韓國 軍事運營分析 學會誌
第 11 卷 第 1 號, 1985. 6.

Implementation of a Simulation Model for a Local Area Network Design.

Chung, Koodon*

Abstract

This thesis provides the implementation of a simulation model for a particular Local Area Network (LAN), employs carrier sense multiple access (CSMA) bus architecture, which implements functions of the Stock Point Logistics Integrated Communication Environment (SPLICE). First, specifications of the model are identified based on the given functional specification and operating system design. Then the approach taken for modeling and programming in GPSS is discussed. Finally, the program and results of the simulation run are provided.

I. INTRODUCTION

Distributed systems presently represent the fastest growing area of the data processing world, the university and research environments, and the military community. Such systems consist of a conglomeration of cooperating, individual computer systems, where each system typically consists of a micor-or minicomputer system, software, and various types of peripherals. The never-ending demands for increased and uninterrupted support at lowest possible cost are the factors influencing the trend toward such systems. Another major attraction of building large systems by coupling large nembers of smaller processors is the expectation of a simpler software design. In a network, it is possible to dedicate some (or all) of the processors to specialized functions, eliminating much of the software complexity associated with large mainframes. Contrary to software development costs, mini- and microcomputers are becoming less expensive and are, therefore, being used increasingly for many functions.

* Rok Air Force

Growing complexity and number of computer resources in the logistic function, and increasing demands for interactive operations at the stock points and inventory control points have led to using a LAN at each site for integration of all computer resources. At this point in the project, development of a simulation model is required as an effective analytical tool to assist in comparing various network topologies and protocols (Ref. 2).

This research covers the implementation phase in the development process. In the development of a simulation model, the following steps are taken.

1. For a LAN which employs Carrier Sense Multiple Acess with Collision Detection (CSMA/CD), the specifications of a simulation model are discussed based on the functional design and the operating system design provided.
2. Based on the specifications, the modeling process is described by building GPSS block diagrams.
3. Programming of the model is discussed by converting the block diagram into GPSS V code.
4. A completed program using hypothetical data, and the results of a simulation run is provided.

II. MODEL DESIGN

A. OVERVIEW OF CSMA/CD

The approach used in CSMA/CD traces its roots back to the radio-based Aloha packet switching network. In the Aloha network, terminals equipped with packet radios shared a common multiaccess radio channel. The attraction of this approach is its simplicity and low cost resulting from elimination of any central control. As the load increase, however, the maximum possible utilization of a pure Aloha channel is only 18 percent (Ref. 6). To eliminate such a degradation of channel utilization, CSMA/CD employs the following techniques.

1. Carrier Sense

This scheme forces the station to defer its transmission if any transmission is in progress. There are three CSMA protocols that can be used to sense the status of channel (Ref. 3).

a. 1-persistent CSMA

When a node has data to send, it first listens to the channel to see if another node is transmitting. If the channel is busy, the station waits until the channel becomes idle. When the node detects an idle channel, it transmits a packet immediately.

b. Nonpersistent CSMA

Before sending, a node senses the channel. If no other node is sending, the node begins transmitting. However, if the channel is already in use, the node does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and repeats the algorithm.

c. P-persistent CSMA

It applies to slotted channels and works as follows. When a station becomes ready to send, it

senses the channel. If it is idle, it transmits with probability p. With a probability q = 1−p, it defers until the next slot. If that slot is also idle, it either transmits or defers again, with probabilities p and q. This process is repreated until either the packet has been transmitted or another node has begun transmitting.

With the above schemes, it is still possible that two or more stations will sense that the channel is idle and begin transmission simultaneously, producing a collision.

2. Collision Detection

Each node continues to monitor the channel during transmission and can provide collision detection when the signal on the channel does not match its own output. In this case, each station stops transmitting, uses a collision consensus enforcement procedure to ensure that all other colliding stations have seen the collision, and then stops. A transmission is then rescheduled for some later time.

3. Backoff Schemes

To avoid repeated collision, each node waits for a random period of time. There are many variations to the CSMA/CD scheme [Ref. 11], all of which pertain to the methods of handling data collisions and retransmission of lost packets. Generally speaking, if there are "N" nodes wishing to transmit in a given time slot (defined as the round trip end-to-end bus propagation delay) after a collision, then the most straightforward strategy would be to let each of these nodes have a probability of 1/N for starting a transmission in this time slot. However, in a decentralized environment, it is virtually impossible for each node to correctly determine this number "N".

a. Linear-feedback Algorithm

A linear-feedback algorithm tries to estimate the number of nodes wishing to transmit in a given time slot (round trip end-to-end bus propagation delay) according to the number of collisions encountered in transmitting a packet. For example, if a node attempted to transmit a packet r times without success then during the re-transmission period, it uses 1/(1+r) as its transmission probability in each time slot. Apparently, this estimation is too conservative and thoughput degradation will occur when the system is heavily loaded [Ref. 11].

b. Binary Exponential Backoff Algorithm

To avoid such a throughput degradation, truncated binary exponential backoff algorithm is used. After r-th collision, all colliding stations set a local parameter, k, a packet at a node will defer k time slots, where k is randomly selected between 0 and 2**r (i.e., 0<k<2**r) until 2**r reaches a certain limit. With this scheme, as the system becomes more heavily loaded, the nodes automatically adapt to the load [Ref. 13].

B. MODEL SPECIFICATION

1. Description

Based on the functional specification and the design strategy of the system provided by Reference 1 and Reference 2, this section describes the specification of a simulation model which enables us to evaluate the performance of a particular LAN which employs a bus architecture.

The system is composed of a single contention bus and a set of nodes. The bus provides the communication path between the nodes in which CSMA/CD protocols are used. Each node contains one or more functional modules (FMs) acting as a server and communication facilities. Since we are interested in the performance of a LAN rather than the function of each node, the term "node" is used as a server unit instead of "functional module".

Transactions are created at terminals with a priority level and are sent to the system, them, the transaction visits a set of nodes via inter-node communication. After being served by the nodes in the predetermined sequence, transactions leave the system. When a transction arrives, the system determines the node to be visited by the transaction. Then messages and control signals required are generated and sent to the destination node. The receiving node creates a process and puts in into a processor queue until the process gets control of the processor. Once a process gets the processor, it continues to execute unless interrupted by higher priority process or timer interrupt. A time quantum is not allocated to each process; however, a process is not allowed to execute indefinitely. If a process has executed for the maximum allowable processor time, a system timer interrupts the process. Then the process can go back to the processor queue at its original priority level [Ref. 2]. After completion of a process the system determines whether service for the transaction has been completed. If it has been completed the transaction leaves the system; otherwise the next node to be visted is identified and control signals and messages are generated. The messages are put into the sending queue until they are transmitted. If the bus is sensed idle by the node, messages are broadcast and propagates through the bus. A node will try to complete the transmission of the first message in the sending queue before attempting the send a second message (if any). The receiving node will transmit a positive acknowledgement if the transmission was successful. If the sending node detected a collision or if a positive acknowledgement had not been received over some period time, the transmission will be rescheduled using the backoff algorithm as described in the last chapter. The system repeats the above procedure until the service for the transaction has completed.

After visiting a set of nodes, the transaction leaves the model. Figure I shows simplified transaction flow in the model.

2. Model Assumptions

1. The one-line, one-server queuing system is employed for the processor at each node.
2. Transaction arrivals are random and characterized by the Poisson distribution. This means the interarrival time is exponentially distributed.
3. Since no specifications are provided for "failure" report handling, it is assumed that the transaction leaves the system at the time of "failure" report.
4. Message length is a random variable and exponentially distributed.
5. The system employs either a linear or binary exponential backoff scheme. All the nodes use same scheme.
6. The probability of collision in the system can be estimated by applying the channel efficiency described in Reference 7.

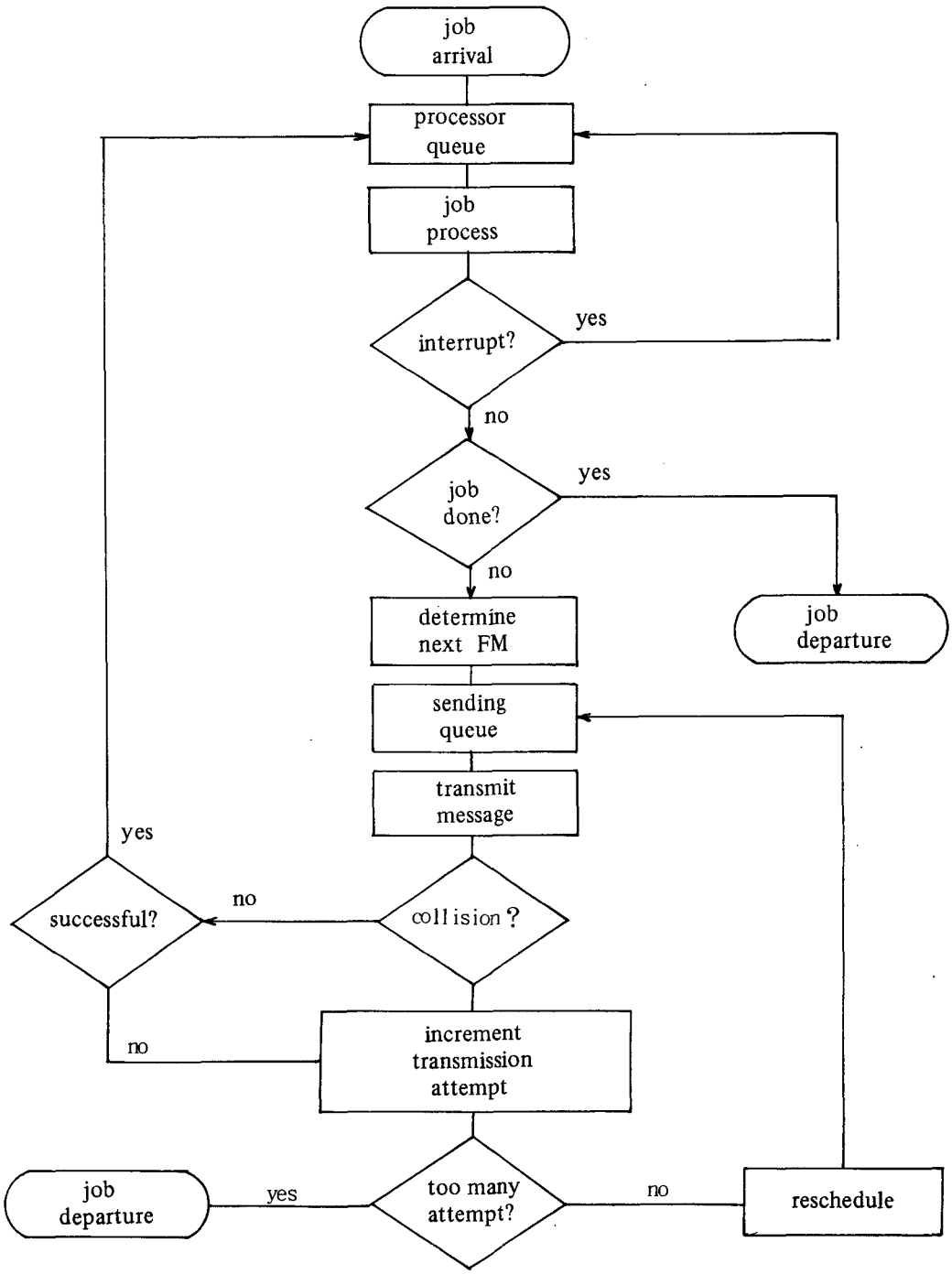

Figure 1. Simplified Transaction Flow.

3. Events in the Model

In our simulation model, discontinuous changes of states occur. The term "event" is used to mean such a change occurring at a specific point in time. As a transaction goes through the model, there are two different kinds of events which may take place independently: processorevents and channel-events.

a. Processor-event

PEVENT 1 (start processing): if a process captures a processor, the processor will start the process immediately.

PEVENT 2 (interrupt): if a process is interrupted by a higher priority process or the timer in the system, that process goes back to the processor queue.

PEVENT 3 (end processing): on the completion of execution, a process releases the processor.

b. Channel-event

CEVENT 1 (transmission): if the channel is sensed idele by a node, then a message is transmitted onto the channel.

CEVENT 2 (defer): if the channel is sensed busy, then the node must defer its transmission attempt until the end of the transmission.

CEVENT 3 (collision): if two or more nodes start their transmissions before they can detect each other's signal, then a collision will occur, and the backoff algorithm will be invoked.

CEVENT 4 (dead packet): after the termination of a message transmission, whether successful or not, the packet will still be traveling toward the two ends of the bus. In the model, we shall refer to this as a "dead packet" although its presence could still affect other nodes transmission decisions.

4. Input Parameters and Output Statistics

a. Input Parameters

Since the physical configuration of the system is not fully specified, we need to develop a flexible model to reflect various physical configurations by using input parameters. The following is a list of input parameters to be considered.

1. Number of nodes in the system.
2. Transaction arrival rate and distribution.
3. Sequence of nodes to be visited for each transaction class.
4. Mean service time of each node per transaction class.
5. Priority level of each transaction class.
6. Packet size and distribution.
7. Channel capacity.
8. Maximum processor time per process.
9. Length of the bus (slot time).

b. Output Statistics

The object of this study is to develop a simulation model which enables us to evaluate a LAN in support of SPLICE. Consequently, the output of the model includes the following statistics:

1. Response time for whole transactions and for each transaction class.

2. Queuing statistics at each node and of the system.
3. Collision at each node and of the system.
4. Channel throughput (utilization)
5. Percent of channel access at each node.

III. IMPLEMENTATION OF THE MODEL

A. SIMULATION LANGUAGE

As described earlier, the movement of transactions in the model can be expressed in terms of an event. This enables us to use a discrete event-driven model to simulate the system. The GPSS simulation language is a popular simulation tool for handling event-driven models and the concurrency of transactions in a model. The GPSS does not provide enough arithmetic functions to calculate channel-efficiency and backoff algorithm, however the HELP blocks enable the GPSS user to incorporate independently written routines into his simulation run [Ref. 5]. FORTRAN routines are used to perform these functions.

B. TIME UNIT OF SIMULATION CLOCK

The passage of time is recorded as a number called clock time, or simply the clock. The clock is initially set to zero and, as the simulation proceeds, it is updated to reflect the passage of time. The unit of time is an integral number which can be used for any time interval chosen by the user. In practice, the unit of time must be small enough to realistically reflect the time spans which occur in the system being modeled. Since this model reflects a computer system, a 10 microsecond unit is appropriate for the time unit.

It is possible to update the clock in uniform steps. In that case, the program must determine if an event is due to occur at the new time [Ref. 4]. In this model, however, the program keeps a record of future events in chronological order, and the clock time is updated to the time of the next most imminent event.

C. SETUP AND USE OF MATRICES

Matrices are used to determine the visitation sequence of transactions, and the mean service times of nodes for each transaction class. Table 1 shows an example of visitation sequences for 12 transaction classes and mean service time of nodes for each transaction class. Using the data provided in Table 1, the following two tables are produced. Table 2 and Table 3 can be used by the model as a visitation sequence matrix and a mean service time matrix, respectively. Before being served by a node, a transaction simply needs to index the appropriate cell of these matrices. There are eight columns in each matrix, corresponding to the maximum number of nodes that any one transaction class must visit. Twelve rows in the matrix represent the transaction classes 1 through 12. Similar to the transaction classes, a system generated process can be included by adding the desired number of rows. Enteries in the body of Table 2 and Table 3 are interpreted

Table 1. Visitation Sequence and Mean Service Times for Transaction Classes (Hypothetical Data)

Trans. Class	Total no. of nodes to be visited	Visitation Sequence (node number) /Mean Service time (millisecond)			
1	3	Node 5/40	Node 6/45	Node 4/50	
2	3	Node 8/55	Node 6/60	Node 4/75	
3	2	Node 6/80	Node 4/92		
4	3	Node 6/66	Node 4/75	Node 2/67	
5	4	Node 5/65	Node 6/54	Node 1/66	Node 3/80
6	2	Node 4/64	Node 1/85		
7	3	Node 7/64	Node 6/58	Node 4/100	
8	3	Node 3/34	Node 6/58	Node 4/55	
9	4	Node 5/45	Node 3/74	Node 6/63	Node 4/87
10	3	Node 3/85	Node 4/88	Node 2/90	
11	3	Node 2/35	Node 6/60	Node 4/100	
12	4	Node 2/55	Node 5/65	Node 6/83	Node 4/85

Table 2. Visitation Sequence Matrix (Hypothetical Data)

	No. of nodes yet to be visited							
Trans. Class	1	2	3	4	5	6	7	8
1	5	6	4					
2	8	6	4					
3	6	4						
4	6	4	2					
5	5	6	1	3				
6	4	1						
7	7	6	4					
8	3	6	4					
9	5	3	6	4				
10	3	4	2					
11	2	6	4					
12	2	5	6	4				

Note: Each cell represents the node number to be visited next.

Table 3. Mean Service Time Matrix (Hypothetical Data)

	No. of nodes yet to be visited							
Trans. Class	1	2	3	4	5	6	7	8
Trans. Cl								
1	400	450	500					
2	550	600	750					
3	800	920						
4	880	750	670					
5	660	540	660	800				
6	650	850						
7	640	580	1000					
8	430	580	550					
9	450	740	630	870				
10	850	880	900					
11	350	600	1000					
12	550	650	830	850				

Note: Each cell represents the mean service time in 10 microsecond units.

as the identifier of a node to be visited next and the mean service time of the next node for a transaction, respectively.

It is a simple matter for a transaction to index the proper cell of the visitation sequence and mean service time matrices. Byte parameter 1 (PB1) is assigned to identify the transaction class and PB1 can be used as a matrix rowindex. Byte parameter 2 (PB2) contains the number of remaining nodes to be visited for the transaction and PB2 can be used as a matrix column-index. To do this. PB2 needs to be initialized with the total number of nodes to be visited. After a transaction has been served by a node, PB2 will be decremented by 1, meaning that its value can be interpreted as the number of nodes yet to be visited. Consequently, the model can recognize the completion of service for a transaction when PB2 has value zero. Then "MB1 (P1, P2)" indirectly specifies the next node number and "MX2 (P1, P2)" indirectly specifies the corresponding mean service time.

All the parameters, savevalues, matrix savevalues, functions, and variables used by the model are defined in Table 4.

Table 4. SNA Definitions

GPSS ENTITY	SNA	DESCRIPTION
PARAMETERS		
Byteword	PB1	Carries Transactions Class
	PB2	No. of nodes yet to visit
	PB3	Next node to be visited
	PB4	Transmission attempt without success
Fullword	PF1	Service time required by current node
	PF2	Packet length to be transmitted
SAVEVALUES		
Byteword	XB$NUM	Total No. of nodes
	XB$MTRY	Max. number of transmission attempts allowed per transaction
	XB$COUNT	Mutiplication count
Halfword	XHj	Number of collisions at node j
	XH$FAILj	Consecutive transmission failures at each node
Fullword	FX$PACK	Mean packet length
	XF$CAPA	Channel capacity
	XF$MTIME	Max. processor time per transaction
	XF$COLj	Total no. of collisions at node j
	XF$SUCj	Total no. of successful transmissions at node j
Float-point	XL$SLOT	Slot time (two way propagation delay)
	XL$PACQU	Channel acquisition probability for one node caculated by equation: $(1-1/Q)^{**}(Q-1)$, where Q is total number of nodes
	XL$TWAIT	Mean number of slots waiting in a contention interval before a successful acquisition of the bus by a node
MATRICES		
Byteword	MB1	Visitation sequence matrix
Fullword	MF2	Mean service time matrix
PUNCTIONS	FN$EXPO	Inverted exponential distribution function
	FN$TRANS	Determines Transaction Classes
	FN$NCNDS	Assigns total no. of node to be visited
	FN$PRTY	Assigns priority level
	FN$NEXT	Identities the next node to be visited
	FN$MXINT	Returns the value 2**PB4

Table 4. (Continued)

GPSS ENTITY	SNA	DESCRIPTION
VARIABLES		
	V$WAIT	Mean waiting time for acquiring the channel
	V$MULT	Computes (1-1/Q)*(1-1/Q)
	V$RAND	Generates random no. between 0 and 0.999
	V$EFFI	Calculates channel efficiency
	V$LENTH	Determines packet lengths
	V$STIME	Determines service time
	V$RESCH	Computes reschedule time for colliding packets
	V$XTIME	Computes bus transit time

D. MODEL DESCRIPTION

The simulation model was developed using the GPSS V block diagram. This model consists of two model segments, initialization and simulation of the system.

1. Model Segment 1 (Initialization)

This part initializes all the input parameters identified in the last chapter according to the configuration of a particular LAN to be simulated.

The block diagram of the model segment 1 is depicted in Figure 2. The GENERATE block in the diagram creates only one transaction because segment 1 needs to be executed once. The visitation sequence matrix (MB1) and mean service time matrix (MX2) are initialized by the MSAVEVALUE blocks. The HELP block invokes a FORTRAN subroutine to calculate XL$TWAIT, the mean number of slots involved in waiting in a contention interval before a successful acquisition of the bus by a node. This will be used to compute channel efficiency in the model segment 2.

All the savevalues that need to be initialized have their value established by INITIAL control statements, not by blocks, therefore it does not appear in the block diagram.

XB$NUM:	Total number of nodes
XB$MTRY:	Max. number of transmission trial per message
XH$MINPK:	Min. packet length
XF$PACK:	Mean packet length
XF$CAPA:	Channel capacity
XF$MTIME:	Max. processor time per process
XL$SLOT:	Slot time

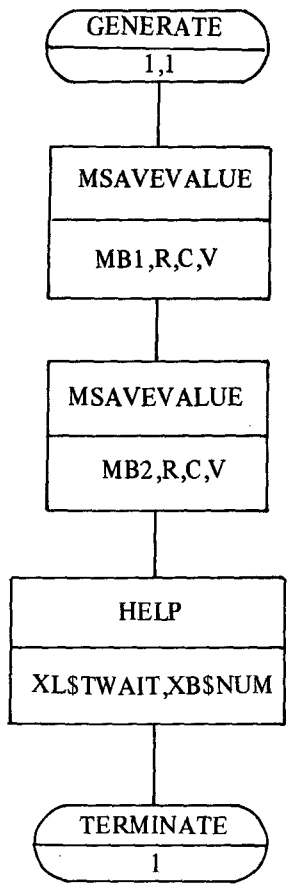

Figure 2. Model Initialization.

Note; Each MSAVEVALUE block represents RXC MSAVEVALUE blocks where R and C are the number of rows and columns of matrix and V is the value to be initialized.

2. Model Segment 2 (Simulation of a LAN)

This segment consists of four distinct functional sections which are: distribution of transactions to the nodes, execution of processes, sending messages, and transiting the bus. Each of these sections will be described prior to discussion of the complete system simulation.

a. Transaction Distribution

Figure 3 shows this portion of the model. The transactions are generated in accordance with the inverted exponential distribution (FN$EXPON) with mean value m.The transaction class for the arriving transaction and the total number of nodes which must be visited by the transaction are determined by FN$TRANS and FN$NONDS, and those values are assigned to PB1 and PB2 by the two ASSIGN blocks. The priority level of arriving transactions are set in the PRIORITY block by the user defined function (FN$PRTY). Then the model identifies the next node to be visited and the mean service time of the node for the transaction. In the next two ASSIGN blocks, the

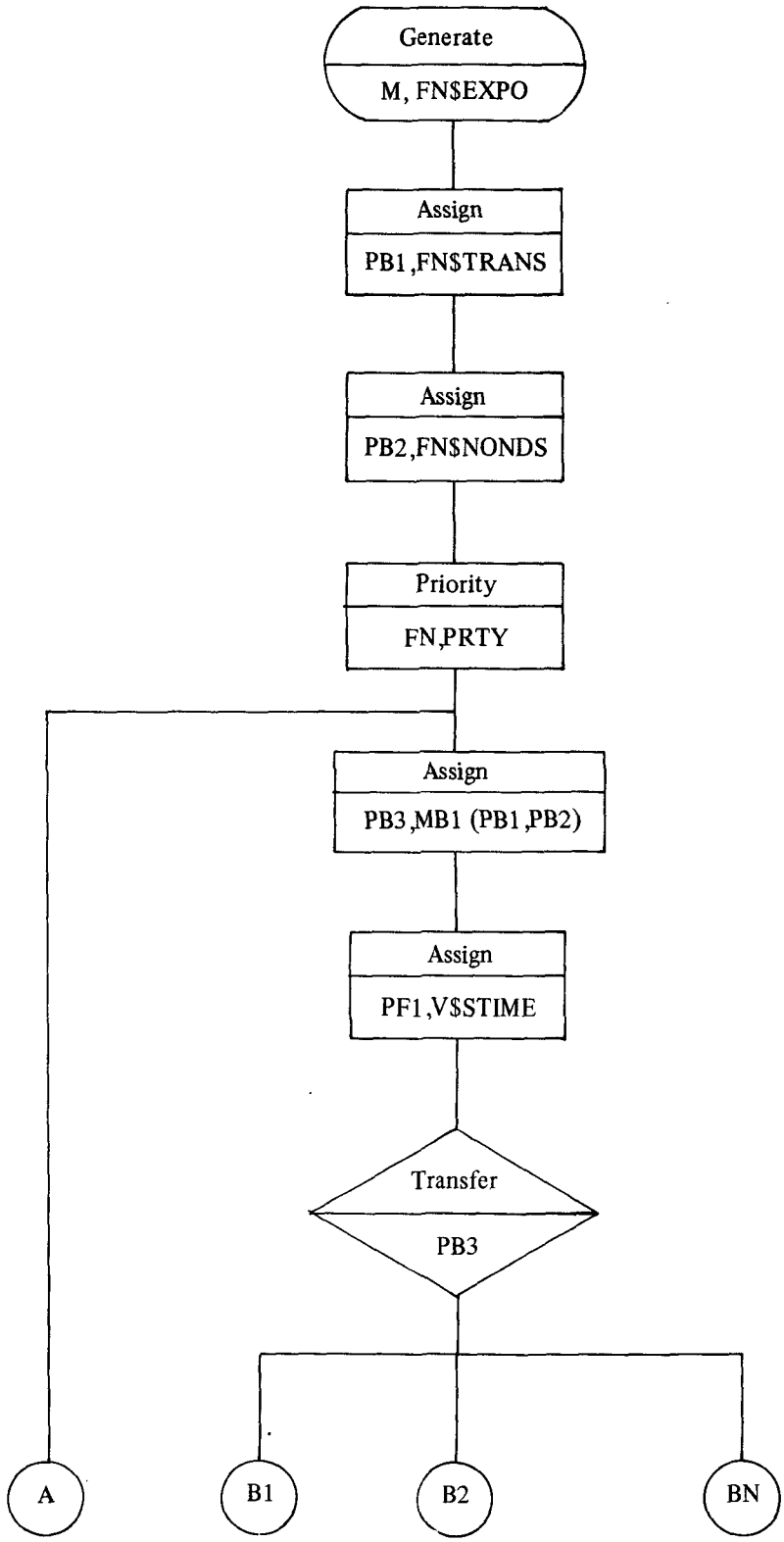

Figure 3. Transaction Distribution.

value of MB1 (PB1, PB2) is assigned to PB3, and PF1 contains the service time of the next node for the transaction. The service time is evaluated by the function, FN$EXPO, with mean service time obtained from the matrix savevalue MX2 (PB1, PB2). After being assigned these parameters, a transaction visits the next node identified by PB3.

The memory requirement is simply not considered because the mean service time and distribution reflects not only CPU time but also I/O time. The size of main memory affects the distribution of service time but not the flow of transactions. The longer service time jobs involve more I/O activity.

b. Execution of Processes

Figure 4 shows this portion of the model for the node-j. When a node receives a message it creates a process and puts the process into processor queue (PROQj) at its original priority level until the process gets the processor.

The PREEMPT block allows only one transaction to be executed at a time, and enables high priority processes. Interrupting the execution of low priority process. High priority interrupts are automatically handled by the PREEMPT block.

There is another type of interrupt we have to consider, the timer-interrupt. If the service time (PF1) is less than or equal to the maximum processor time (XF$MTIME), the process completes its execution without timer interrupt. If the PF1 is greater than the XF$MTIME, however, the process will be interrupted by the timer. In this case the process executes for the maximum allowed processor time and the maximum allowed processor time is subtracted from the service time. The interrupted process, either by high priority process or timer, goes back to the $PROQ_j$ at its own priority level for the remaining service.

When the process has completed its execution, it releases the processor. Then it is necessary to determine whether the service for the transaction has been completed. It is accomplished by decrementing the number of nodes yet to be visited (PB2). If it has a value of zero, the service has been completed. Then the transaction leaves the model. Otherwise, messages and control signals to be sent to the next node are generated.

c. Sending Messages

Figure 5 shows the portion of the model for node-j transmitting messages. If a transaction needs to visit another node, a generated message is put into the global bus interface queue (GBIQj) and waits until the node gets control of the bus (BGUS).

Colliding packets are identified by applying the channel efficiency, that fraction of time the bus is carrying good packets (without collision). We present a set of formulas with which to characterize the performance of a contention bus when it is heavily loaded [Ref. 7] The equations are as follows:

1. $A = (1-(1/Q))^{**}(Q-1)$

   where;

   A is the probability that exactly one station attempts a transmission in a slot, and gets the control of the channel. Q represents the number of stations continuously queued to transmit a packet.

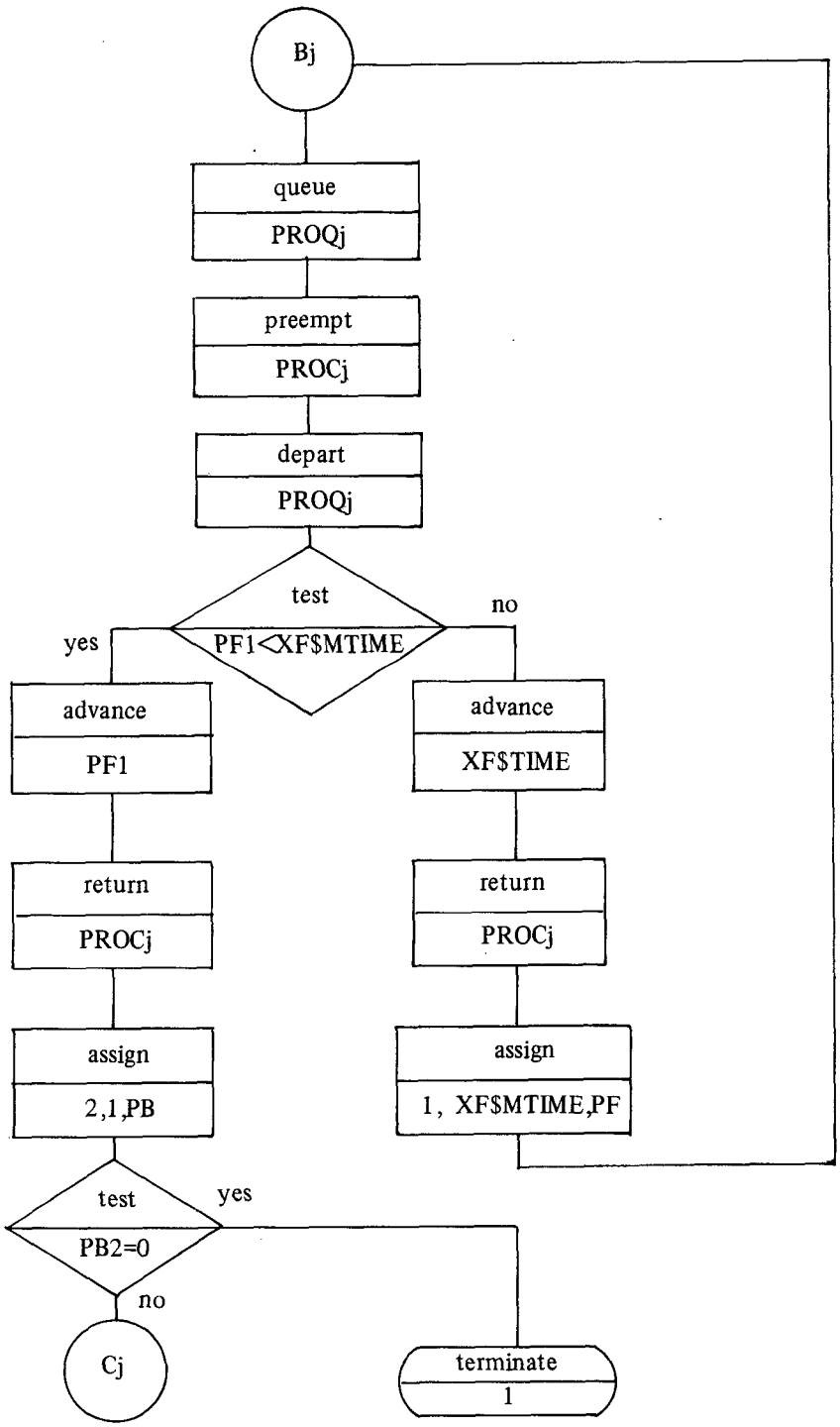

Figure 4. Execution of Processes.

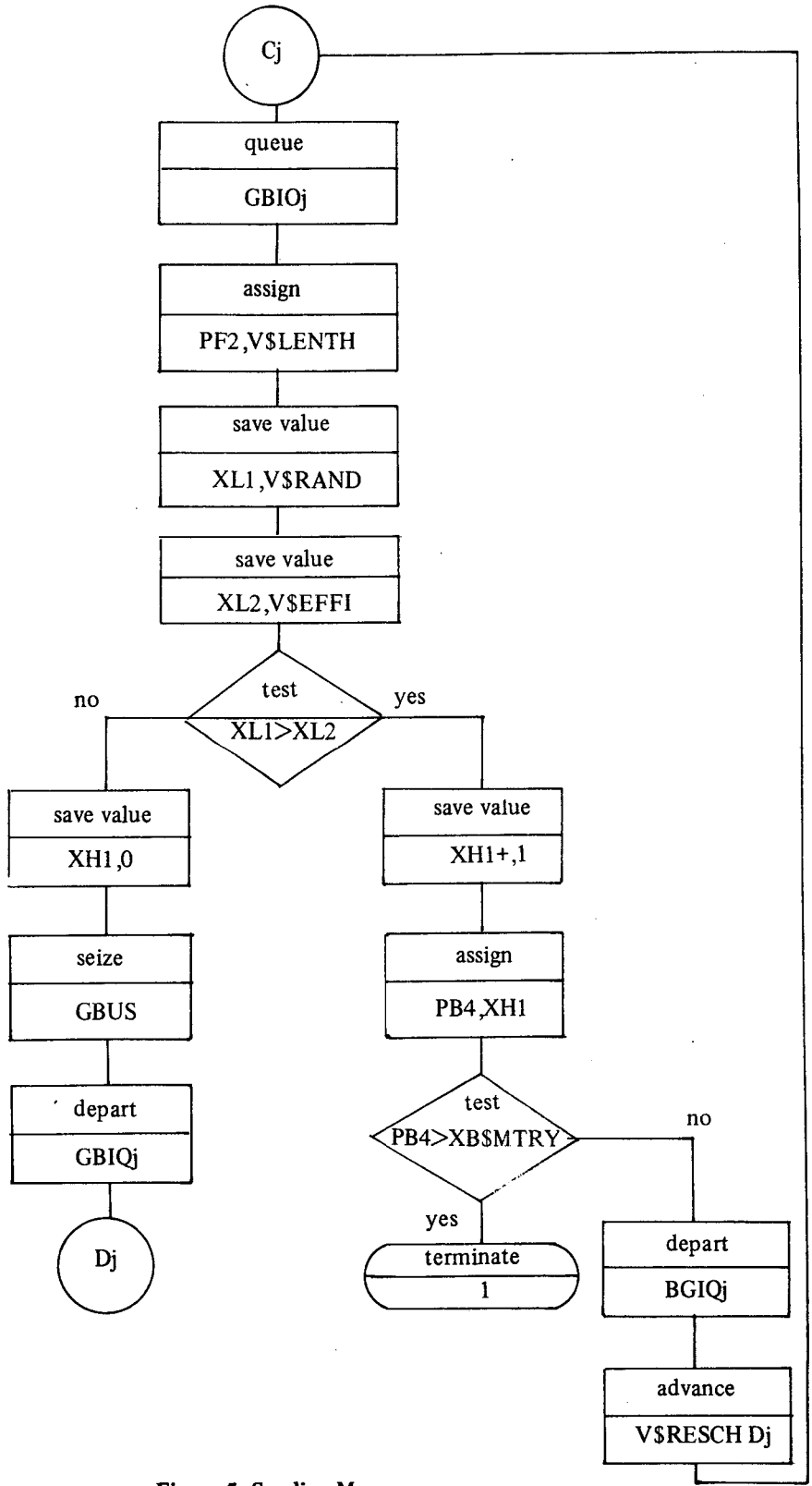

Figure 5. Sending Messages.

2. $W = (1-A)/A$

where:

W is the mean number of slots waiting in a contention interval before a successful acquisition of the bus by a station.

3. $E = (P/C) / (P/C) + (W*T)$

where:

E, P, C, and T represent the channel efficiency, packet length in bits per packet, the peak channel capacity in bits per second, and a time slot in seconds, respectively.

In our model, E is used to estimate the probability of transmitting a packet without collision at each node. When a transaction enters the TEST block, a random number (0.1 - 0.999) is generated. If the random number is greater than E, it is assumed that collision has taken place. Otherwise, the transmission is considered to be successful.

Even though a packet is transmitted without collision, there is the possibility of unseccessful transmission caused by packet errors or receiving node failure. The probability of packet error in the Ethernet, by experiment, is very small (about 1 packet per 2,000,000 packets) [Ref. 8]. The effect of error packets is small enough to be ignored in this process. Unsuccessful transmissions which are caused by system failure can be simulated using FUNAVAIL AND FAVAILABLE blocks. However, no statistics are available for those blocks at this moment.

The packet leaves the GBIQj if the transmission was successful. The colliding packet uses the bus for a unit of time slot and increments the number of transmission attempts for the transaction. If the number of transmission attempts becomes equal to the user defined number (XB$MTRY), a system failure is assumed and the transaction leaves the system. Otherwise, the transmission time is rescheduled and the transaction returns to the GBIQj and waits until the event time has been reached. The HELP block invokes a FORTRAN subroutine to calculate 2**r, where r is number of transmission trials without success for the message, that is used to compute the backoff time.

d. Transiting the Bus

Figure 6 illustrates this portion of the model. Once a packet gets the control of the GBUS, it is transmitted without interruption. When the transmission is accomplished, the transaction releases the GBUS making the GBUS free and visits the next node. Since the PB1 was decremented by one and its value is not zero, the next node can be identified by the same procedure explained in the transaction distribution.

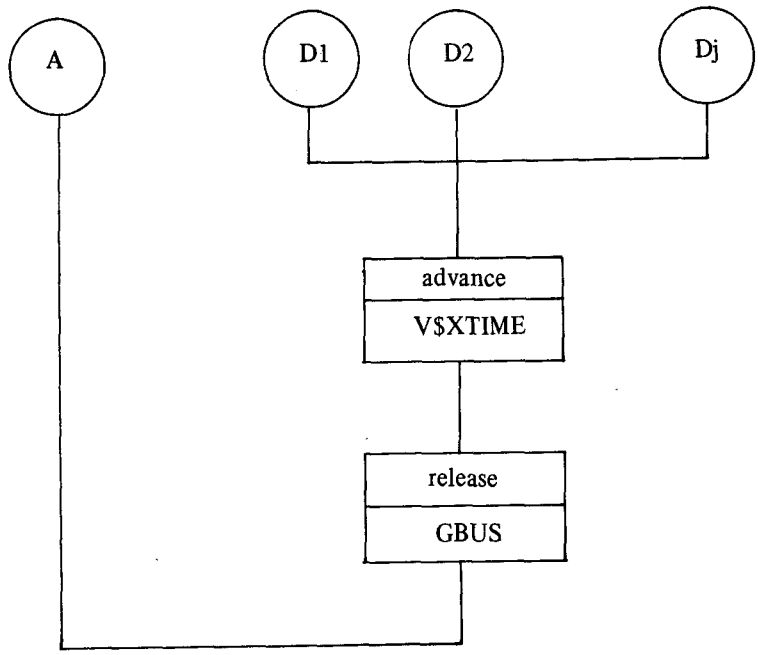

Figure 6. Transiting the Bus.

IV. PROGRAMMING THE MODEL

It is straightforward to write a GPSS program using the block diagram we have constructed. Although two HELP blocks are used in the block diagram, HELP blocks and FORTRAN routines are not used. This program uses a block of GPSS V code using loop and a function (FN$MXINT) to calculate the channel efficiency (XL$EFFI) and backoff time because no information is available to link two programs written in different programming languages.

This chapter describes the transition from a block diagram provided in the last chapter to GPSS V codes. To clarify the transaction flow, the statements used to collect statistics and to generate outputs are not discussed in this chapter.

A. MODEL SEGMENT 1

Because the physical configuration of the system is not fully specified, the following hypothetical data were used to initialize the model.

1. Visitation sequence matrix: Table 2
2. Mean service timematrix: Table 3
3. Total number of nodes: 8
4. Max. number of transmission trials per transaction: 3

5. Min. packet length: 800 bits
6. Mean packet length: 8000 bits
7. Channel capacity: 50 megabits per second
8. Max processor time per process: 10000 (10 microseconds)
9. Slot time: 1 (10 microsecond)

Coding of the block diagram in Figure 2 is described here. Matrix initialization is accomplished by R*C MSAVEVALUE statements:

MSAVEVALUE N,R,C, VLUE, TYPE

where: N is matrix number, R and C are the total number of row and columns in the matrix, V is the value to be assigned, and T represents the type of savevalues.

Other savevalues are initialized by the following INITIAL control statement.

```
INITIAL   XB$NUM, 8/ XB$MTRY, 3/SH$MINPK, 800
INITIAL   XF$PACK, 8000/XF$CAPA, 50000000/XF$MTIME, 10000
INITIAL   XL$SLOT, 0.00001
```

The program below calculates the mean number of slots of waiting in a contention interval, W:

```
        SAVEVALUE PACQU,1,XL              A = 1
        SAVEVALUE COUNT, XB$NUM,XB        COUNT = Q
LOOP    SAVEVALUE PACQU,V$MULT,XL         A = A*(1-1/Q)
        SAVEVALUE COUNT-, 1, XB           COUNT = COUNT-1
        TEST LE   XB$COUNT.1.LOOP         IF COUNT > 1 go to LOOP
        SAVEVALUE TWAIT,V$WAIT,XL         W = (1-A)/A
*       Returns to value of A*(1-1/Q)
MULT    FVARIABLE XL$PACQU*(1-1/XB$NUM)
*       Calculates  (1-A)/A
WAIT    FVARIABLE (1-XL$PACQU)/XL$PACQU
```

The above is equivalent to the following HELP block and FORTRAN routine.

```
        HELPC     #CHANEFF,XL$TWAIT,XB$NUM,XL$PACQU
        SUBROUTINE CHANEFF (W,Q,A)
                   REAL W,Q,A
                   A = (1-1/Q)**(Q-1)
                   W = (1-A)/A
                   RETURN
                   END
```

Segment 1 terminates after initializing all the input parameters.

B. MODEL SEGMENT 2

Table 5 shows the "Workload Forecast Table" provided by the project sponsor. This was used to determine the mean interarrival time and the distribution of transaction classes.

Table 5. Peak Hour Workload Characteristics (First Six Month Period in Year Five)

Trans. Class	Trans/sec	Frequency	Cumulative Frequency
1	.6034	.0396	.0396
2	.8543	.0561	.0957
3	.4053	.0266	.1223
4	.2081	.0184	.1407
5	2.2000	.1444	.2851
6	.3229	.0212	.3063
7	.7262	.0476	.3539
8	.6935	.0455	.3994
9	.0538	.0035	.4029
10	.4397	.0288	.4317
11	.7523	.0494	.4811
12	7.9074	.5189	1.0000
TOTAL	15.6871	1.0000	

Mean Interarrival Time = 1/Total Transaction Arrival Rate
= 1/15.6871 (second)
= 0.06562 (second)
= 6562 (10 microsecond)

The following program corresponds to the transaction distribution part of the block diagram shown in Figure 3.

```
* - INVERTED EXPONENTIAL FUNCTION
*   Determines interarrival time and packet length

EXPO FUNCTION FN1, C24
0,1/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69
0.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12
0.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5
0.98.3.9/.99,4.6/.995,6.2/.999,7/.9997,8

* - TRANSACTION CLASS DISTRIBUTION
*   Determines transaction class based on the Table II

TRANS FUNCTION RN2, D12
0.0396,1/.0957,2/.1223,3/.1407,4/.2851,5/.3063,6
03539,7/.3994,8/.4029,9/.4317,10/.4811,11/1.0,12
```

```
* – TOTAL NO. OF NODES TO VISIT
*    is determined by Table I for each transaction class

NONDS FUNCTION  PB1, L12
1,3/2,3/3,2/4,3/5,4/6,2/7,3/8,3/9,4/10,3/11,3/12,4

* – PRIORITY ASSIGNMENT
*    The priority level of transactions are assigned as follows
*         transaction class  1 - 3:  priority 1
*         transaction class  4 - 6:  priority 2
*         transaction class  7 - 9:  priority 3
*         transaction class 10 - 12: priority 4

PRITY FUNCTION  PB1, L12
1,1/2,1/3,1/4,2/5,2/6,2/7,3/8,3/9,3/10,4/11,4/12,4
          GENERATE    1,FN$TRANS,PB      transaction class
          ASSIGN      2,FN$NONDS,PB      no. of nodes to visit
          PRIORITY    FN$PRTY            assign priority level
  DISTR   ASSIGN      3,MBk(PBk,PB2),PB  next node number
          ASSIGN      1,V$STIME,PF       service time
          TRANSFER    FN,NEXT            go to next node
*         This statement identifies service time to referring to
*         the service time matrix (MX2).
STIME FVARIABLE  MX2(PB1,PB2)*FN$EXPO
```

Since all nodes perform the same functions, using different parameters, macros are used in order to reduce duplication of codes. Two macros are used, NODE and SEND, which corresponds to the execution of processes (Fig. 4) and sending messages (Fig. 5), respectively. These macros are coded as follows:

```
* – NEXT NODE TO VISIT
*    This is identified by referencing the visitation sequence matrix. The contents of the
*    matrix are interpreted by this function.

NEXT FUNCTION PB3,L8
1,NODE1/2,NODE2/3,NODE3/4,NODE4/5,NODE5/6,NODE6/7,NODE7/8,NODE8

* – MAXIMUM INTEGER FUNCTION
*    returns the value of 2**PB4 which is interpreted as the maximum no. of slots to wait
*    for retransmit after collision.
```

```
NODE    STARTMACRO  Parameters
NODEj   QUEUE       PROQj                  enter processor queue
        PREEMPT     PROQj,PR               interrupt by priority
        DEPART      PROQj                  leave processor queue
        TEST LE     PF1,XF$MTIME,TINTj     timer interrupt?
        ADVANCE     PF1                    then go to TINTj
        RETURN      PROCj                  else execute process
        ASSIGN      2-,1,PB                releases the processor
        TEST E      PB2,0,SENDj            PB2 = PB2-1
        TABULATE    TRTM                   job done?
        TRANSFER    FN,TABLE               transit time table
TINTj   ADVANCE     XF$MTIME               to to TABLE
        RETURN      PROCj                  process for max time
        ASSIGN      1-,XF$MTIME,PF         releases the processor
        TRANSFER    PROQj                  service time-max time
        ENDMACRO                           go back to the PROQj
SEND    STARTMACRO  Parameters
SENDj   QUEUE       GBIQj                  enter GBIQj
        ASSIGN      2,V$LENTH,PF           PF2 = message length
        SAVEVALUE   1,V$RAND,XL            XL1 = random no. 0-.999
        SAVEVALUE   2,V$EFFI,XL            XL2 = channel efficiency
        TEST G      XL1,XL2,XMITj          if successful,
                                             then go to XMITj
        advance     1                      collision detection time
        SAVEVALUE   1+,1,XH                increament number of
                                             transmission trials
        ASSIGN      4,XH1,PB               PB4 = no. of trials
        TEST G      PB4,XB$MTRY,BACKj      too many attempts?
        TERMINATE   1                      then leave the model
BACKj   ADVANCE     V$RESCH                else delay the trans
        TRANSFER    ,SENDj                   go back to GBIQj
XMITj   SAVEVALUE   j,0,XH                 reset no of trans trial
        SEIZE       GBUS                   get the control of GBUS
        DEPART      GBIQj                  leave the GBIQj
        TRANSFER    ,BUS                   to to BUS
        ENDMACRO                           end of macro
*       Computes packet length using mean packet length and
*       FN$EXPO.
```

```
LENGTH FVARIABLE (XF$PACK-XH$MINPK)*FN$EXPO+XH$MINPK
*       Generates a random number between 0 and .999
RAND    FVARIABLE       RN4/1000
*       Calculates channel efficiency (P/C)/(P/C+W*S)
EFFI    FVARIABLE       (PF2/XF$CAPA)/PF2/XF$CAPA+XL$TWAIT*XL*SLOT)
*       Generates a random number between 0 and FN$MXINT
RESCH   FVARIABLE       RN3*FN$MXINT/1000
```

The last part of the model, transiting the bus, is coded as follows:

```
BUS     ADVANCE     V$XTIME             bus transit time
        RELEASE     GBUS                release the GBUS
        TRANSFER    ,DISTR              to to next node
*       Computes bus transition time
XTIME   FVARIABLE   (XL$SLOT+PF2/XL$CAPA)/XL$SLOT
```

System parameters can be changed simply by reinitializing savealues and matrix savevalues provided in the model segment 1. Based on the configuration used in this chapter a completed program listing and all output statistics can be provided by the author.

V. CONCLUSION

The model provided in this thesis is constructed to be used to evaluate the performance of a CSMA/CD bus LAN. Since the physical configuration of a SPLICE system is not fully specified and a specialized model is not able to answer new and unexpected questions, it was desired to build a more generalized model that can be used to evaluate various system configurations and workload characteristics of the SPLICE system. In order to meet the requirement, the model was divided into two model segments. Model segment 1 is dedicated to initializing the system configuration. The system simulation is performed by model segment 2. Based on the hypothetical data, modeling and programming of the two model segments were described.

REFERENCES

1. Schneidewind, Norman F., Functional Design of a Local Area Network for Stock Point Logistics Integrated Communications Environment, December 1982.
2. Schneidewind, Norman F. and Dolk, Daniel R., A Distributed Operating System Design and Dictionary/Directory for the Stock Point Logistics Integrated Communications Environment, November 1983.
3. Tanembaum, Andrew S., Computer Networks, Prentice-Hall, Inc., Englewood Cliff, New Jersey, 1975.

4. Gordon, Geoffrey, The Application of GPSS V to Discrete System Simulation, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.
5. IBM Technical Publications Department, GPSS User's Manual, IBM Company Publication Department, New York, 1971.
6. Abramson, N., "The Throughput of Packet Boradcasting Channels," IEEE Trans. Comm. 25, No. 1, pp. 281-285, January 1977.
7. Metcalfe, R. M. and Boggs, D. R., "Ethernet: Distributed Packet Switching for Local Networks," CACM, Volume 19, No. 7, pp. 395-404, July 1976.
8. Shoch, John R. and Hupp, Joh A., "Measured Performance of an Ethernet Local Network," CACM, Volume 15, No. 12, pp. 711-720, December 1980.
9. Shoch, John R., et al., "Evolution of the Ethernet Local Computer Network," Computer, Volume 15, No. 8, pp. 10-28, August 1982.
10. Watson, Bruce, "Configuration Dependent Performance of a Prioritized CSMA Broadcast Network," Computer, pp. 51-58, February 1981.
11. Tobogi, F. A. and Hunt, V. B., Performance Analysis of Carrier Sense Multiple Access with Collision Detection, Proceedings of the LACN Symposium, pp. 217-244, May 1979.
12. Weitzman, Cay, Distributed Micro/Minicomputer Systems, Prentice-Hall, Inc., Englewood Cliff, New Jersey, 1980.
13. Hughes, Herman D. and Li, Liang, A Simulation Model of the Ethernet, Department of Computer Science, College of Engineering, Michigan State University, Technical Report TR #82-008, 1982.