      NEXTNOD = 0
      NLINK2 = 0
4000  DNODE=ATT(4)
      DNODE=NEIDATA(DNODE,1)
      SNODE = ATT(3)
      SNODE=NEIDATA(SNODE,1)
      ATT(8)=SCM
C     TEST IF THIS PACKET IS A LOCAL DELIVERY
      IF(NODE.NE.DNODE) GO TO 4030
      ATT(6)=-1.0
      IF(IFIX(ATT(1)).EQ.1) 4010, 4020
 4010 XTTICK = 0
      CALL UPDAT
      ATT(5)=-10.0
 4020 CONTINUE
      NEXTSCM=ATT(4)
      ATT(5)=NEXTSCM
C     SET ATT5 NEGATIVE IF PACKET IS TO BE DELIVERED
C     TO ANOTHER SCM (SAME NODE) OR IF A STATUS PACKET
      IF((NEXTSCM.NE.SCM).OR.(IFIX(ATT(1)).EQ.1))ATT(5)=-NEXTSCM
```

```
      GO TO 4131                                                        109310
 4031 CONTINUE                                                          109320
C     IF A SOURCE NODE USE SROUTE, IF A TANDEM NODE USE                 109330
C      ROUTE - TO DETERMINE NEXT NODE                                   109340
      IF(SNODE.EQ.NODE)GO TO 4060                                       109350
C     CONVERT OUTGOING LINK TO NEXTNODE                                 109360
      NLINK1 = ROUTE(NODE,DNODE)                                        109370
      IF (NLINK1.LE.0) GO TO 4090                                       109380
      DO 4040 I = 1, NRNODES                                            109390
           IF (LINKS(NODE,I,1).GT.0) NLINK2 = NLINK2 + 1                109400
 4040      IF (NLINK2.EQ.NLINK1) GO TO 53                               109410
      WRITE (6,9116) NLINK1                                             109420
 9116 FORMAT(" NEXTNODE NOT FOUND FROM ROUTE ARRAY, NLINK1.",I4)        109430
      GO TO 4090                                                        109440
   53 NEXTNOD = I                                                       109450
      GO TO 4090                                                        109460
C     CONVERT OUTGOING LINK TO NEXTNODE                                 109470
 4060 LINK1 = SROUTE(NODE,DNODE)                                        109480
      IF (NLINK1.LE.0) GO TO 4090                                       109490
      DO 4070 I = 1, NNODES                                             109500
           IF (LINKS(NODE,I,1).GT.0) NLINK2 = NLINK2 + 1                109510
 4070      IF (NLINK1.EQ.NLINK2) GO TO 4080                             109520
      WRITE (6,9110) NLINK1                                             109530
 9110 FORMAT(" NEXTNODE NOT FOUND FROM SROUTE ARRAY, NLINK1.",I4)       109540
      GO TO 4090                                                        109550
 4080 NEXTNOD = I                                                       109560
C     IF UNABLE TO DELIVER PACKET (NEXTNOD=0),DISCARD PACKET            109570
 4090 IF(NEXTNOD.NE.0)GO TO 4100                                        109580
      ATT(5)=-SCM                                                       109590
      ATT(5) = -10.                                                     109600
      GO TO 4131                                                        109610
C     USE TROUTE TO DETERMINE WHICH TRUNK TO USE TO                     109620
C     DELIVER PACKET                                                    109630
 4100 NEXTSCM=TROUTE(NODE,NEXTNOD,1)                                    109640
      ENTRTIM = 93.4                                                    109650
      IF(ATT(1).LT.2.99) ENTRTIM = 20.1                                 109660
      IF(ATT(1).LT.1.99) ENTRTIM = 15.                                  109670
```

```
C       F NEXTSCM EQ 0,SCM IS NOT CONNECTED - DISCARD                    109670
        F(NEXTSCM.GT.0)GO TO 4110                                        109680
        ATT(5)=-SCM                                                      109690
        ATT(5) = -10.                                                    109700
        GO TO 4130                                                       109710
C       F NEXTSCM EQ THIS SCM, THEN DETERMINE DELIVERY LINES             109720
 4110   F(NEXTSCM.NE.SCM)GO TO 4120                                      109730
        TT(5) = SCM                                                      109740
        TT(5)=TROUTE(NODE,NEXTNOD,2)                                     109750
        TT(6) = (DATANET(KSCM,TROUTE(NODE,NEXTNOD,2)+1)/185.)+ENTRTIM    109760
        GO TO 4130                                                       109770
C       PACKET IS TO BE SENT TO ANOTHER SCM IN THIS NODE                 109780
C        (CODE THE SCM NEGATIVE TO PASS TO OBERT NODE 9)                 109790
 4120   LINE=TROUTE(NODE,NEXTNOD,2)                                      109800
        TT(5)=-NEXTSCM                                                   109810
        ATT(5) = TROUTE(NODE,NEXTNOD,2)                                  109820
        TT(6) = (DATANET(KSCM,TROUTE(NODE,NEXTNOD,2)+1)/185.)+ENTRTIM    109830
 4130   ALL PUTATT(ATT)                                                  109840
        WRITE(5,9007)NODE,SCM                                            109850
9007    FORMAT(" NODE=",I6,"SCM=",I6,"ATT IN PA<ROUTE")                  109860
        WRITE(5,9008)(ATT(I),I=1,6)                                      109870
9008    FORMAT(" ",10F8.2)                                               109880
        RETURN                                                           109890
        END                                                              109900
GEN,DIN2SIM,SPIKE,7,31,1980,0,50,2000*,20,1,E,0,10,1,1,,,,,26,20*        109910
SOU,2, ,1*CLOCK FOR ALL SCMS                                             109920
VAS,2, ,UF,0*                                                            109930
ACT,2, ,CO,12.5*                                                         109940
ACT,2, ,US,0*                                                            109950
SIN,3,CLOCK,1,1,D,I*                                                     109960
REG,25,1,1*                                                              109970
ACT,25,26,US,3*                                                          109980
SIN,26,UPONLINE,1,1,D,I*                                                 109990
DEF,1*SUBNETWORK FOR SCM #1                                              110000
SOU,5, ,1,(50)10*          TAC SOURCE NODE                               110010
VAS,5, ,UF,1,(50)20*                                                     110020
ACT,5, ,EX,1,(50)30*                                                     110030
```

```
ACT,5,7,(5)40*                                                          110040
REG,5, ,1,(50)50*                                                       110050
ACT,6,7,(5)60*                                                          110060
QUE,7/INPUTBUF,(6)S/2,(50)70* SCM INPUT QUEUE                           110070
ACT,7, ,US,1,7/PROCESSOR,(50)80*                                        110080
REG,8,1,1,F,(50)90*                                                     110090
ACT,8,1,UF,101,(8)1,AS.LT.-.10,(50)100*                                 110100
ACT,9,10,(8)2,A6.LE.0.0,(50)110* LOCAL DELIVERY                         110110
ACT,8, 1,(8)3,A6.LE.1.,(50)120* LINE 1                                  110120
ACT,8, 2,(8)4,A6.LE.2.,(50)130*  LINE 2                                 110130
ACT,8, 3,(8)5,A6.LE.3.,(50)140* LINE 3                                  110140
ACT,8,14,(8)6,A6.LE.4.,(50)150* LINE 4                                  110150
ACT,8,15,(8)7,A6.LE.5.,(50)160* LINE 5                                  110160
REG,9, ,1,(50)170*          MAY GO TO NODE 5 OR 8 OF ANY SCM            110170
ACT,9,21,(50)180*                                                       110180
SIN,21/DISCARD,1,1,D,I,(50)190*                                         110190
SIN,10/LOCALDELIVERY,1,1,D,I,(50)200*                                   110200
QUE,11/OUTBUFF1,(5)S/2,(50)210* OUTPUT BUFFER FOR LINE 1                110210
ACT,11.16,UF,102,(50)220*                                               110220
REG,16,1,1,(50)230*                                                     110230
QUE,12/OUTBUFF2,(5)S/2,(50)240* OUTPUT BUFFER FOR LINE 2                110240
ACT,12,17,UF,102,(50)250*                                               110250
REG,17,1,1,(50)260*                                                     110260
QUE,13/OUTBUFF3,(5)S/2,(50)270* OUTPUT BUFFER FOR LINE 3                110270
ACT,13.18,UF,102,(50)280*                                               110280
REG,18,1,1,(50)290*                                                     110290
QUE,14/OUTBUFF4,(5)S/2,(50)300* OUTPUT BUFFER FOR LINE 4                110300
ACT,14.19,UF,102,(50)310*                                               110310
REG,19,1,1,(50)320*                                                     110320
QUE,15/OUTBUFF5,(5)S/2,(50)330* OUTBUFFER FOR LINE 4                    110330
ACT,15,20,UF,102,(50)340*                                               110340
REG,20,1,1,(50)350*                                                     110350
ESN*                                                                    110360
DUP,2,5*                    SUBNETWORK FOR SCM 2                        110370
REP,20.VAS,3,1,UF,2*                                                    110380
REP,30.ACT,5,5,EX,2*                                                    110390
REP,80,ACT,7,8,US,2*                                                    110400
```

```
ESN*
DUP,3,*                          SUBNETWORK FOR SCM 3
REP,2*,VAS,5,1,UF,3*
REP,3*,ACT,5,5,EX,3*
REP,8*,ACT,7,8,US,3*
ESN*
DUP,4,*                          SUBNETWORK FOR SCM 4
REP,2*,VAS,5,1,UF,4*
REP,3*,ACT,5,5,EX,4*
REP,8*,ACT,7,8,US,4*
ESN*
DUP,5,*
REP,2*,VAS,5,1,UF,5*
REP,3*,ACT,5,5,EX,5*
REP,8*,ACT,7,8,US,5*
ESN*
DUP,6,*
REP,2*,VAS,5,1,UF,6*
REP,3*,ACT,5,5,EX,6*
REP,8*,ACT,7,8,US,6*
ESN*
DUP,7,*                          SUBNETWORK FOR SCM 7
REP,2*,VAS,5,1,UF,7*
REP,3*,ACT,5,5,EX,7*
REP,8*,ACT,7,8,US,7*
ESN*
DUP,8,*                          SUBNETWORK FOR SCM 8
REP,2*,VAS,5,1,UF,8*
REP,3*,ACT,5,5,EX,8*
REP,8*,ACT,7,8,US,8*
ESN*
DUP,9,*                          SUBNETWORK FOR SCM 9
REP,2*,VAS,5,1,UF,9*
REP,3*,ACT,5,5,EX,9*
REP,8*,ACT,7,8,US,9*
ESN*
DUP,1*,*                         SUBNETWORK FOR SCM 10
```

```
REP,2?,VAS,5,1 ,UF,1?*                                                  110787
REP,3? ACT,5,? ,EX,1?*                                                  110797
REP,8?,ACT,7,8 ,US,1?*                                                  110807
ESN*                                                                    110817
DUP,11,E*                    SUBNETWORK FOR SCM 11                      110827
REP,2?,VAS,5,1 ,UF,11*                                                  110837
REP,3?,ACT,5,5 ,EX,11*                                                  110847
REP,8?,ACT,7,8 ,US,11*                                                  110857
ESN*                                                                    110867
DUP,12 E*                    SUBNETWORK FOR SCM 12                      110877
REP,2?,VAS,5,1 ,UF,12*                                                  110887
REP,3?,ACT,5,5 ,EX,12*                                                  110897
REP,8?,ACT,7,8 ,US,12*                                                  110907
ESN*                                                                    110917
DUP,13,E*                    SUBNETWORK FOR SCM 13                      110927
REP,2?,VAS,5,1 ,UF,13*                                                  110937
REP,3?,ACT,5,5 ,EX,13*                                                  110947
REP,8?,ACT,7,8 ,US,13*                                                  110957
ESN*                                                                    110967
DUP,14,E*                    SUBNETWORK FOR SCM 14                      110977
REP,2?,VAS,5,1 ,UF,14*                                                  110987
REP,3?,ACT,5,5 ,EX,14*                                                  110997
REP,8?,ACT,7,8 ,US,14*                                                  111007
ESN*                                                                    111017
DUP,15,E*                    SUBNETWORK FOR SCM 15                      111027
REP,2?,VAS,5,1 ,UF,15*                                                  111037
REP,3?,ACT,5,5 ,EX,15*                                                  111047
REP,8?,ACT,7,8 ,US,15*                                                  111057
ESN*                                                                    111067
DUP,16,E*                    SUBNETWORK FOR SCM 16                      111077
REP,2?,VAS,5,1 ,UF,16*                                                  111087
REP,3?,ACT,5,5 ,EX,16*                                                  111097
REP,8?,ACT,7,8 ,US,16*                                                  111107
ESN*                                                                    111117
FIN*                                                                    111127
 1 1 2     8.97.3813 5 6          2182. 645. 645.   1.   ?..51.93        001
 1 1                                  .14  .22  .12  .07.06.04.09.36     002
```

```
 1 1                                          .50  .50  .50  .50.50.50.50.50  003
 1 1        9                              7    7    7    7    7                004
 2 1 2     3.97.381316 7 3                2152. 839. 839. 6 1.   0..01.99      001
 2 1                                      .14  .22  .12  .07.06.04.09.36      002
 2 1                                      .50  .50  .50  .50.50.51.50.50      003
 2 1        9                              7    7    7    7    7                004
 3 2 2     24.00.47 215 5                 500.1298.  50.   0.   0..01.99      001
 3 2                                      .12  .43  .14  .07.12.06.06.10      002
 3 2                                      .50  .50  .50  .50.50.50.50.50      003
 3 2        9                              7    7    7    7    7                004
 4 2 2     24.00.716 710                  1298.  50. 196.   0.   0..01.99      001
 4 2                                      .12  .43  .14  .07.12.06.06.10      002
 4 2                                      .50  .50  .50  .50.50.50.50.50      003
 4 2        9                              7    7    7    7    7                004
 5 3 3      1.71.25 3 114 8                50. 645.1298. 392.   0..01.99      001
 5 3 3                                    .05  .43  .11  .08.06.14.01.22      002
 5 3                                      .33  .33  .33  .33.33.33.33.33      003
 5 3        7                              7    7    7    7    7                004
 6 3 3      1.71.25 910 115              392. 250. 645.1298.   0..01.99      001
 6 3 3                                    .05  .43  .11  .08.06.14.01.22      002
 6 3                                      .33  .33  .33  .33.33.33.33.33      003
 6 3        7                              7    7    7    7    7                004
 7 3 3      1.71.25 910 4 2              392. 250.  50. 645.   0..01.99      001
 7 3 3                                    .05  .43  .11  .08.06.14.01.22      002
 7 3                                      .34  .34  .34  .34.34.34.34.34      003
 7 3        6                              7    7    7    7    7                004
 8 4 2     11.79.49111314 5              2423.2244. 306. 392.   0..01.99      001
 8 4                                      .09  .18  .12  .30.06.12.07.16      002
 8 4                                      .50  .50  .50  .50.50.51.50.50      003
 8 4        9                              7    7    7    7    7                004
 9 4 2     11.79.491218 6 710            2423. 306. 392. 392. 542..01.99      001
 9 4                                      .09  .18  .02  .30.06.12.07.16      002
 9 4                                      .50  .50  .50  .50.50.51.50.50      003
 9 4        9                              7    7    7    7    7                004
10 5 1     15.29.30 9 6 7 4               542. 250. 250. 196.   0..01.99      001
10 5                                      .06  .38  .13  .17.18.05.06.16      002
10 5                                      1.   1.   1.   1.01.01.01.01.0      003
```

```
10 5       7                              7    7    7    7    7              004
11 6 2   8. 14.47 814              2423.1557.   0.   0.   0..01.99           001
11 6                                      .06  .14  .04  .06.05.33.10.22     002
11 6                                      .50  .50  .50  .50.50.50.50.50     003
11 5       9                              7    7    7    7    7              004
12 5 2   8. 14.47 91513            2423.1657. 379.   0.   0..01.99           001
12 5                                      .06  .14  .04  .06.05.33.10.22     002
12 5                                      .50  .50  .50  .50.50.50.50.50     003
12 5       9                              7    7    7    7    7              004
13 7 1   19. 18.24 1 2 812         2192.2132.2244. 373.   0..01.99           001
13 7                                      .15  .25  .31  .37.11.16.12.13     002
13 7                                      1.0  1.0  1.1  1.01.01.01.01.0     003
13 7       7                              7    7    7    7    7              004
14 8 3   17. 10.2711 8 5           1657. 306.1296.   0.   0..01.99           001
14 8                                      .14  .13  .02  .14.08.14.07.28     002
14 8                                      .33  .33  .33  .33.33.33.33.33     003
14 8       7                              7    7    7    7    7              004
15 8 3   17. 10.2712 6 3           1657.1298.1298.   0.   0..01.99           001
15 8                                      .14  .13  .02  .14.08.14.07.28     002
15 8                                      .33  .33  .33  .33.33.33.33.33     003
15 8       7                              7    7    7    7    7              004
16 8 3   17. 10.27 9 4 2            965.1298. 839.   0.   0..01.99           001
16 8                                      .14  .13  .02  .14.08.14.07.28     002
16 8                                      .34  .34  .34  .34.34.34.34.34     003
16 8       6                              7    7    7    7    7              004
15 3      .01.01              17.0 5080                    1.0               910
7/8/9 OR EOR
      PROGRAM STRIP(OUTPUT,TAPE1,TAPE6=OUTPUT,TAPE2)
C      THIS PROGRAM STRIPS ALL TRACE DATA RECORDS FROM QGERT
C     OUTPUT AND (1) PRINTS THE REMAINING QGERT OUTPUT AND (2),
C     SELECTS DATA RECORDS 5, 10 AND 16 THRU 21 NODE NUMBERS
C     TO PROVIDE DATA FOR TRACKING IN THE NETWORK.
      INTEGER DONE
      DIMENSION LINE(27),NODE(9)
      DATA NODE/"     8-","     10-","     15-","     17-","     18-",
     X"     19-","     20-","     21-","NODES0"/
C     DONE=0,  QGERT DATA TO **TRACE**
```

```
C      DONE=1,  TRACE DATA TO QGERT SIMULATION
C      ONE=2,   QGERT DATA TO END OF FILE
       N=0
   11  READ(1,90)(LINE(I),I=1,22)
   90   ORMAT(2A10,A5,A2,2(A3,A9),2A10,A3,A5,5(A3,A7))
        F(EOF(1).NE.1)GO TO 100
       IF((LINE(6).EQ."      ***").AND.(LINE(7).EQ."TRA"))GO TO 20
       WRITE(5,90)(LINE(I),I=1,22)
       GO TO 11
   20  READ(1,90)(LINE(I),I=1,22)
        F(EOF(1).NE.1)GO TO 100
C      ELECT PACKET NODE = 8, 10 AND 16 - 21
C      RITE PACKET DATA TO TAPE2
       IF(LINE(6).EQ."      ")GO TO 2
       DO 40 I=1,9
   40  IF(LINE(3).EQ.NODE(I))GO TO 5
       IF(LINE(2).EQ."%%%%%%%%%%") GO TO 10
       GO TO 20
   59  READ(1,91)(LINE(I),I=23,27)
        F(EOF(1).NE.1)GO TO 100
   91  FORMAT(81X,5(3X,A7))
       WRITE(2,92)LINE(3),(LINE(2*I),I=2,11),(LINE(I),I=23,27)
   92  FORMAT(A6,A2,2A9,A10,A6,10A7)
       GO TO 20
  100  CONTINUE
       STOP
       END
7/8/9 OR *EOR
SORT
FILE,INPUT=TAPE2(CR),OUTPUT=TAPE3(CR)
FIELD,SRT3(1,5,DISPLAY),SRT2(18,9,DISPLAY),SRT1(27,16,DISPLAY)
KEY,SRT1(A,COBOL6),SRT2(A,COBOL6),SRT3(A,COBOL6)
END
7/8/9 OR *EOR
      PROGRAM SUMARIZ(TAPE4,TAPE5,OUTPUT,TAPE6=OUTPUT,TAPE3)
       IMENSION NODESUM(8,6),ATT(10),DATA(8,4),PROCTIM(25,3,2)
      X,PERCENT(3)
```

```
      REAL MARKTIM
      INTEGER TNODE,SCM,SCMNODE(25)
C     THIS PROGRAM SUMMARIZES THE SORTED PACKET DATA OF THE AUTODIN II
C     NETWORK IN TAPE3 TO A SINGLE RECORD ON TAPE4 FOR ANALYSIS.
      DO 5 I=1,25
    5 SCMNODE(I)=0
      DO 10 L=1,25
      DO 10 I=1,3
      DO 10 J=1,2
   10 PROCTIM(L,I,J)=0.0
C     THE FIRST 8 DATA CARDS MUST BUILD THE NODE-SCM ARRAY
      DO 120 I=1,8
      READ(3,900)ITRANS,(ATT(L),L=1,10)
      IF(EOF(3).NE.0)GO TO 400
  900 FORMAT(36X,I5,1X,10F7.0)
      IF(ITRANS.EQ.I)GO TO 100
      WRITE(5,910)
  910 FORMAT(" 900 CARDS MISSING - NO NODE TO SCM ARRAY POSSIBLE",//
     X" PROCESSING STOPPED IN SUMARIZ")
      STOP
  100 DO 110 J=1,5
      IF(IFIX(ATT(J)).LE.25.0)SCMNODE(IFIX(ATT(J)))=I
  110 NODESCM(I,J)=IFIX(ATT(J))
      WRITE(5,920)I,(NODESCM(I,J),J=1,6)
  920 FORMAT("    NODE,",I2," CONTAINS THE FOLLOWING SCMS:",6I8)
  120 CONTINUE
C     BEGIN SUMMARIZING THE SORTED PACKET DATA FOR A COMPLETE TRANSACTION
C     NUMBER INTO A SIMGLE RECORD
  200 READ(3,915)NODE,TIMEIN,TIMEOUT,MARKTIM,ITRANS,(ATT(I),I=1,10)
      IF(EOF(3).NE.0)GO TO 400
  915 FORMAT(I5,3X,2F9.2,F10.2,I5,1X,10F7.2)
  205 DO 210 I=1,8
      DO 210 J=1,4
  210 DATA(I,J)=0.0
      NOP=0
C     CHECK IF FIRST NODE RECORD ENTERING AN SCM IS AN 8 (FOR NODE 8)
  220 IF(NODE.NE.8)225,226
```

```
C    CHECK IF THIS IS A STATUS PACKET
 225 IERR=1
     IF((HOP.NE.0).AND.(IFIX(ATT(1)).NE.1))GO TO 300
     SCM=ATT(8)
     INODE=SCMNODE(SCM)
C    COMPUTE PROCESSOR TIME IN ATT??????
     PROCTIM(SCM,1,1)=PROCTIM(SCM,1,1)+ATT(7)
     PROCTIM(SCM,1,2)=PROCTIM(SCM,1,2)+1.0
     TIME=MARKTIM+ATT(7)
     GO TO 235
 225 CONTINUE
     IF(HOP.EQ.0)TIME=MARKTIM
C    COMPUTE INPUT QUEUE TIME (NODE 7)
     SCM=ATT(8)
     INODE=SCMNODE(SCM)
C    COMPUTE PROCESSING TIME
     PROCTIM(SCM,IFIX(ATT(1)),1)=PROCTIM(SCM,IFIX(ATT(1)),1) +
    X TIMEOUT-TIMEIN
     PROCTIM(SCM,IFIX(ATT(1)),2)=PROCTIM(SCM,IFIX(ATT(1)),2)+1.0
 231 DATA(INODE,1)=TIMEIN-TIME
     TIME=TIMEOUT
C    IS THIS PACKET TO BE TRANSMITTED TO ANOTHER NODE
     READ(3,215)NODE,TIMEIN,TIMEOUT,MARKTIM,ITRANS,(ATT(I),I=1,10)
     IF(EOF(3).NE.0)GO TO 400
C    COMPUTE OUTPUT QUEUE TIME AND TRANSMISSION TIME
 235 IF((NODE.LT.15).OR.(NODE.GT.20))GO TO 240
     DATA(INODE,2)=TIMEIN-TIME
     DATA(INODE,3)=TIMEOUT-TIMEIN
     TIME=TIMEOUT
     HOP=HOP+1
C    COMPUTE HOP SEQUENCE NUMBER
     DATA(INODE,4)=HOP
     ITRANS=ITRANS
     READ(3,215)NODE,TIMEIN,TIMEOUT,MARKTIM,ITRANS,(ATT(I),I=1,10)
     IF(EOF(3).NE.0)GO TO 400
C    IS THIS THE SAME PACKET
     IERR=2
```

```
      IF(ITRANS.EQ.LTRANS)220,300
  240 CONTINUE
C     IS THIS PACKET GOING TO BE DELIVERED/DISCARDED - NODE=10 OR 21
      IF((NODE.NE.21).AND.(NODE.NE.10))GO TO 300
C     COMPUTE TOTAL TIME PACKET IN SYSTEM
      TTIME=TIMEOUT-MARKTIM
C     OUTPUT QUEUE TIME AND TRANSMISSION TIME ARE ZERO
      DATA(INODE,2)=0.0
      DATA(INODE,3)=0.0
      DATA(INODE,4)=HOP+1
      IF((NODE.EQ.21).AND.(ATT(1).GE.2.0))HOP=-10.-HOP
      WRITE(6,934)MARKTIM,(ATT(I),I=1,4),ATT(10),HOP,TTIME
  934 FORMAT(" ",F9.2,6F4.0,F10.2)
      WRITE(4,935)MARKTIM,(ATT(I),I=1,4),ATT(10),HOP,TTIME,((DATA(I,J),
     X J=1,4),I=1,8)
  935 FORMAT(F10.2,6F4.0,F10.2,4(/2(3F10.2,F4.0)))
      GO TO 200
C     CONVERT SOURCE AND DESTINATION SCMS TO NODES
  300 ATT(3)=SCMNODE(IFIX(ATT(3)))
      ATT(4)=SCMNODE(IFIX(ATT(4)))
      WRITE(6,915)NODE,TIMEIN,TIMEOUT,MARKTIM,ITRANS,(ATT(I),I=1,10)
C     ASSIGN IERR=2 IF PACKET DOES NOT PROCESS THRU THE NETWORK
C            IERR=1 IF PACKET IS MISSING A BEGIN NODE 8 (NOT STATUS PACKET)
      WRITE(6,936)IERR
  936 FORMAT(" ABOVE RECORD, ERROR CODE ",I3//)
      IF(IERR.EQ.1)200,205
  410 CONTINUE
C     WRITE OUT PROCESSOR TIME INFORMATION
      DO 430 I=1,25
      IF(SCMNODE(I).EQ.0)STOP
      WRITE(6,940)
  940 FORMAT(14X," PROCESSOR TIME UTILIZATION SCM =",I8,
     X//21X," QUANTITY   PROCESSOR   PERCENT",
     X/21X," OF PACKETS   TIME",//)
      DO 410 J=1,3
      PERCENT(J)=0.0
  410 IF(PROCTIM(I,J,2).GE.1.0)PERCENT(J)=100.*PROCTIM(I,J,1)/TIME
```

```
      WRITE(5,950)((PROCTIM(I,J,2),PROCTIM(I,J,1),PERCENT(J)),J=1,3)
  950 FORMAT(5 ),"  STATUS PACKETS   ",F7.0,3X,F9.2,4X,F5.2,
     X/5X,"  SHORT PACKETS    ",F7.0,3X,F9.2,4X,F5.2,
     Y/5X,"  LONG PACKETS     ",F7.0,3X,F9.2,4X,F5.2,//)
      SUM1=0.0
      SUM2=0.0
      PRCNT=0.0
      DO 420 J=1,3
      SUM1=SUM1+PROCTIM(I,J,1)
  420 SUM2=SUM2+PROCTIM(I,J,2)
      PRCNT=SUM1/TIME
      WRITE(5,960)SUM1,SUM2,PRCNT,TIME
  960 FORMAT(5 ),"  TOTAL",11X,F7.0,3X,F9.2,4X,F5.2)
  430 CONTINUE
      STOP
      END
/*/*/* JOB *EOR
RUN NAME          FORT DIETRICK DOWN
VARIABLE LIST     TIME,ATT1 TO ATT4,SYS,HOP,TTIME,X01 TO X32
INPUT FORMAT      FIXED(F1 .2,6F4.0,F10.2/2(3F10.2,F4.0)/2(3F10.2,F4.0)/
                  2(3F10.2,F4.0)/2(3F10.2,F4.0))
INPUT MEDIAN      DISK
VAR LABELS        TIME,MARKTIME/ATT1,TYPE OF PACKET/ATT2,PRIORITY/
                  ATT3,SOURCE NODE/ATT4,DESTINATION NODE/
                  SYS,TYPE OF WORKLOAD/HOP,NUMBER OF HOPS/
                  TTIME,TOTAL TIME IN NETWORK/
                  X01,NODE 1 INPUT QUEUE/X02,NODE 1 OUTPUT QUEUE/
                  X03,NODE 1 TRANSMISSION TIME/X04,NODE 1 HOP SEQUENCE/
                  X05,NODE 2 INPUT QUEUE/X06,NODE 2 OUTPUT QUEUE/
                  X07,NODE 2 TRANSMISSION TIME/X08,NODE 2 HOP SEQUENCE/
                  X09,NODE 3 INPUT QUEUE/X10,NODE 3 OUTPUT QUEUE/
                  X11,NODE 3 TRANSMISSION TIME/X12,NODE 3 HOP SEQUENCE/
                  X13,NODE 4 INPUT QUEUE/X14,NODE 4 OUTPUT QUEUE/
                  X15,NODE 4 TRANSMISSION TIME/X16,NODE 4 HOP SEQUENCE/
                  X17,NODE 5 INPUT QUEUE/X18,NODE 5 OUTPUT QUEUE/
                  X19,NODE 5 TRANSMISSION TIME/X20,NODE 5 HOP SEQUENCE/
                  X21,NODE 6 INPUT QUEUE/X22,NODE 6 OUTPUT QUEUE/
```

```
                X23,NODE 5 TRANSMISSION TIME/X24,NODE 6 HOP SEQUENCE/
                X25,NODE 7 INPUT QUEUE/X26,NODE 7 OUTPUT QUEUE/
                X27,NODE 7 TRANSMISSION TIME/X28,NODE 7 HOP SEQUENCE/
                X29,NODE 8 INPUT QUEUE/X30,NODE 8 OUTPUT QUEUE/
                X31,NODE 8 TRANSMISSION TIME/X32,NODE 8 HOP SEQUENCE/
VALUE LABELS    ATT1(1)STATUS PACKET(2)SHORT PACKET(3)LONG PACKET/
                ATT2(9)NETWORK PRIORITY(10)CAT I(20)CAT II(30)CAT III
                (40)CAT IV/ATT3 TO ATT4(1)A_BANY(2)ANDREWS(3)FT. DETRICK
                (4)GENTILE(5)HANCOCK(6)MCCLELLAN(7)NORTON(8)TINKER
SELECT IF       (TIME GE 1000 AND TIME LT 1200 AND HOP GE 0)
CONDESCRIPTIVE  HOP,TTIME
*COMPUTE        TIME=250.*TRUNC(TIME/250.)
BREAKDOWN       TABLES=TTIME BY TIME BY ATT2 BY ATT1/
                       TTIME BY ATT2
*COMPUTE        TIME=100.*TRUNC(TIME/100.)
BREAKDOWN       TABLES=TTIME BY TIME BY ATT2/
                       TTIME BY TIME BY ATT2 BY ATT1/
                       TTIME BY HOP BY ATT1/
                       TTIME BY HOP/
                       TTIME BY TIME BY HOP
*SELECT IF      (X04 GT 0)
CONDESCRIPTIVE  X01 TO X03
*SELECT IF      (X08 GT 0)
CONDESCRIPTIVE  X05 TO X07
*SELECT IF      (X12 GT 0)
CONDESCRIPTIVE  X09 TO X11
*SELECT IF      (X16 GT 0)
CONDESCRIPTIVE  X13 TO X15
*SELECT IF      (X20 GT 0)
CONDESCRIPTIVE  X17 TO X19
*SELECT IF      (X24 GT 0)
CONDESCRIPTIVE  X21 TO X23
*SELECT IF      (X28 GT 0)
CONDESCRIPTIVE  X25 TO X27
*SELECT IF      (X32 GT 0)
CONDESCRIPTIVE  X29 TO X31
*SELECT IF      (ATT1 EQ 2)
```

```
*COMPUTE        PERCNT=1
*IF             (TIME GE 2000)PERCNT=2
*IF             (TIME GE 3000)PERCNT=3
*COMPUTE        TRAF=1
*IF             (TIME GE 2000)TRAF=2
*IF             (TIME GE 3000)TRAF=3
*COMPUTE        PRI=1
*IF             (ATT2 EQ 10)PRI=2
*IF             (ATT2 EQ 20)PRI=3
*IF             (ATT2 EQ 30)PRI=4
*IF             (ATT2 EQ 40)PRI=5
ANOVA           TTIME BY PERCNT(1,3)/
                TTIME BY PERCNT(1,3),PRI(1,5)/
                TTIME BY PERCNT(1,3),PRI(1,5),TRAF(1,3)/
*SELECT IF      (ATT1 GE 2)
*COMPUTE        PERCNT=1
*IF             (TIME GE 2000)PERCNT=2
*IF             (TIME GE 3000)PERCNT=3
*IF             (TIME GE 4000)PERCNT=1
*IF             (TIME GE 5000)PERCNT=2
*IF             (TIME GE 6000)PERCNT=3
*IF             (TIME GE 7000)PERCNT=1
*IF             (TIME GE 8000)PERCNT=2
*IF             (TIME GE 9000)PERCNT=3
*COMPUTE        TRAF=1
*IF             (TIME GE 4000)TRAF=2
*IF             (TIME GE 7000)TRAF=3
*COMPUTE        PRI=1
*IF             (ATT2 EQ 10)PRI=2
*IF             (ATT2 EQ 20)PRI=3
*IF             (ATT2 EQ 30)PRI=4
*IF             (ATT2 EQ 40)PRI=5
ANOVA           TTIME BY PERCNT(1,3)/
                TTIME BY PERCNT(1,3),TRAF(1,3)/
                TTIME BY PERCNT(1,3),PRI(1,5)/
*COMPUTE        TIME=1000.*TRUNC(TIME/1000.)
*SELECT IF      (HOP GE 2)
```

```
BREAKDOWN       TABLES=TTIME BY TIME BY HOP/
                       TTIME BY TIME BY HOP BY ATT1
*SELECT IF      (TIME GE 2000)
*COMPUTE        TIME=1000.*TRUNC(TIME/1000.)
BREAKDOWN       TABLES=TTIME BY TIME BY ATT1/
                       TTIME BY HOP BY ATT1/
                       TTIME BY HOP
*SELECT IF      (SYS GE 2)
*COMPUTE        TIME=1000.*TRUNC(TIME/1000.)
BREAKDOWN       TABLES=TTIME BY TIME BY ATT2 BY ATT1/
                       TTIME BY HOP BY ATT2 BY ATT1/
                       TTIME BY TIME BY HOP/
                       TTIME BY ATT2 BY ATT1
*SELECT IF      (ATT3 EQ 15)
*COMPUTE        TTIME=1000.*TRUNC(TIME/1000.)
BREAKDOWN       TABLES=TTIME BY ATT2 BY ATT1/
                       TTIME BY HOP BY ATT2 BY ATT1
*SELECT IF      (ATT3 EQ 15)
T-TEST          GROUPS=TIME(5000)/VARIABLES=TTIME
*SELECT IF      (HOP GE 2)
T-TEST          GROUPS=TIME(5000)/VARIABLES=TTIME
FINISH
```

VITA

James W. Healy was born [redacted] He graduated from [redacted] in 1958. He later attended Brigham Young University from which he received the Bachelor of Science degree in Mathematics/Statistics in 1967 and the Master of Science degree in Mathematics/Statistics in 1968. He worked as an Operations Analyst with Autonetics Division of Rockwell International Corporation for two years. In 1970 he began employment with the U. S. Air Force as an Actuary at McClellan Air Force Base, Sacramento, California. In 1976 he was assigned to supervise the Data Analysis Section at McClellan until entering the School of Engineering, Air Force Institute of Technology, in July, 1979.

PII Redacted

Permanent address: [redacted]

PII Redacted

PII Redacted

Denzel H. Henderson was born [redacted] He graduated from [redacted] in June, 1953. He received a Bachelor of Science degree in Mathematics from Wright State University, Dayton, Ohio in June, 1972, and a Masters of Business Administration in financial management also from Wright State University in August, 1976. Mr. Henderson began a career in data processing at Tinker Air Force Base, Oklahoma City, Oklahoma during July, 1957 and moved to Headquarters Air Force Logistic Command (AFLC), Wright-Patterson AFB, Dayton, Ohio during January 1961. He has held positions up to and including lead programmer and lead analyst in several data processing organizations and was for four years a systems programmer in the Standard Operating Systems Office at Headquarters AFLC. While employed in the Office of the Assistant for Logistics Management Systems he was assigned to attend the School of Engineering, Air Force Institute of Technology in October, 1979.

Permanent address: [redacted]

PII Redacted

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/80D-9	2. GOVT ACCESSION NO. AD-A100 788	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SIMULATION AND ANALYSIS OF AUTODIN II NETWORK DESIGN		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James W. Healy Denzel Henderson		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Engineering Center (DCEC) Reston, Virginia Logistics Air Staff for Data Automation Pentagon, Washington, D.C.		12. REPORT DATE December 1980 13. NUMBER OF PAGES 271
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Approved for public release; destination unlimited		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

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

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17
Fredric C. Lynch, Major, USAF
Director of Public Affairs
16 JUN 1981

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Communications Network
AUTODIN II
Routing Algorithm
Packet - Switching

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This thesis report presents a discussion of the AUTODIN II system, describes a simulation of the AUTODIN II backbone network and presents an analysis of two aspects of the AUTODIN II system. A general overview of the AUTODIN II topology and architecture is presented. A detailed discussion of the present state of the AUTODIN II routing algorithm, it's function, modules and information structures is presented. A discrete packet simulation is developed with dynamic capabilities to redefine the network configuration or workload. The

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

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

simulation program inputs and outputs are described to provide future AUTODIN II system analysts the capability of defining their particular requirements. A users manual is also developed. A presentation of the analyses and results of opposite views is given; (1) from the system designer's view of overall AUTODIN II capability and (2) from the user's view of what the effects will be for a specific AUTODIN II user. The use of the simulation model by other organizations for analysing management problems and concerns is encouraged.