USAFA-TR-75-1

DYNAMIC COMMUNICATION NETWORK SIMULATOR (NETSIM)

CAPT DONALD E. WILLIS

DEPARTMENT OF ASTRONAUTICS
AND
COMPUTER SCIENCE
USAF ACADEMY, COLORADO 80840

JANUARY 1975

FINAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

Prepared for

AIR FORCE WEAPONS LABORATORY
KIRTLAND AFB, NM 87115

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO 80840

Editorial Review by Lt Colonel W. A. Belford, Jr
Department of English and Fine Arts
USAF Academy, Colorado 80840

This research report is presented as a competent treatment of the subject, worthy of publication. The United States Air Froce Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the author.

This report has been cleared for open publication and/or public release by the appropriate Office of Information in accordance with AFR 190-17 and DODD 5230.9. There is no objection to unlimited distribution of this report to the public at large, or by DDC to the National Technical Information Service.

This report has been reviewed and is approved for publication.

Philip J. Erdle

PHILIP J. ERDLE, Colonel, USAF
Vice Dean of the Faculty

Additional copies of this document are available through the National Technical Information Service, U. S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22151.

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California 90291

13 October 1975

To Whom It May Concern:

The NETSIM model as originally developed and described in this paper has been modified extensively. The following improvements have been added:

a. Antenna gain patterns can be input along with static antenna pointing angles. The model will utilize this data to calculate the appropriate transmitter and receiver antenna gains for each specific data link as a function of network geometry for each transmission event. The gain patterns may be modified as a function of time.

b. The signal strength loss as a function of distance and transmission frequency is now taken into account on each communications link.

c. The effects of transmitter power and receiver sensitivity are taken into account on each communications link.

d. Coverage limitations due to terrain blockage may be input for ground stations.

e. Channelization may now be input for each node. Both voice and data links may be specified. Transmission frequency and data rate for digital channels can be chosen.

f. The "sounding rocket" has been added as a node type (other node types are satellite, aircraft, ground and balloon). Burnout velocity, burnout altitude and burnout time must be input for each node of this type.

g. Model output has been modified to show both qualitative and quantitative measures of network performance at each node. Articulation Index is given for voice channels and bit error rate for data channels for all user nodes and all messages. In addition, the first and last times of message receipt are given for each node, which when the network is saturated with messages, yields complete network coverage as a function of time.

The above modifications were made under Air Force Weapons Laboratory contract to R & D Associates, Marina del Rey, California. Points of contact for comments and questions concerning the model are:

Capt. Lamar P. Craig
AFWL/ELE
Kirtland AFB, NM

or

Donald E. Willis
R & D Associates
P.O. Box 9695
Marina del Rey, CA 90291
Telephone (213) 822-1715

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER USAFA-TR-75-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Dynamic Communication Network Simulator (NETSIM)		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER USAFA-TR-75-1
7. AUTHOR*(s)* Capt Donald E. Willis		8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Astronautics and Computer Science (DFACS) US Air Force Academy, CO 80840		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Dean of Faculty, US Air Force Academy, Colorado and Air Force Weapons Laboratory Kirtland AFB, NM 87115		12. REPORT DATE January 1975 13. NUMBER OF PAGES 29
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

Network Simulation
Communication
Adaptive Network
Distributed Network
Computer Simulation

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

Advances in communication hardware have made possible the implementation of adaptive distributed networks of time variant topology. Static network simulation models have been incapable of predicting the dynamic effects on overall network performance of changes in linking, routing, and message processing algorithms. A dynamic model is described which will simulate performance of distributed heuristic networks to microsecond accuracy. It will accept as parameters user defined subroutines for message

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

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

routing, geographic nodal movement, heuristic linking, message processing and noise generation. Written in Extended ALGOL the model makes use of independent task processing capabilities of the Burroughs B6700 computer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Advances in communication hardware have made possible the implementation of adaptive distributed networks of time variant topology. Static simulation network models have been incapable of emulating dynamic effects of changes in linking, routing, and message processing algorithms on overall network performance.

This paper describes an event triggered network simulation model written in Extended ALGOL for the Burroughs B6700 computer. It uses the B6700 independent task processing capabilities as an integral feature of the simulation structure. The model was created to simulate store-and-forward networks of changing topology, but can be adapted to simulate circuit switching systems as well. Moveable nodes, multiple messages, programmed or random message insertion, variable message length, variable message process time, and varied message routing disciplines are but a few of the characteristics of this simulator. Virtually any program parameter may be drawn, monte carlo fashion, from a versatile set of probability distributions.

The following description is designed to give the user sufficient insight into the ALGOL structure so that he might understand the basic design of the program and so that he

might appreciate the versatility of this simulator. The program, although written entirely in ALGOL, incorporates ideas suggested by the SIMULA simulation language and therefore is not trivial. Without a fundamental knowledge of ALGOL, the reader will be handicapped in following this discussion.

INDEPENDENT TASK CONTROL

Normally when a procedure (a subprogram analogous to the FORTRAN subroutine) is called, the following events take place:

1. The main program is temporarily suspended.
2. The subprogram is activated and runs as an independent task working on its local data structure.
3. The subprogram completes.
4. The subprogram data structure is eliminated (it will be reinitialized if and when the subroutine is called again).
5. The main program is reactivated and processing continues.

The ALGOL language as implemented on the Burroughs B6700 computer allows this typical subprogram calling sequence to be altered significantly. (See Figure 1.)

1. A "task identifier" may be associated with a subprogram when it is called.
2. By use of the CALL verb the subprogram may be run as a task completely independent of the main program.

3. By use of the CONTINUE verb the subprogram may be interrupted (passing control back to a main program or another subroutine) without destroying its local data structure.

4. The subprogram can later be re-entered and continue processing as if it had never been interrupted.

These language constructs allow the programmer to create a very powerful and versatile simulation structure. If one views a subprogram as a process, multiple copies of that process (each with a unique task identifier) can be created and run independently. They can pass control and data back and forth as if they were independent processing nodes in a communication network, which, in a software sense, they indeed are. In NETSIM the NODE subroutine will be used in this way with multiple copies being created and functioning as independent nodes. The NODE subroutine is an ALGOL procedure created by the user and therefore can incorporate complicated logic to process messages, route messages, move from one geographical position to another, etc. The full power of the Extended ALGOL language is available to the user in creating this procedure.

The remainder of this paper consists of a detailed explanation of model constructs, their capabilities and limitations, and an example of a specific network model to illustrate the flexibility and versatility of the model.

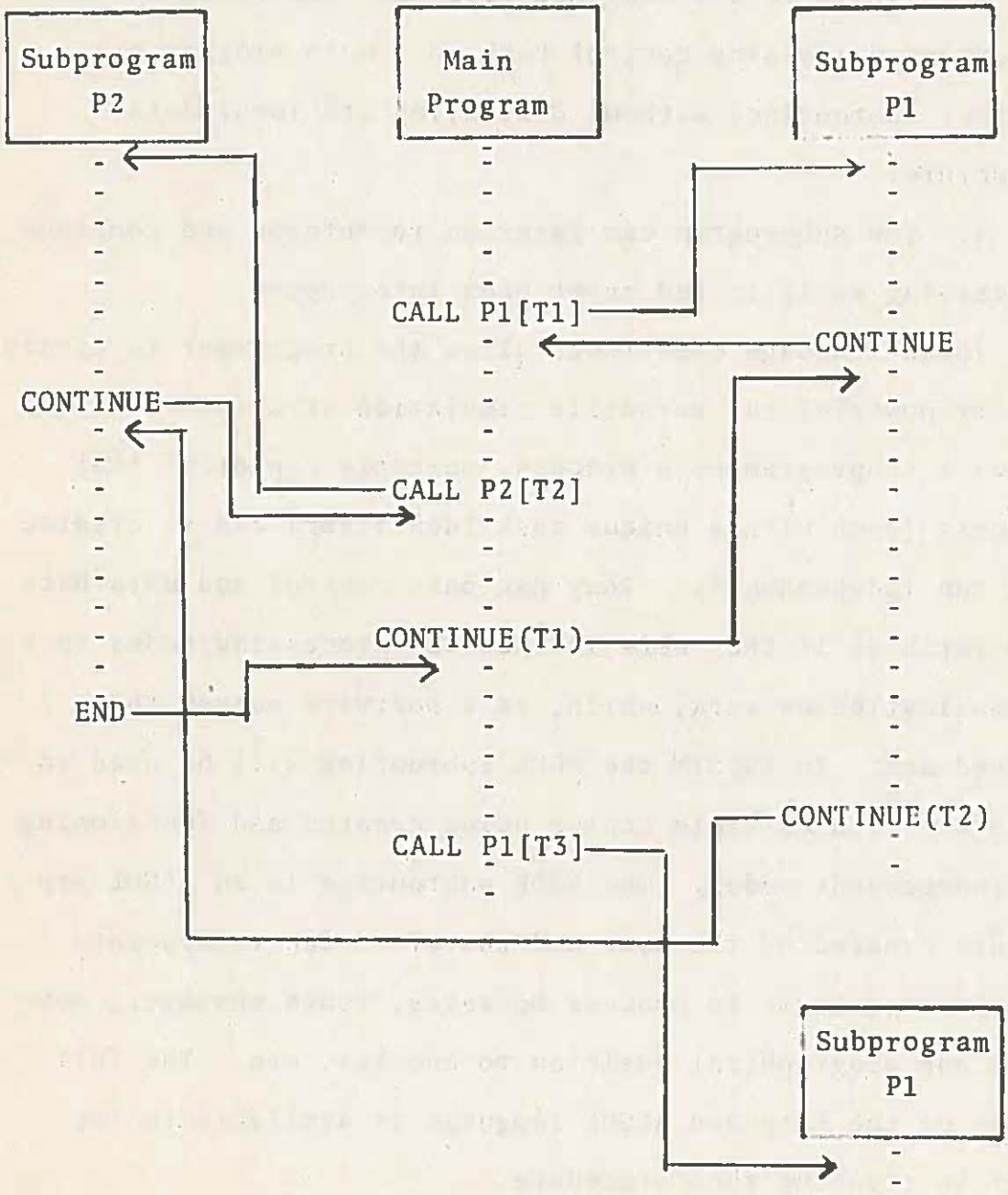

Figure 1. Independent Task Processing

GENERAL SIMULATION CONSTRUCTS

This section contains an explanation of the general simulation constructs available to the model user. These constructs are not necessarily related to communication network simulation and can be used to simulate any dynamic system on the Burroughs B6700 system.

The heart of the model is the "next event clock" through which independent node tasks pass control back and forth. Figure 2 shows how the clock is implemented as a singly linked list.

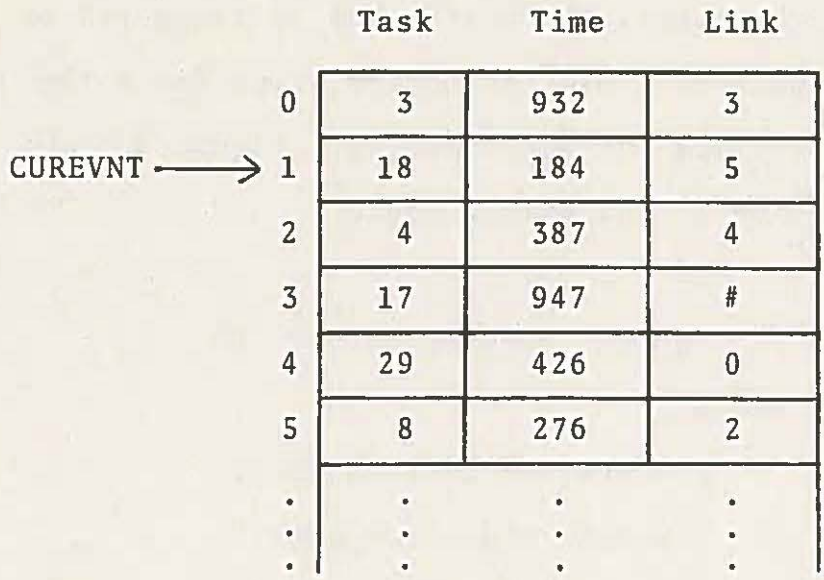

Figure 2. Event Clock

Each entry on the clock is considered an event, i.e. an activation of a particular task (node) at a particular time. The index specifying the currently active event is called

CUREVNT. The time associated with CUREVNT is called SIMTIME and is represented on the clock by an integer between 0 and 2^{29}-1. The clock can therefore represent 2^{29} separate points in time at which an event can take place. By scaling time appropriately, say in seconds, a total simulation period of four hours can be covered at microsecond accuracy. Two events occurring within the same microsecond would appear to be simultaneous to the model. Because of program design, two such events would be executed in the order in which they were placed on the clock.

As an event completes, the next event is triggered by passing control back to a DRIVER routine which finds the next event on the clock and activates it. Figure 3 shows a simplified version of the DRIVER logic.

```
        WHILE  "events exist"  DO
BEGIN
        CUREVNT := LNK[CUREVNT];
        "Delete old clock entry"
        CONTINUE(CURTSK):
END;
```

Figure 3. Driver Subroutine Logic

The re-entry point and local data structures of each independent node task are automatically kept by the B6700

executive system. This executive system feature creates an ideal structure to simulate independent and parallel processes. The following simulation constructs have been defined to aid in implementing this structure.

SCHED verb

SCHED(T,A,PT) is an ALGOL procedure which will create an event on the clock to activate task T at time A. If A=0, it will insert the event immediately after task PT. It is therefore possible to schedule events at a specific time or after a specific event.

RESCHED verb

RESCHED(T,A,PT) is an ALGOL procedure which schedules events in a manner similar to SCHED. RESCHED also deletes from the clock any previously scheduled event for task T. This verb is used to accomplish a "preemptive" scheduling of tasks already scheduled.

PASSIVATE verb

This verb is used to suspend processing in a task and pass control back to DRIVER to activate the next event.

HOLD verb

This verb is used when a task must reschedule itself for some time in the future. HOLD(A) will cause a task to schedule itself A time units in the future and then PASSIVATE.

The preceding constructs manipulate the clock and passage of control between tasks. Tasks are removed from the system along with their local data structures when they terminate via a normal subroutine return. New tasks can be created by use of the CREATE verb.

CREATE verb

CREATE(A)<procedure call>[E[Z]] will cause a "copy" of the procedure named in the <procedure call> to be

1. Assigned a unique task identifier designated by global system parameter Z.
2. Initialized with current parameter values.
3. Scheduled to become active at its first executable statement at time A.

QUEUES

The final simulation construct needed is that of a queue. Queues in this model are implemented as cyclical buffers of user defined size. They may be represented as follows:

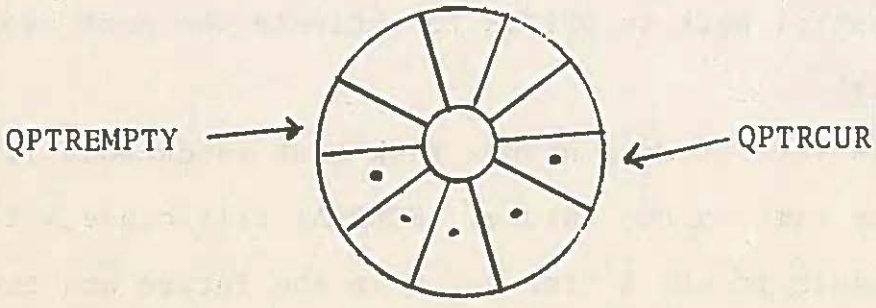

Figure 4. Queue Structure

Two pointers are used to indicate the next index to be used for storing an item (QPTREMPTY) and for extracting the "top" item (QPTRCUR). An additional boolean parameter (QCONTSFLAG) is used to indicate whether or not the queue is empty.

INSERT verb

INSERT(A,B) will insert item B into queue A. INSERT is programmed as a boolean procedure which yields a false value if the queue is full.

EXTRACT verb

EXTRACT(A) will extract the "top" item from queue A. EXTRACT is programmed as an integer procedure and takes on the value of the extracted item. If the queue is empty, it takes on a flag value.

By their design, all queues are first in, first out. Items stored in the queues are integers between 0 and 255 which can be used to identify messages, tasks, or any other numerically indexed entity in the system.

NETWORK SIMULATION CONSTRUCTS

In order to build a generalized network simulation model, specific network constructs were added to the previously defined general simulations structure. These constructs can be grouped under three headings-- messages, nodes, and links.

MESSAGES

Messages are represented by a global 2-dimensional array which can be accessed and modified by any task (node). The array can be initialized by the user if a specific message scenario is desired, or it can be stochastically or deterministically generated during a simulation run. Each row in the array represents one message and contains the following entries:

SOURCE	Node of origin of message
DESTINATION	Node of destination of message
CURNT	Current node
PRIORITY	Message priority
ERLVL	An indicator of error level to be interpreted and used as desired.
MLENGTH	An indicator of message length to be interpreted and used as desired.

Depending upon the user's interpretation of the system, a "message" may be an entire message, a packet, a byte, a character, or whatever is desired. The message originates at the SOURCE node and is modified by nodal subroutine logic as it passes through the network.

NODES

NODES are represented by a logical structure which operates on other NODES and messages, and by a global data structure which contains information which must be available

to other nodes in the network. The global 2-dimensional array NODES contains:

POSITION	LATITUDE
	LONGITUDE
	ALTITUDE
TIME OF POSITION	
TRTIPE	Transmission type
RECTIPE	Receipt type
PROCTIME	Parameter to be used in the calculation of processing delay
(UNUSED) 5 words	To be defined and used as needed

The logic to be employed by a node in accomplishing its task is defined by the user as an ALGOL subroutine (or set of procedures) and is simply placed in the model as the logic of the NODE procedure. Nodal logic might handle error detection, routing, etc. An example of NODE logic is shown in the next section of this paper.

LINKS

Links in this model do not exist in the usual sense in that they are not input at initialization time. Input data consists of nodal and message characteristics as outlined in the previous two sections. Links are examined at each transmission event. For example, given a UHF message broadcast from a satellite to ground stations, airplanes, ships,

and other satellites, the following logic might apply for the satellite NODE:

1. Choose transmission frequency.
2. For each potential receiver:

   IF "receiver frequency" = "transmission frequency"

   AND

   IF "transmitter and receiver are line-of-sight"

   AND

   IF "transmitter and receiver are within range"

   THEN

   "Transfer message from transmitter node to receiver node queue with a delay and noise level dependent upon the distance and transmitter power."

Also included in the model is a global array GROUNDLINES which would allow nodes to bypass calculations such as those above if connected via groundline, cable, microwave, etc.

EXAMPLE OF MODEL USE

NETWORK DESCRIPTION

The network to be simulated is a store and forward network with a redundant multitiered nodal distribution as shown in Figure 5. Nodes can be assumed to be equipped with radio receivers and transmitters. Frequencies are arranged so that each node can receive from any node in a previous row and transmit to all three nodes in the next row. A message is

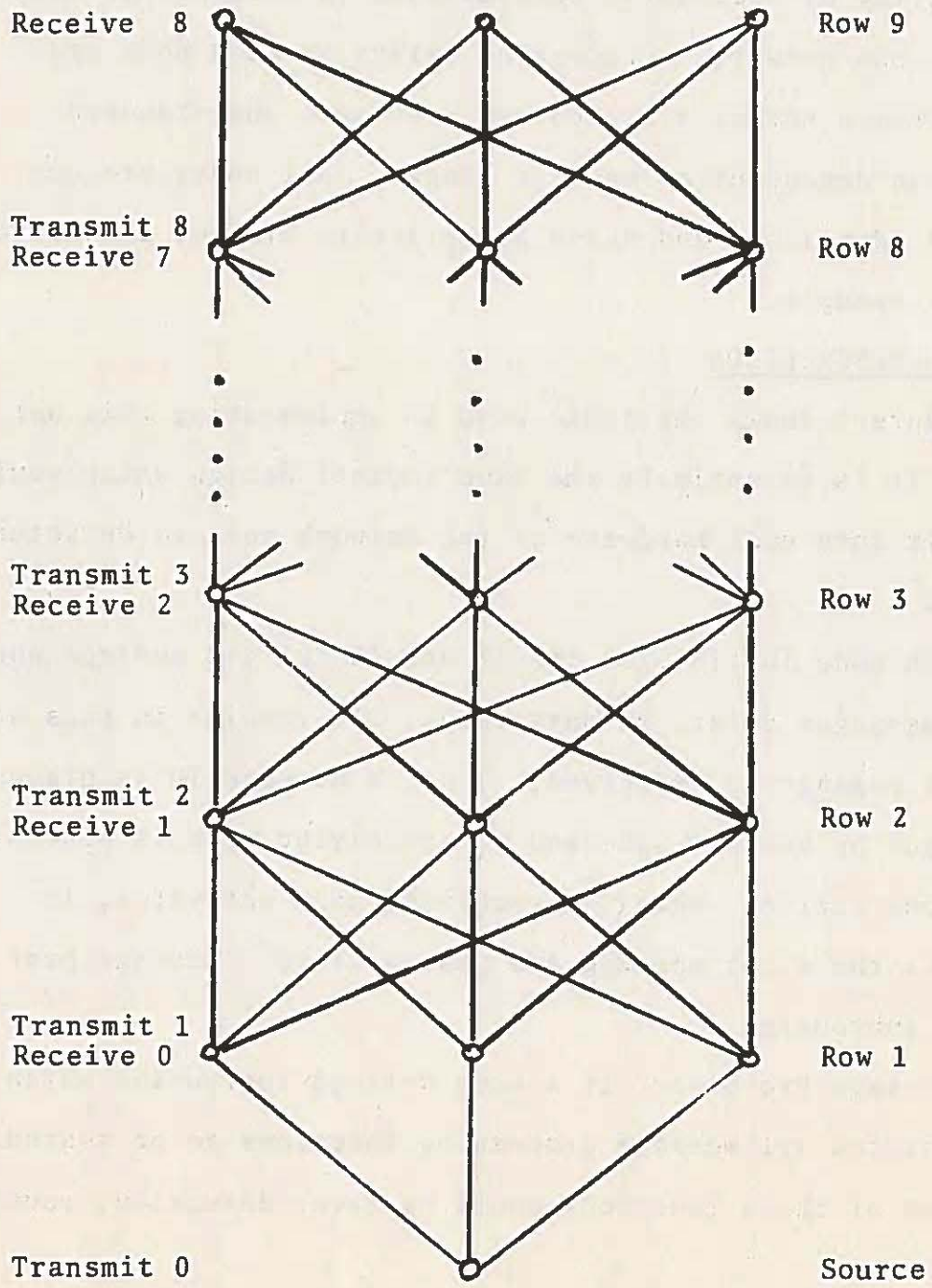

Figure 5. Tiered Network Example

transmitted only once by each node no matter how many times it receives it. (It is possible to receive the same message three times from the three nodes in the previous row.) A probability of successful transmission is attached to each link in the network. Processing delays at each node are drawn from a normal distribution with mean and standard deviation dependent on message length. All nodes are considered identical, and noise and priority are not considered in this example.

PROGRAM DESCRIPTION

Figure 6 shows the logic used in implementing this network. It is essentially the same logical design which would be built into node hardware if the network were to be actually tested.

Each node initializes itself and checks its message queue; if no messages exist, it passivates. It remains in this state until a message is "received," i.e., a message ID is placed in the queue by another node and the receiving node is scheduled to become active. When the receiving node activates, it extracts the first message and passes it to a message processor subroutine.

"Message Processor" is a user defined subroutine which incorporates all message processing functions to be tested. Examples of these functions would be error detection, routing

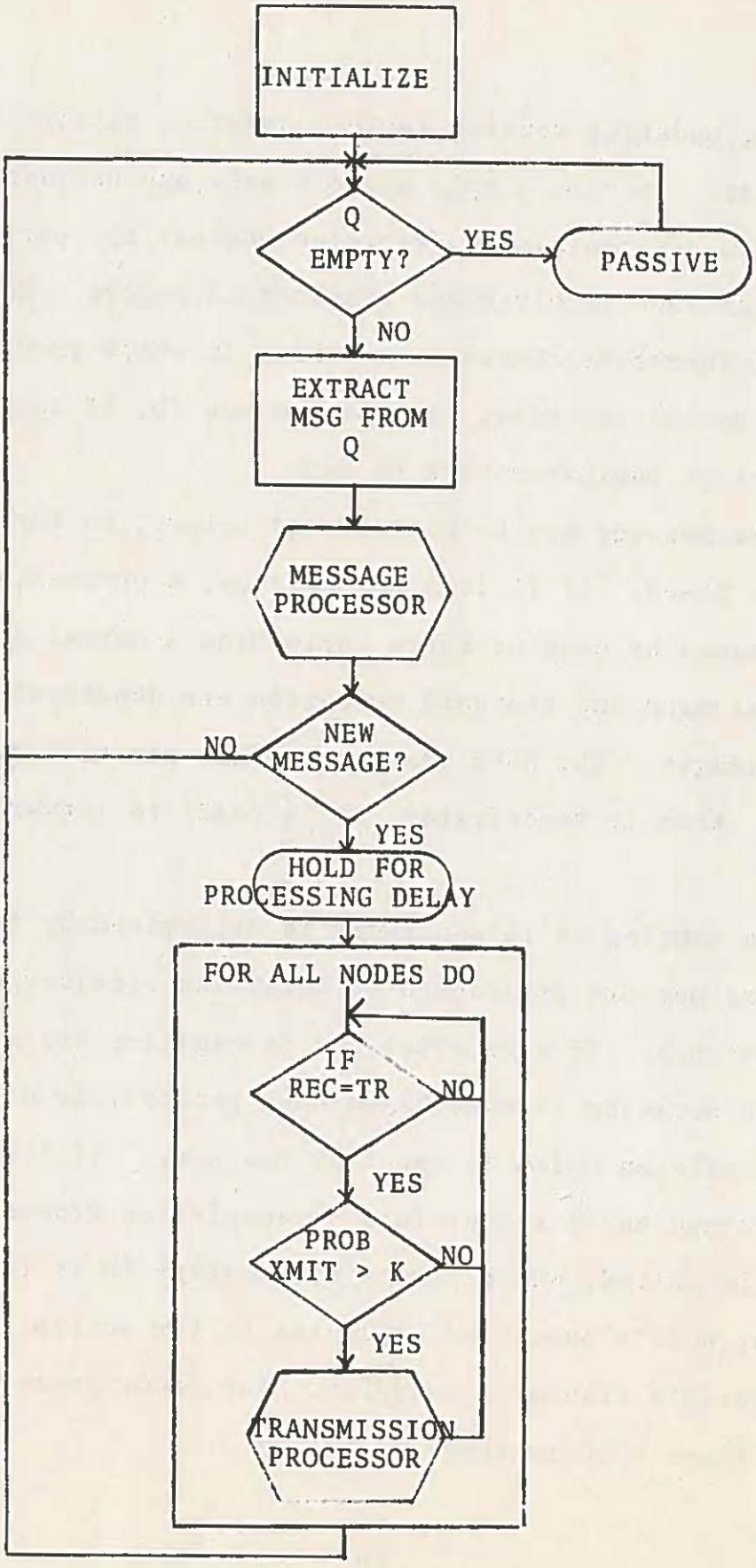

Figure 6. Logic of Node Task

decisions, updating routing tables, updating historical tables, etc. In this simple network only one decision has to be made, and that is to determine whether the particular message has been received and transmitted before. Message Processor therefore checks some tables in which previous message IDs are recorded, records the new ID, if appropriate, and passes an indicator back to NODE.

If the message has been processed before, no further action is taken. If it is a new message, a processing delay is determined by drawing monte carlo from a normal distribution whose mean and standard deviation are dependent upon message length. The NODE task then HOLDs for that amount of time. When it reactivates, it is ready to forward the message.

Since routing in this network is determined by frequency, the entire network is scanned to determine receiver/transmitter matches. If a receiver and transmitter are matched, a boolean decision is made based on a probability of successful transmission which is input by the user. If transmission is determined to be successful, "Transmission Processor" subroutine is called, which places the message ID in the receiving node's queue and schedules it for activation after an appropriate transmission delay. The ALGOL procedure which defines NODE is shown in Figure 7.

```
PROCEDURE NODE(NID);
VALUE NID; INTEGER NID;
          BEGIN
INTEGER   CURRENTMSG,LN,NBRPROC,I;
BOOLEAN   KILLFLAG;
REAL      TM,HT;
LABEL     AWAKE,PASSIVE;
ARRAY     HIST[Ø:5Ø];
                                    % INITIAL ENTRY POINT
               NBRPROC:=-1;         % FOR INITIALIZATION
               SCHEDFLAG[NID]:=1;
               FOR I:= Ø STEP 1 UNTIL 5Ø DO HIST[I]:=-1;
               CONTINUE;            % PASS CONTROL BACK TO
                                    % CREATING PROGRAM AND
                                    % WAIT TO BE ACTIVATED.
AWAKE:         SCHEDFLAG[NID]:=Ø;   %(ACTIVATION POINT)
               IF QCONTSFLAG[NID]=Ø THEN
                                    % IF THE Q IS EMPTY
             BEGIN                  % THEN
PASSIVE:       PASSIVATE;           % PASSIVATE AND WAIT
                                    % TO BE ACTIVATED.
               GO TO AWAKE;         % (ACTIVATION POINT)
             END;                   %  IF Q NOT EMPTY
               CURRENTMSG := EXTRACT(NID);
                                    %  EXTRACT THE TOP MSG
                                    %  AND CALL MSG PROCESSOR.
               MP(CURRENTMSG,NID,HT,KILLFLAG,GHIST,NBRPROC);
                                    %  MSG PROCESSOR WILL
                                    %  PASS FLAG FOR PREVIOUSLY
                                    %  PROCESSED MESSAGE.
               IF KILLFLAG = TRUE   %  IF PREVIOUSLY PROCESSED
               THEN GO TO AWAKE;    %  THEN GET NEXT MSG.
               BUSYFLAG[NID]:=1;    %  IF NEW MSG THEN SET
               HOLD(HT);            %  BUSYFLAG AND DELAY
                                    % FOR A TIME TO SIMULATE
                                    % MESSAGE PROCESSING DELAY
               BUSYFLAG[NID]:=Ø;    %(REACTIVATION POINT)
                                    % FOR ALL NODES
               FOR I:= Ø STEP 1 UNTIL MAXNODES DO
                                    % IF RECEIVER MATCHES XMITTER
               IF TRTIPE[NID] = RECTIPE[I] THEN
               IF DRAW(.7,U) THEN   % AND PROBABILITY OF SUCCESS
                                    % IS TRUE THEN
               TRP(CURRENTMSG,NID,I); % CALL TRANSMISSION PROCESSOR.
               GO TO AWAKE;         % GO GET NEXT MESSAGE.
          END OF NODE PROCEDURE;
```

Figure 7. Code for NODE Task

This network is a constant topology network. To change it to a variable topology network, one need only add two things. One is a position update routine to describe nodal movement with respect to time, and the other is a line-of-sight check before transmission. A boolean procedure called LOS is provided in the model for this purpose. It checks line-of-sight between two nodes and returns a true or false value along with the distance between nodes.

RESULTS

Data values were input to simulate a nine tiered network of 28 nodes (27 plus a source node) and 78 links. Link probabilities of 0.5, 0.6, and 0.7 were tested and compared with analytical results. For each tier in the network, the tables below compare the probability of at least one node in that tier receiving the message. Additionally, the probabilities of exactly zero, one, two, and three nodes in the ninth tier are shown. Analytical and simulator values are in very close agreement.

Run time for the model was approximately 55 to 65 seconds per 100 messages through the network. It is also significant to note that all procedures were developed and debugged in approximately 5 to 8 man hours.

SUMMARY

By this point the reader should sense the wide range of applications of NETSIM. This section will serve as a summary and will point out some modeling caveats. The user must input the following information:

1. For each node:
    a. time of creation
    b. initial latitude
    c. initial longitude
    d. initial altitude
    e. transmitter type (frequency, e.g.)
    f. receiver type (frequency, e.g.)
    g. message process time
    h. maximum queue length for receiver
    i. position update interval (time)
    j. a procedure to update node latitude, longitude, and altitude.
    k. a procedure for message processing which may include error detection, routing update, store, and forward requirements. In addition, the process delay time may be randomly generated from a selected distribution.
    l. a procedure for transmission processing as a function of message identification number and the receiving node number.

Table 1. Message Receipt Probability with 0.5 Link Success Probability

NETSIM

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	1.00				
3	0.97				
4	0.95				
5	0.92				
6	0.91				
7	0.90				
8	0.86				
9	0.84	0.16	0.11	0.26	0.47

ANALYTIC

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	0.99				
3	0.97				
4	0.95				
5	0.93				
6	0.91				
7	0.89				
8	0.87				
9	0.86	0.14	0.10	0.30	0.45

Table 2. Message Receipt Probability with 0.6 Link Success Probability

NETSIM

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	1.00				
3	1.00				
4	0.99				
5	0.98				
6	0.98				
7	0.98				
8	0.97				
9	0.97	0.03	0.03	0.24	0.70

ANALYTIC

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	1.00				
3	1.00				
4	0.99				
5	0.99				
6	0.99				
7	0.98				
8	0.98				
9	0.98	0.02	0.03	0.21	0.74

Table 3. Message Receipt Probability with 0.7 Link Success Probability

NETSIM

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	1.00				
3	1.00				
4	1.00				
5	1.00				
6	1.00				
7	1.00				
8	1.00				
9	1.00	0.00	0.02	0.10	0.85

ANALYTIC

TIER	AT LEAST ONE	EXACTLY			
		ZERO	ONE	TWO	THREE
1	1.00				
2	1.00				
3	1.00				
4	1.00				
5	1.00				
6	1.00				
7	1.00				
8	1.00				
9	1.00	0.00	0.00	0.09	0.90

2. For each message:
    a. choice of random or programmed insertion
    b. for programmed insertion, indicate message source, destination, priority, error level parameter, message length, and time of insertion
    c. for random insertion, indicate probability distribution types for the generation of each or the parameters above
    d. a combination of programmed and random may be used.
3. For each arc:
    a. existence criteria as a function of distance, transmitter and receiver types, and damage function.

In the example presented, messages are not treated at the bit level. Therefore, some assumptions must be understood regarding what can and cannot be modeled. First, it appears that a receiver can receive many messages at the same simulation time. This characteristic accurately models a system where, say, a tier of transmitters simultaneously transmits to a single node and where the transmissions are synchronized so that each is in a small time window and overlaps cannot result. If the desired system is not of this type, it is easy to model the situation where, e.g., simultaneous message receipt blocks all receipt.

Peripheral procedures are drawn from the standard Burroughs packages. These include the probability distributions and the statistical output procedures available in NETSIM.

NETSIM has also been used to model various switching circuit networks of time variant topology which are under consideration for Air Force applications. For security reasons these networks cannot be described in this paper, but it does appear feasible to use the NETSIM model and methodology to evaluate many network characteristics such as nodal logic, self-healing routing schemes, nodal position effects, etc., which are difficult or impossible to evaluate with static network models.

REFERENCES

1. Organick, Elliott I. Computer System Organization, The B5700/B6700 Series, Academic Press, 1973, New York and London.

2. "Burroughs B6700/B7700 ALGOL Language Reference Manual," Document # 5000649, Burroughs Corp, Detroit, Michigan, 1974.

3. "Burroughs Information Processing System SIMULA Reference Manual," Burroughs Corp, Detroit, Michigan, 1970.

4. "On Distributed Communications: Summary Overview," Paul Baran, Rand RM-3767-PR, August, 1964.

5. "On Distributed Communications: Digital Simulation of Hot-Potato Routing in a Broadband Distributed Communications Network," Sharla P. Boehm and Paul Baran, Rand RM-3103-PR, August, 1964.

6. "On Distributed Communications: Introduction to Distributed Communications Networks," Paul Baran, Rand RM-3420-PR, August, 1964.