A COMBINED LAP-D/CSMA-CD SIMULATION MODEL FOR THE STUDY OF LOCAL AREA NETWORKS

by

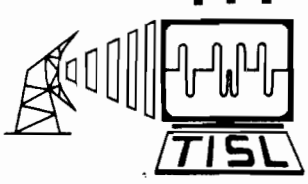
Mohammed S. Kashefipour
Graduate Research Assistant

Telecommunications and Information Sciences Laboratory
University of Kansas
Lawrence, Kansas 66045

Technical Report TISL-6731-4

Supported by
National Science Foundation
and
AT&T Information Systems

Principal Investigator: Dr. Victor S. Frost



TELECOMMUNICATIONS AND INFORMATION SCIENCES LABORATORY
THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.
2291 Irving Hill Drive - Campus West Lawrence, Kansas 66045

ABSTRACT

The performance of Local Area Networks(LANs) cannot be adequately predicted by modeling the media access or link access mechanism alone. A simulation model is developed that integrates the Link Access Protocol(LAP-D) and the Medium Access Protocol(CSMA/CD) into a single combined simulation model of a LAN, in order to predict the system performance. The model allows for explicit specification of the processing times for each protocol function. The integrated LAP-D/CSMA-CD model is used to investigate the sensitivity of the LAN performance relative to different timeout periods. Also observed the LAN performance under background load versus transport stations load. Finally, effect of window size on performance of LANs was studied.

Table of Contents

List of Figures

List of Tables

1.0 Introduction

A simulation LAN model is developed to predict the performance of local area networks. So far various studies[4,5] of local area networks have been performed with different degrees of accuracy model or predict the performance of LANs. The simulation model developed here is capable of more accurately predicting the performance of LANs. This model integrates the Logical Link Control Protocol (LAP-D) and Medium Access Control Protocol (CSMA-CD) into a single combined simulation model of LAN. In addition the model allows the specification of the processing times for each protocol function in the MAC and LLC. It should be noted that by integration of LLC and MAC and incorporating the associated processing times and network latencies more precise predictions of LANs can be obtained.

This report is divided into 6 chapters. The second chapter gives an overview of the International Organization for Standards(ISO) Reference Model and discusses the protocol hierarchies in the computer communications networks. The third chapter describes different LAN medias, topologies, and access methods, and compares LAN standard relationship to ISO Reference Model. Also this chapter gives an overview description of integrated LAN model. The fourth chapter of this report is devoted to a description of the Medium Access Control Protocol considered here and the implemented MAC model with associated validation results. The implemented MAC model(CSMA/CD) is regarded as a substitute for IEEE/Std. 802.3 [1] for the study of LANs. The fifth chapter describes the Link

Access Protocol D-Channel [2], also reviews the simulated LAPD primary functions, and finally validation of LAPD is given. The LAP-D protocol is very compatible with LLC IEEE/Std. 802.2 [3] which gives us the necessary tool the predict the performance of LANs. The final chapter of this report describes the integration of MAC(chapter 4) and LLC(chapter 5) in order to obtain an accurate performance prediction of LANs. This chapter describes the integrated simulation model in detail, also defines the processing times used in the integrated model. The performance of the integrated LAN simulation model under different timeout periods, load conditions, and window sizes are then discussed.

References:

1. ANSI/IEEE 802.3,' Local Area Networks - CSMA/CD',IEEE Inc., 1985

2. AT&T Information Systems,' Digital Multiplexed Interface Technical Specification',AT&T Information System, April 1985, pp IV-176, IV-259

3. ANSI/IEEE 802.2,' Local Area Networks - Logical Link Control ', IEEE Inc., 1984

4. B.Wood,' A Performance Evaluation of Local Area Networks',Technical Report TISL -6731-3,1986

5. K.Mills,M.Wheatly, and S.Heatly,' Prediction of Transport Protocol Through Simulation',IEEE Communications Society Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks,May 6-8, University of Kansas, Lawrence, Kansas 1986.

2.0 Overview of the ISO Reference Model

This chapter discusses the protocol hierarchies in the computer communication networks and the need for a structured standard protocols. The following sections of this chapter are devoted to describe the seven layers of OSI .

2.1 Protocol Hierarchies

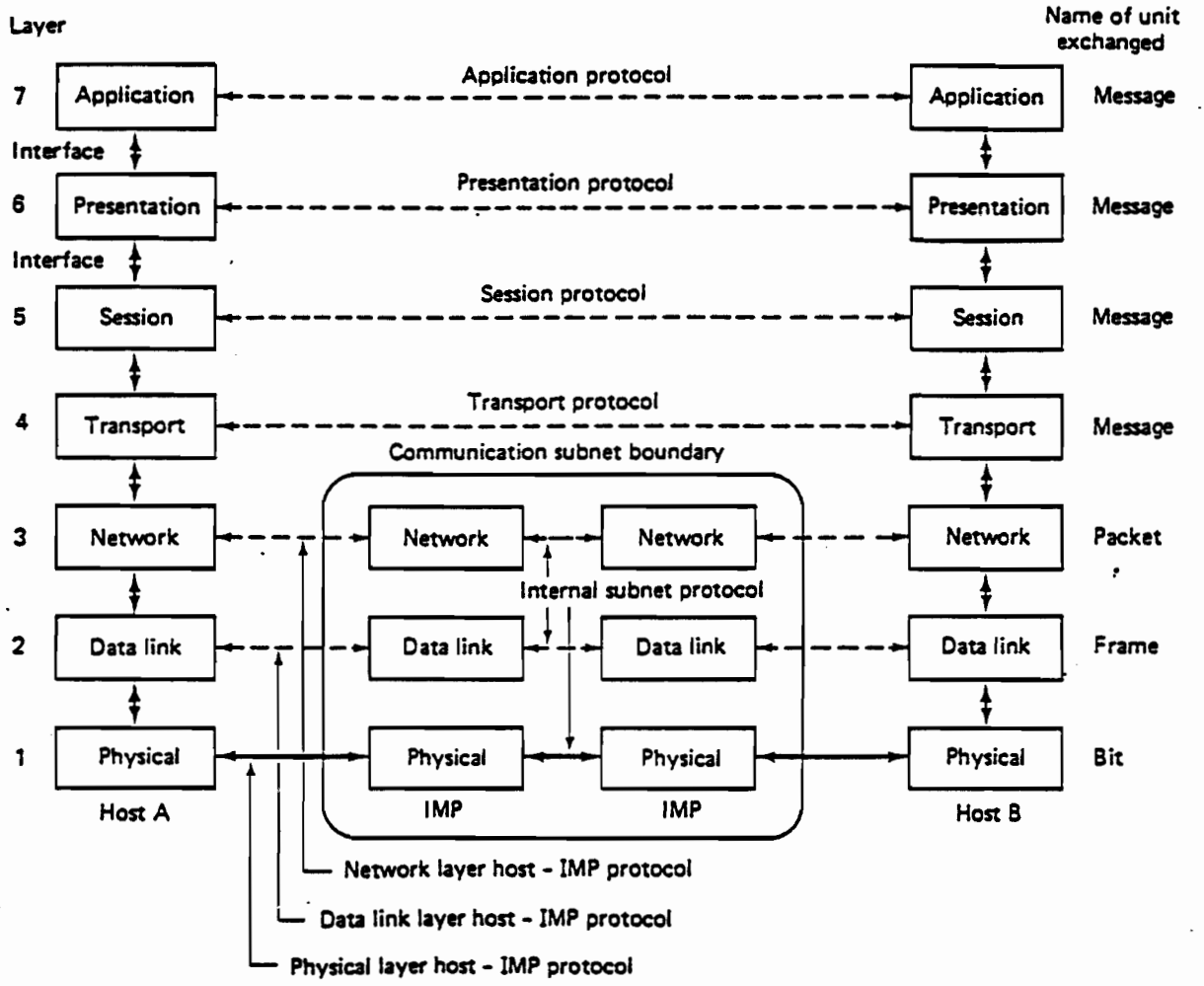
Most network architectures have little in common because they are designed to function with certain machines from different vendors. In 1977 the International Organization for Standardization (ISO) created the Open System Interconnection (OSI) Reference Model that serve as a framework which in turn defines a series of standard protocols. The ISO is the international standard group whose membership comprises the national standards organization of each member country. The ISO and the CCITT(Consultative Committee on International Telephone and Telegraph) usually try to cooperate on issues of telecommunication standards. In July 1979 the seven-layer OSI model was created.

The seven layers of the OSI model are shown in figure 2.1. Notice how this figure maps the seven layers with the terms,protocol,software programs, application programs , and system network architecture. If two machines carrying on a dialogue ,each layer in a machine carries on a conversation with an equal layer on another machine. In reality , the data are not transferred directly from , let 's say layer 3 to 3, but instead it is passed down

through the layers , from 3 to 2 and on to layer 1 , where it is finally transmitted over the physical link. There is physical communication between two machines only at lowest layer (physical layer) as opposed to the virtual communication used by higher layers. In figure 2.1 virtual communication is shown by dotted lines and physical communication by solid lines. Between each adjacent layer there is an interface that allows the data/information to be passed up and down the seven layers.

The purpose of these seven layers is to define the various functions that must be carried out when two machines communicate. Note that each layer of protocol/software performs its own specific functions and passes its portion of the message up or down depending upon whether the message is heading toward layer 7 or layer 1. In sections 2.1.1 through 2.1.7 each layer of the OSI model is discussed starting at the bottom layer.

Figure 2.1 Communications Using the Seven Layer OSI Model (From[2])



2.1.1 The Physical Layer

The physical layer is concerned primarily with transmitting data bits over a communication circuit. The main purpose of this layer is to define the rules by which one side sends(host) a "1" bit so that the other side(terminal) defines it as a "1" bit when it is received. This layer is concerned with very basic functions like voltages of electricity ,timing factors such as 1200 bits/sec or 833 usec/bit (bit time),full duplex or half duplex transmission, rules for establishing the initial connection , how to disconnect when the transmission is complete , and connector cable standards such as RS 232-C , RS 449 , or X.21.

It should be noted that layer 1 is the basic link over which all data pass. Communication between layers 2 - 7 at a host and layers 2 - 7 at a terminal are only virtual communications. In practice, the messages must be passed down to layer 1 for the actual movement of the messages between the host computer and a terminal.

2.1.2 The Data Link Layer

This layer is to provide an environment which is free of transmission errors for the transmission circuit which was established in layer 1. This error-free transmission link interfaces closely with layer 3 (network layer). The data link layer accomplishes its task by breaking up the input data into data frames , transmitting these frames and processing acknowledgment frames that are sent back to acknowledge the received data.

Since layer 1 accepts and transmits only a serial stream of bits without any regard to meaning or structure ,it is up to data link layer to create and recognize frame boundaries and check for errors during transmission. In addition this layer solves the problems caused by damaged ,lost,or duplicate frames so that the next layer will be working with error free messages.

Another issue at layer 2 is how to keep a fast transmitter from drowning a slow receiver. Some mechanism and procedure are employed to let the transmitting terminal know that the available buffer space at the receiver is filling to a critical level. This procedure and the error handling usually are integrated , although the problem of a terminal overrunning another terminal is also handled in some the layers above the layer 2. Some the typical data link level protocols are X.25 , High-level Data Link Control(HDLC) ,and Synchronous Data Link Control(SDLC). More detailed functions of data link layer are discussed in the subsequent chapters.

2.1.3 The Network Layer

This layer provides for the functions of internal network operations such as addressing and routing. The network layer actually controls the operation of the combined layers 1,2 and 3. This sometimes is called the subnetwork and/or the packet switching network function. Packets are created in this layer from messages and routed through the network.If a packet destined for another node is received ,this layer reroutes the packet down through layers 2

and 1 for transmission. It should be noted that the actual routing strategies are not defined in the Reference Model. What is defined is that the network layer may or may not maintain the sequence of the data handled for the transport layer.

2.1.4 The Transport Layer

This layer often is called the host-to-host layer. It provides the facilities that allow end users to pass messages between themselves across several intervening stations/nodes. Layer 4 also includes functions to do all user addressing, data assurance(control), and flow control of message from source to destination across simple or complex networks.

The transport layer is a end-to-end layer because a program at the source machine can carry on a virtual conversation with a similar program on a destination machine using message headers and control messages. The physical path still goes to layer 1 and across to the destination machine. The transport layer also can multiplex several streams of messages into a physical circuit by creating multiple connections that enter and leave each host computer. The transport header determines which message belongs to which connection. Layer 4 functions usually are carried out by software, but they are rapidly becoming part of the firmware as we get standards that are more clearly defined.

2.1.5 The Session Layer

The session layer is responsible for initiating, maintaining, and terminating each logical session between end users. In addition, this layer is responsible for managing and structuring all session requested data transport actions. Session initiation must arrange for all the desired and required services between session participants. Required services could be logging on to circuit equipment, use of various terminal types or features, security reliance, and the software tasks for half/full duplex. The session layer also might keep track of various accounting functions so the correct party receives the bill later.

The session layer is very close to the transport layer, although it has more application-oriented functions than the transport layer. Because the session layer usually is handled by host computer, it would be easy for the session and transport layers to be merged into a single layer.

2.1.6 The Presentation Layer

This layer performs a selectable set of message transformations and formating in order to present data to the end users. The presentation layer has items such as video screen formatting, peripheral device coding, encryption, and compaction. Other items such as number of printed lines per screen, characters per line, cursor addressing, and the like can be handled by this layer.

2.1.7 The Application Layer

The application layer is the end user's access to the network and therefore is developed by the individual user organization. Each user programs determines the set of messages and any actions it might take upon receipt of a message. The application layer also considers network management statistics, remote system initiation and termination, network monitoring, application diagnostics, making network transparent to users, simple processor sharing between host computers, use of distributed data bases, and industry-specific protocols such as banking and airline reservations.

References:

1. A.Meijer,P.Peeters,' Computer Network Architectures' ,Pilman Publishing Inc.,1982,pp 14-28

2. A.Tanenbaum,' Computer Networks ',Prentice Hall,1981

3. M.Schwartz,' Telecommunications Networks ',Addison-Wesley Publishing Company,1987

3.1 Overview of Local Area Networks

3.1.1 General Description

A local area network (LAN) has three distinctive characteristics which separates it from other computer communication systems[1].

1. A diameter of not more than a few kilometers
2. A total data rate exceeding 1 Mbps
3. Ownership by a single organization

A local area network is built to interconnect a variety of electronic office equipment , microprocessors , minicomputers , host computers, and other terminal equipment within an organization. Without a local area network the automated office can not function. This chapter discusses different media , topologies , and access methods used in local area networks. Also consider LAN standard relationship to ISO Reference Model . Finally the developed integrated LAN model will be discussed.

3.1.1.1 Media Selection

There are essentially three transmission media that dominates local networks installations : twisted wire pairs , coaxial cables , and fiber optics.

Twisted wire pair is a short-term LAN medium alternative because it is too limited in its capacity/distance relationship to support future high volume data transmission which is unique to the local area networks.

Coaxial cable comes in all sizes,shapes,and characteristics which each has some unique properties. Coaxial cable supports frequencies up to 300 or 400 MHz which meets the LAN high speed data transmission requirement. Today the main disadvantage of using coax is that coax is comparatively expensive to buy and to install and few buildings are prewired with coax[2].

Fiber optics uses a glass fiber that is drawn into a long cylinder which acts as a wave guide for light. The advantage of fiber optics as compare with coax and twisted wire pair is : extreme bandwidth capacity ,immunity to noise , and more secure against taps. It has more bandwidth available to meet current and future high speed data transmission. It is an excellent media for massive data transfer on a point-to-point basis but it becomes expensive as a multidrop line. The major disadvantage of fiber optics is that it is an immature technology and is difficult and expensive to cut into the medium for multiple users.

3.1.1.2 Topologies

Based on the transmission media , different cable topologies are possible. Three of these are shown in figure 3.1. In the star topology each node is attached to a central node , or called hub . The hub typically provides some sort of switching function or intelligence[2]. This can introduce some reliability problems for the network; if the hub fails , the network will be disconnected.

The most general topology is the tree in which groups of nodes are connected to form a tree as in figure 3-1(b). If there are two paths between some pairs of stations would suffer from interference between the two signals.

The shared-bus topology is a popular topology for local area networks. This is the topology used in the Ethernet LAN . In this topology each node simply taps into the bus. The controller is placed in the interface between the work-stations and the bus. This interface is known as a transeiver, and it connects directly to the bus using a passive connection. This way the cable does not have to be altered to add new stations.

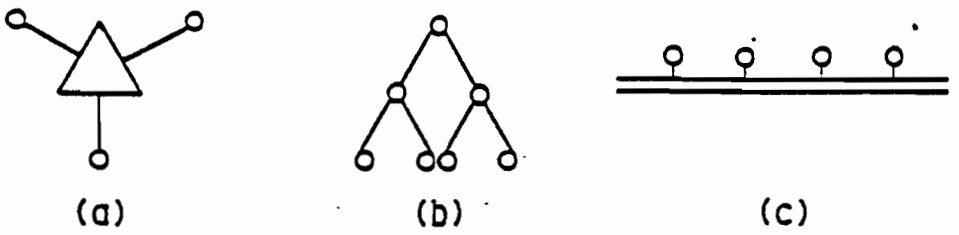


Figure 3.1 LAN Topologies (From[2])

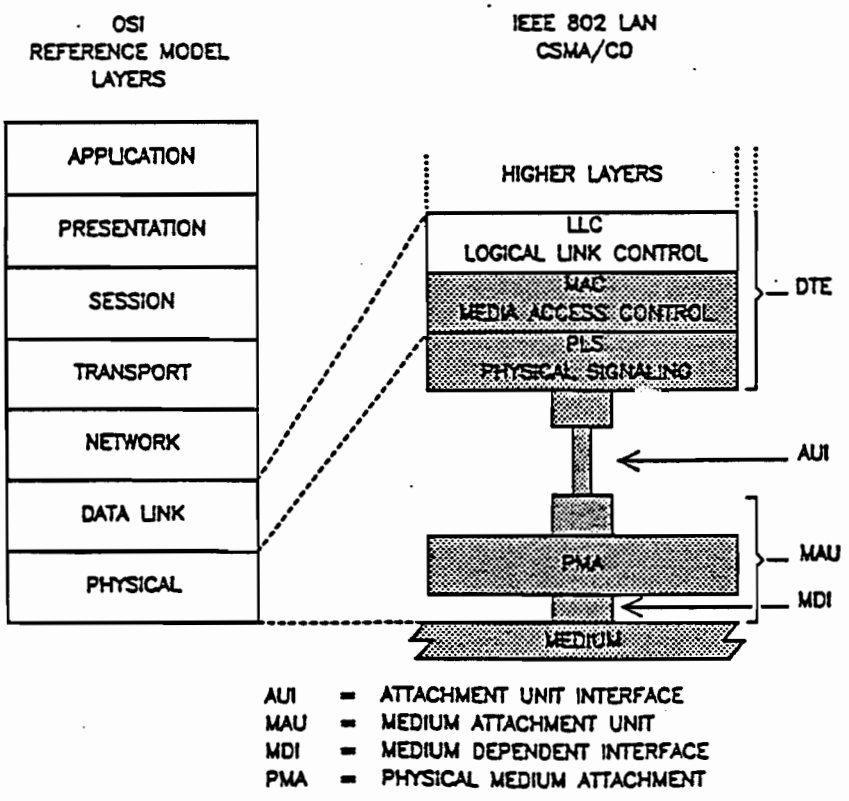


Figure 3.2 LAN Standard Relationship to the OSI Reference Model (From[12])

3.1.1.3 Access Methods

Some LAN functions require specific network access methods. The user can select either dedicated channels or random access schemes. Dedicated channels may be frequency(FDM) or time(TDM). Dedicated channels can be inefficient because capacity is wasted and the medium generally remains idle following the end of transmission. Contention or random access schemes appear to provide a simple and flexible access method that can dynamically accommodate changing user needs.

Three of the most common medium access protocols used in local area networks are defined by IEEE 802 Committee and are as follows : IEEE 802.3-CSMA/CD, IEE802.4-Token Bus, and IEEE 802.5 Token Ring. The description of CSMA/CD protocol is discussed in chapter 4 , but the description of IEEE 802.4 and IEEE 802.5 will not be given in this report. The redear is encouraged to refer to [3,4] for further details on Token Ring and Token Bus.

3.1.2 LAN Standards and Relationship to ISO Reference Model

The objective of the local area network standards is to ensure compatibility between equipment made by different manufactures. By using standards data communications can take place between devices with a minimum effort[5]. Standards must provide specifications of common interfaces and protocols for local area networks.

In August 1979 project 802 was proposed to the Institute of Electrical and Electronics Engineers (IEEE) Computer Society to develop a local area network standard . The task of IEEE 802 was to specify the means by which devices could communicate over a local area network. The IEEE 802 standards imposes the following requirements for the local area networks:

1. An operating range of up to 4 Km at the minimum data rate of 1Mbps
2. A bus or tree topology
3. Conformance with layers 1 and 2 of the OSI Model
4. To provide the communications needs of data devices
5. An application environment of the commercial or light industrial type
6. Low error rate

A local area network reference model is shown in figure 3.2 it is also compared to the ISO Reference Model. A local area network has three layers : physical layer , media access layer , and logical link layer. The IEEE 802 Committee does not define any standards for higher layers interface, but provides guidelines for the standard to satisfy users' needs with respect to existing higher level software , inter-network issues , addressing , and network management.

3.1.2.1 LAN - Physical Layer

This layer performs some basic functions as OSI physical layer. It is concerned with the nature of the transmission medium and the details of device attachment and the electrical signaling.

3.1.2.2 LAN - Media Access Control Layer

The media access mechanism provides the necessary rules by which two or more stations share a single transmission medium. The IEEE 802 currently defines three of the most popular media access methods:

1. IEEE 802.3 - CSMA/CD
2. IEEE 802.4 - Token Bus
3. IEEE 802.5 - Token Ring

3.1.2.3 LAN- Logical Link Control Layer

The Logical Link Control(LLC) layer is independent of the underlying access method and transparent to higher layers. This layer is concerned with establishing , maintaining , and terminating a logical link between devices. It provides a reliable communication path between two devices. The LLC supports the following functions:

1. Datagram : some form of connectionless service to support highly interactive traffic for sending and receiving Unnumbered Information(UI) frames

2. Virtual Circuit : virtual circuit is a connection oriented service which provides flow control , sequencing , and error recovery.

3. Multiplexing : a single physical link attaches a station to a LAN which provides data transfer with multiple end points over that link.

4. Multicast,broadcast : a service in which a station sends a message to multiple stations or all stations.

These services are specified in terms of primitives that can be viewed as commands or procedure calls with associated parameters. The unacknowledged connectionless service provides for only two primitives across the interface between the next highest layer and LLC. L-DATA.request is used to pass a frame to LLC for transmission. L.DATA.indication is used to pass a frame to higher layer entity from LLC upon reception. The connection oriented service includes L-DATA-CONNECT.request and L-DATA-CONNECT.indication which serve the same function as above , also in addition L-DATA-CONNECT.confirm which acknowledges the previous associated L-DATA-CONNECT.request. Table 1 summarizes the LLC services. The function of Logical Link Control primitives are:

1. Unacknowledged Connectionless Services. The primitives associated with unacknowledged connectionless data transfer are:

   + The L-DATA.request primitive is passed to the LLC sublayer to request a frame to be sent using unacknowledge connectionless procedures.

   + The L-DATA.indication primitive is passed from the LLC sublayer to indicate the arrival of a frame.

2. Connection Oriented Services. The primitives associated with connection oriented data transfer are:

   - The L-DATA-CONNECT.request primitive is passed to the LLC sublayer to request that a frame be sent using connection-oriented procedures.

   - The L-DATA-CONNECT.indication primitive is passed from the LLC sublayer to indicate the arrival of a frame.

   - The L-DATA-CONNECT.confirm primitive is passed by the LLC sublayer to convey the results of the previously associated L-DATA-CONNECT.request primitive.

   - The L-CONNECT.request primitive is passed to the LLC sublayer to request that a logical link connection be established between a local Link layer Service Access Point(LSAP) and a remote LSAP.

- The L-CONNECT.indication primitive is passed from the LLC sublayer to indicate the results of an attempt by a remote entity to establish a connection to a local LSAP.

- The L-CONNECT.confirm primitive is passed from the LLC sublayer to convey the results of the previous associated L-CONNECT.request primitive

- The L-DISCONNECT.request primitive is passed to the LLC sublayer to request the immediate termination of a link connection.

- The L-DISCONNECT.indication primitive is passed from the LLC sublayer to indicate to the network layer that a connection has been terminated.

- The L-DISCONNECT.confirm primitive is passed by the LLC sublayer to convey the results of the previous associated L-DISCONNECT.request primitive.

- The L-RESET.request primitive is passed to the LLC sublayer to request that a connection be immediately reset to the initial state.

- The L-RESET.indication primitive is passed from the LLC sublayer to indicate that a connection has been reset.

- The L-RESET.confirm primitive is passed from the LLC sublayer to convey the results of the previous associated L-RESET.request primitive.

- The L-CONNECTION-FLOWCONTROL.request primitive is passed to the LLC sublayer to control the flow from the LLC sublayer of L-DATA-CONNECT.indication primitives related to a connection.

- The L-CONNECTION-FLOWCONTROL.indication primitive is passed from the LLC sublayer to control flow from the network layer of L-DATA-CONNECT.request primitives related to a connection.

Unacknowledged connectionless service
L-DATA.request
L-DATA.Indication

Connection-oriented service
L-DATA-CONNECT.request
L-DATA-CONNECT.Indication
L-DATA-CONNECT.confirm
L-CONNECT.request
L-CONNECT.Indication
L-CONNECT.confirm
L-DISCONNECT.request
L-DISCONNECT.Indication
L-DISCONNECT.confirm
L-RESET.request
L-RESET.Indication
L-RESET.confirm
L-CONNECTION-FLOWCONTROL.request
L-CONNECTION-FLOWCONTROL.Indication

Table 3.1 Logical Link Control Primitives (From[5])

Both the virtual circuit and multiplexing capabilities can be supported with the concept of the Service Access Point(SAP). Figure 3.3 shows three stations attached to a local network in which each station has its own address. Further, the link layer supports multiple SAPs , each with its own address which LLC provides communication between SAPs[5]. Let describe an example to clearly

explain the communication between the SAPs . Referring to figure 3.3, suppose that a process or application X in station A is to send a message to a process in station C . In order to make this connection possible , X attaches itself to SAP1 and requests a connection to station C SAP1. Station A's link layer sends to the local area network a connection request frame. Then network delivers this frame to station C SAP1, if station C SAP1 is free, returns a connection-accepted frame to station A SAP1. Next all data from X will be packetized into a frame which includes source (A,1) and destination (C,1) addresses. Note that (A,1) can only accept frames from (C,1) for the set up connection , and (A,1) would declare itself as busy for other stations . At the same time , process Y can attach to (A,2) and exchange information with (B,1) which is an example of LLC multiplexing capability. In addition , various other processes in A could use (A,3) to send datagrams to various destinations.

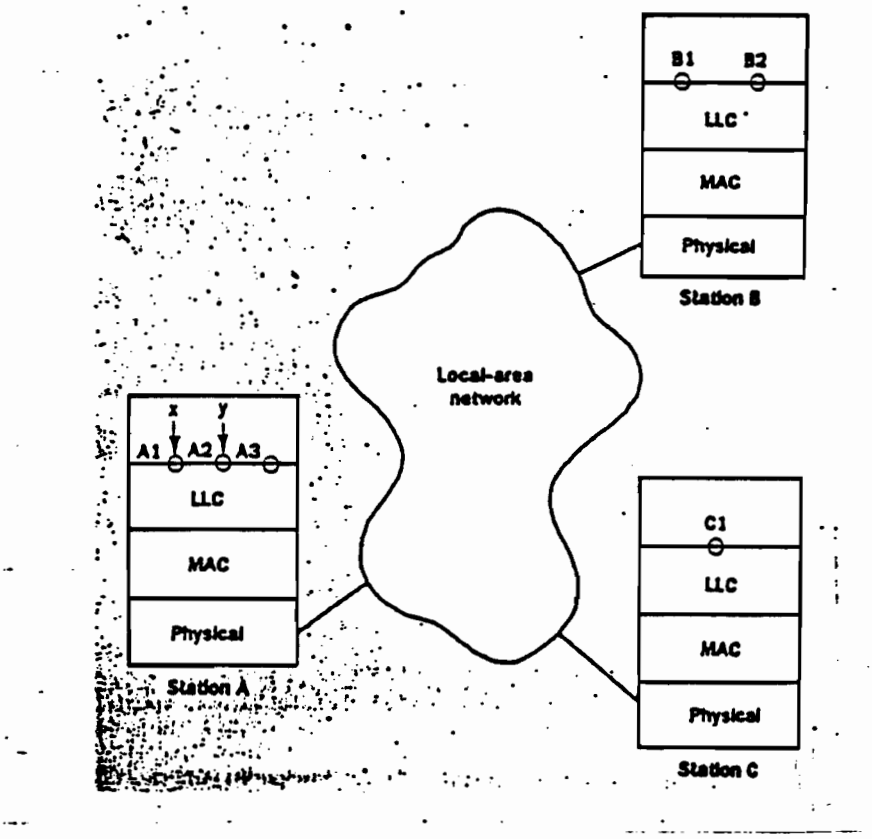


Figure 3.3 Local Area Network Link Control Scenario (From[5])

3.2 The Integrated LAN Model

Previous sections discussed the basic functions required by the IEEE 802 for the local area networks. This section provides an overview of the integration of Link Access Protocol(LAP-D) as the IEEE 802.2 LLC and Medium Access Protocol(CSMA/CD) as the IEEE 802.3 MAC into a single combined simulation model of a LAN. Chapter 6 of this report describes the integrated LAN model in detail.

The developed LAN simulation model is designed to predict the performance of local area networks. Up to now various studies of the local area networks has been performed which did not accurately model or predict the actual network performance. The past studies were more concerned with the performance of Logical Link Control Protocol (LAP-D) [6] or Medium Access Control Protocol (CSMA/CD) [2]. One study in particular combined LLC and MAC but did not consider the network software and hardware processing times and decision latencies in the model[7] which is a vital factor in network performance.

Recent studies[8,9,10] of the end-to-end performance for distributed applications of packet switching networks have shown that the performance at the user level is an order of magnitude lower than that at the physical link level. This phenomenon is explained by the fact of slow processing of packets at network interface which takes place at various levels of network architecture. Packet processing in local area networks are mostly performed in data link layer. Media Access Control Protocol normally is implemented in hardware which requires relatively short processing time and its implementation is on the order of that of the medium. On the other hand , the Logical Link Control Access Protocol is currently implemented in software which requires somewhat longer processing times. The packet processing delay could introduce a potential bottleneck in the performance of local area networks[8].

In order to make any improvements in performance of LANs, we must be able to predict the LAN performance accurately as a function of the network parameters. The simulation model developed is here is capable of predicting the actual performance of LANs. This model integrates the Logical Link Control Protocol and Medium Access Control Protocol into a single combined simulation model of LAN. In addition model allows the specification of the processing times for each protocol function in the MAC and LLC protocols. It should be noted by integrating the MAC and LLC protocols and incorporating the associated processing times and network latencies, more precise predictions of LANs can be obtained. In chapter 6, the description of processing times used in the MAC and LLC protocols will be defined, and also discus the performance and more details of the integrated model.

REFERENCES :

1. A.Tanenbaum,' Computer Networks ',Prentice Hall,1981,pp 286-288

2. B.Wood,' A Performance Evaluation of Local Area Networks ',Technical Report TISL-6731-3,1985

3. B.Moukadam,'A Simulation Study of a Token Ring LAN with Lognormal Message Interarrival and Heterogeneous Traffic',Master Project, University of Kansas, 1985

4. M.Torkzadeh

5. W.Stallings,' IEEE Project 802 ',Computerworld,Feb. 1984,pp 142-148

6. P.T.Brady,' Performance of LAPD Protocol with Link Errors and Propagation Delay ',ICC 1985,pp 121-125

7. K.Mills,M.Wheatley, and S.Heatley,' Prediction of Transport Protocol Through Simulation ', IEEE Communications Society Workshop on Computer Aided Modeling, Analysis And Design of Communication Links and Networks , May 6-8, University of Kansas, Lawrence Kansas, 1986

8. H.Kanakia and F.Tobagi,' Performance Measurement of a Data Link Protocol ',ICC 86,pp 894-898

9. W.Zwaenepoel,' Protocols for Large Data Transfers Over Local Area Networks ' , Proc. of 9th Data Comm. Sympo.,British Columbia, Sept. 1985, pp 22-32

10. K.A.Lantz,W.I.Nowicki,and M.Theimer,' Factors Affecting the Performance of Distributed Applications ',Proc. of ACM Sigcomm 84 Sympo. of Comm. Arch. and Protocols, pp 116-123

11. J.Rance,' IEEE Project 802 - A Status Review ',Proc. of Local Networks ,83(Europe),pp 399-413

12. ANSI/IEEE Std.,' Local Area Networks - Logical Link Control IEEE 802.2 ' ,IEEE,inc.,1984,pp 25-34

4.0 Description of the CSMA/CD LAN Model

CSMA/CD is the simplest form of Medium Access Control(MAC) adopted for IEEE 802. This technique was proposed by Metcalf and Boggs[7] in 1976 which has become known as Ethernet. The Ethernet is a baseband local network developed as a joint effort by Digital Equipment Corporation, Intel Corporation, and the Xerox Corporation.

This chapter reviews the CSMA/CD protocol, also gives a description of the implemented simulation model developed by TISL[3], and finally the validation for this model will be given.

4.1 Media Access Control Protocol - CSMA/CD

The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is very simple in concept. Don't get on the network and transmit at random - listen first! CSMA/CD protocol also referred to listen-before and listen-while transmitting. CSMA/CD is a refinement of the CSMA protocol which attempts to overcome the inefficiency of CSMA. Under CSMA, when two packets collide the medium remains unusable for the duration of transmission of both damaged packets. For packets that are long as compare with propagation time, the amount of wasted capacity can be considerable. This waste is reduced by using the collision detection mechanism, that is continue to listen to the medium while transmitting.

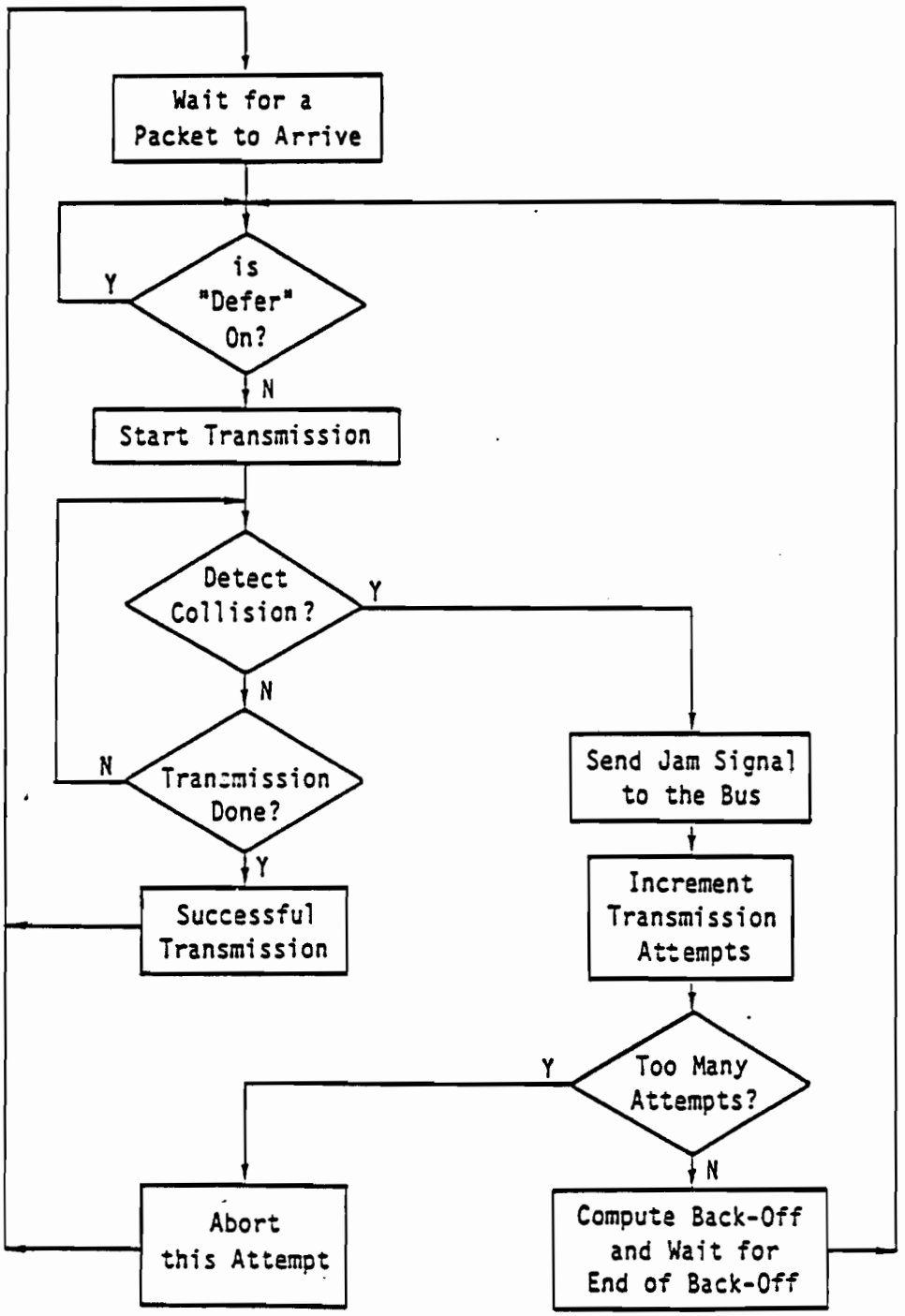
In the CSMA/CD a station listens or "senses" the communication channel to determine if any other stations are transmitting ,if another station is transmitting, then station will defer until channel is sensed idle. When the communication channel is no longer busy,the station will begin transmission after waiting the interframe spacing. While transmitting, station will monitor the channel to make sure the packet is received correctly.

Problems arise when more than one station try to transmit their packets at the same time. If this happens, the packets collide on the channel. When collision is detected, the station stops transmitting its packet and in turn transmits a jam signal to other stations on the network. The jam signal is used to inform all other transmitting nodes that a collision has occurred and they should stop transmitting. The station then calculates a random amount of time to defer transmission, and waits that amount of time before trying to transmit again. This waiting time is referred as backoff time.

In the CSMA/CD model described here, the backoff scheme is determined using a truncated binary exponential. This algorithm determines the backoff time by first sampling a uniform distribution,and then taking a random sample and multiplying it by the slot time. The size of the uniform distribution is dynamic and depends on the number of retransmission attempts. The complete description of backoff algorithm is given in IEEE 802.3 CSMA/CD page 39 - 40. The decision logic for the CSMA/CD protocol is shown in

figure 4-1.

Figure 4.1 CSMA/CD Flow Diagram (From[3])



4.2 CSMA/CD Simulation Model

The CSMA/CD LAN Model[3] developed is written in a simulation language called SLAM[6](Simulation Language for Alternative Modeling). The model is divided into two distinct parts. The first part is the Network Model, where SLAM blocks are used to simulate the arrival of packets to the network and the build up of packets at the node queues. The second part is the Discrete Event Model, where the CSMA/CD protocol and the propagation of the bus are simulated. The SLAM driver file for the simulated CSMA/CD model and the Discrete Event part of the model with the associated flow charts are given in[3]

The variables used to model CSMA/CD are divided into four types. The first variable type used are the SLAM variables. The second are model variables, the third are CSMA/CD variables , and the fourth are measurement variables. The description of the variables and the CSMA/CD simulated model are given in [2,3].

4.3 Validation of the Simulation Model

This section describes validation of the simulation model. To validate the accuracy of the simulation model from[3], a comparison between predictions derived from theory and those obtained via simulation model was constructed. The theoretical results were obtained from the work of Lam[4]. In [4] the average packet delay versus throughput was reported for CSMA/CD networks. The theoretical results are for various ratios of the propagation delay

to packet time. This ratio is known as "ALPHA" which is a critical parameter in CSMA/CD networks. However, for small ALPHA (0.001), it does not seem to make a difference in the network performance. The comparison between the simulation and the theoretical results are shown in figure 4.2. The parameters for the simulation model are listed in table 4-1.

Table 4-1

Simulation Model Parameters

ALPHA = 0.0017

Slot time = 20 usec

Attempt Limit = 16

Backoff Limit = 10

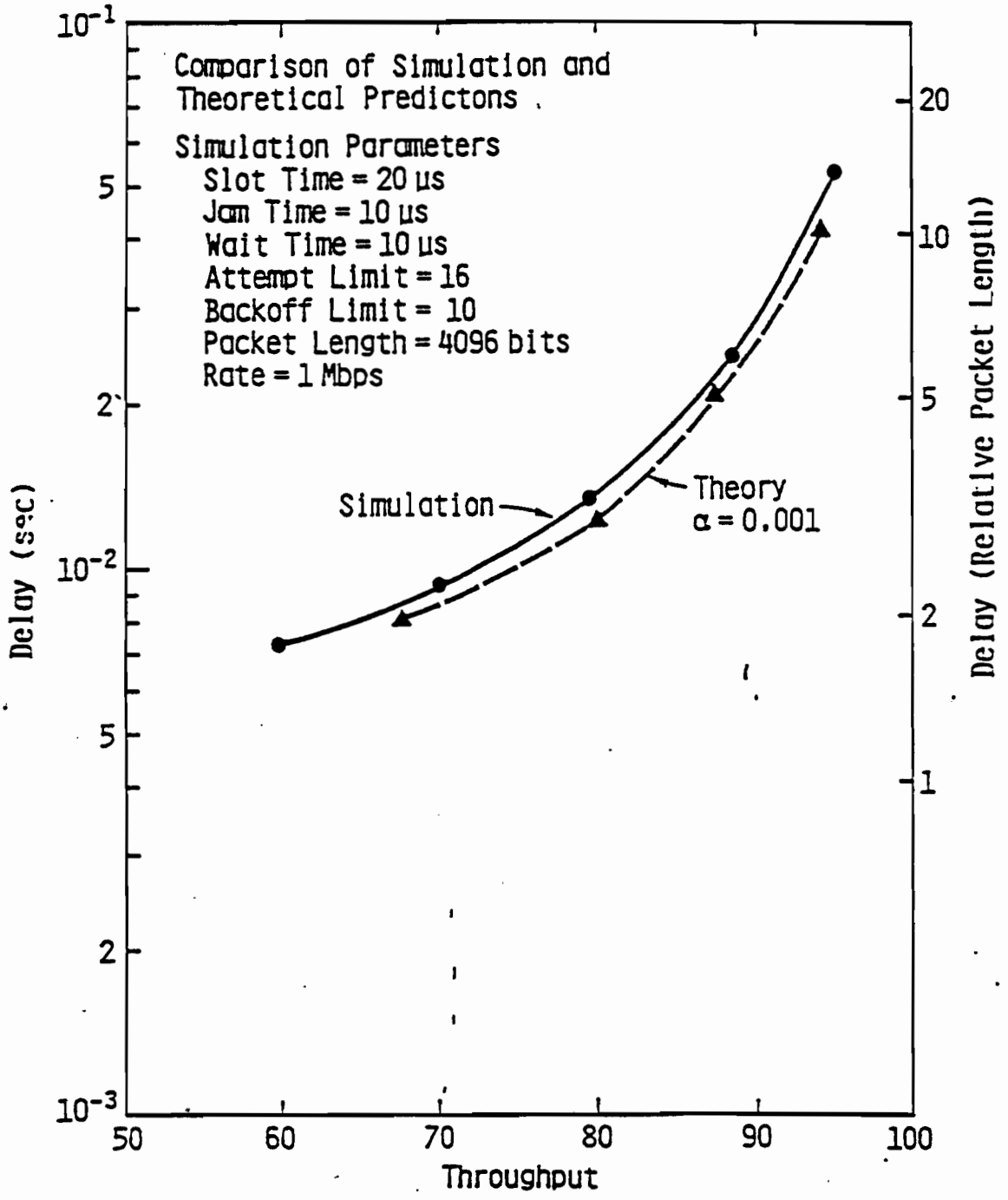
Packet Length = 4096 bits

It can be noted that there is a close agreement between the theoretical and simulation results.

Eventhough the results presented in figure 4-2 indicates that the simulation model is a accurate, there is factor that must be considered before interpreting the simulation results. Primarily, as the slot time increases, the variance on the delay estimates increases. This phenomenon is explained by the fact that the backoff algorithm uses the slot time to calculate backoff time. Therefore larger slot time would imposes a greater variance on the packet which in turn cause an increase in the overall delay. Thus, under high loads, there will be more collisions which causes lager

backoff ranges and backoff time variances. Clearly care must be taken in interpreting the simulation results for highly loaded CSMA/CD networks[3].

Figure 4.2 Simulated VS. Theoretical Performance (From[3])



References:

1. W.Stallings,' Data and Computer Communications ',Macmillan Publishing Company,NewYork,1985,pp 344-346

2. E.M.Friedman,' Computer Simulation Model for a CSMA/CD Local Area Network Utilizing a Multirate Voice Coding Technique for Load Control',Technical Report,TISL - 6731-2,May 1985

3. B.Wood,' A Performance Evaluation of Local Area Networks',Technical Report,TISL - 6731-3,January 1986

4. S.S.Lam,' A Carrier Sense Multiple Access Protocol for Local Networks', Computer Networks, No.4, April 1980, pp 21-32

5. ANSI/IEEE 802.3,' Local Area Networks CSMA/CD ',IEEE Inc., 1985

6. A.B.Pritsker,' Introduction to Simulation and SLAM II ',Halsted Press Inc., 1984

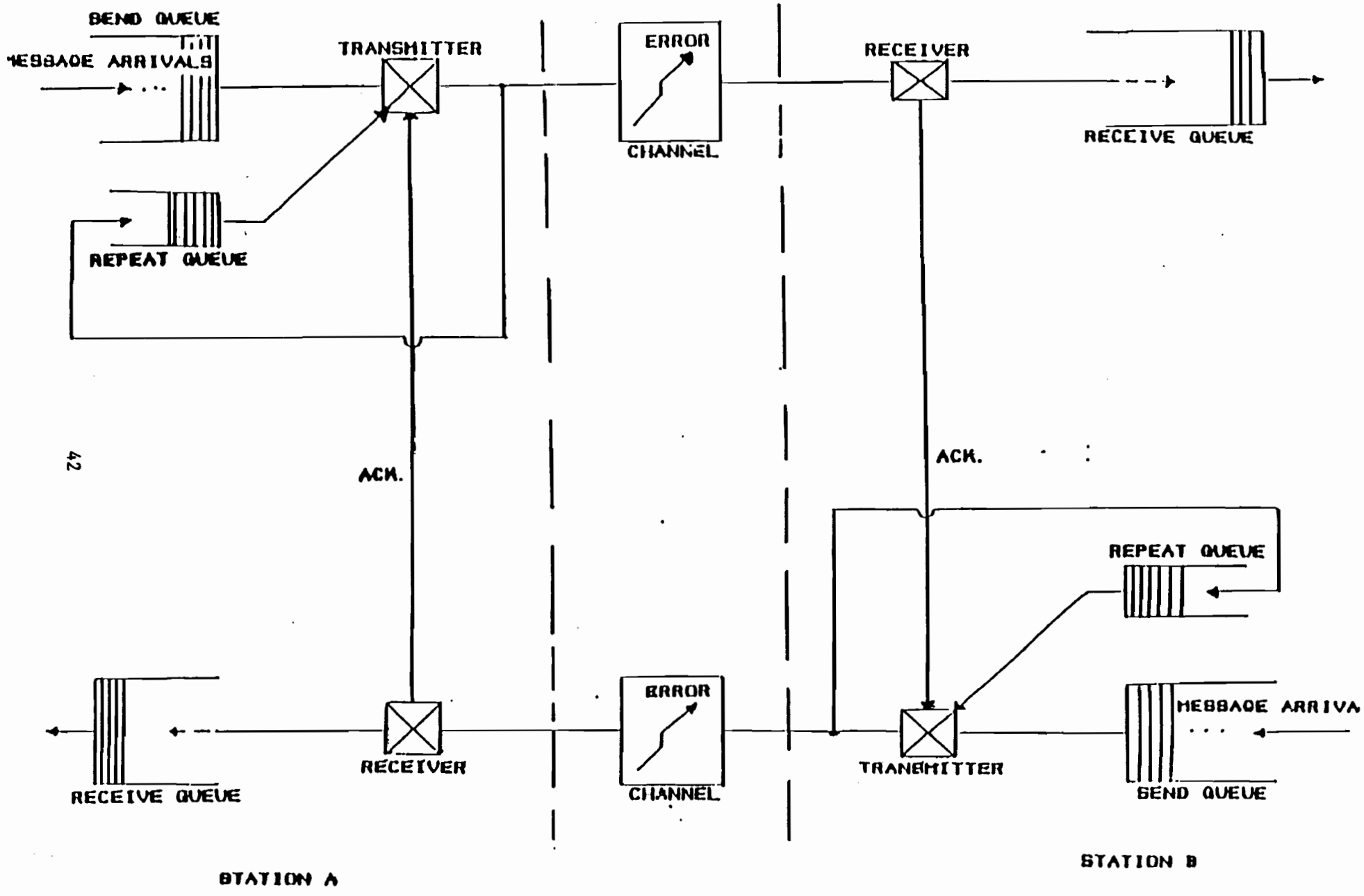
7. R.M.Metcalfe,and D.R.Boggs,' Ethernet:Distributed packet Switching for Local Computer Networks', Commun. ACM, Vol. 19, pp 395-404, July 1976

5.0 The Link Access Protocol - D Channel

This chapter describes the simulated Link Access Protocol D-Channel and validates the LAPD simulation model. The Data Link Layer is the second layer of the OSI Reference Model which is consists of two sublayers namely Link Access Protocol(LAP) and Medium Access Protocol(MAC). Link Access Protocol-D channel is a multiplexed version of the LAPB protocol which is based on CCITT Recommendation Q.921[1] that defines the ISDN User-Network Interface Data Link Layer Specification. A two octet address field is provided to designate the Data Link Connection Identifier,also supports the Modulo 8 and Modulo 128 (Extended) sequence numbering[1].

LAPD protocol is directly related to the Logical Link Control (LLC) sublayer specified by the IEEE/Std 802 Local Area Network Protocol. As it mentioned in chapter 3 the LLC sublayer constitutes the top sublayer in the Data Link Layer. Although there might be some differences between LAPD and IEEE's LLC in terms of call set ups and call disconnects, the simulated portion of LAPD protocol is identical to the procedures defined by IEEE/Std 802.3(LLC) which this gives us the necessary tool to study the performance of LANs by combining the LAPD and CSMA/CD. Figure 5.1 shows a Link Level Model for LAPD protocol.

Figure 5.1 Link Level Model For LAPD Protocol



5.1 Link Access Protocol Description

Before describing the LAPD protocol, let define the frame structure , the protocol variables and protocol functions.

5.1.1 LAPD Frame Structure

This section defines the Data Link Layer frame structure for data communication systems using bit-oriented procedures. Figure 5.2 shows LAPD frame structure. The frame has the following fields:

- Flag : 8 bits
- Address : 16 bits
- Control : 8 or 16 bits
- Data : Variable
- Frame Check Sequence(FCS) : 16 bits
- Flag : 8 bits

The flag, address, and control fields that precede the data field are known as a Header. The FCS and flag fields following the data field are referred to as a Trailer.

5.1.1.1 Flag Fields

All frames are delimited by 'flag' sequence 01111110. This sequence can never occur in the data of any frame because of the 'bit-stuffing' transparency mechanism. The transmitter will always insert an extra 0 bit after each occurrence of five 1's in the frame. This additional bit is removed whenever five contiguous 1 bits are detected at the receiver, except for the case of a flag being detected. The function of the flags is to provide frame level synchronization.

5.1.1.2 Address Field

The address field is used to identify the source and destination of an end-to-end connection. An address is 8 bits long, but by prior agreement an extended format(16 bits) may be used. The single octet address of 11111111 is interpreted as the all-stations address in both basic and extended formats. It is used to allow the primary to broadcast a frame for reception by all stations.

5.1.1.3 Control Field

The control field defines the exact function of the frame, there are three basic types : Information, Supervisory, and Unnumbered. These are referred to as I-frames, S-frames, and U-frames respectively. The formats of these three types of frames are now defined:

- The I-frame format is used to provide information transfer across the link. I-frames also used for acknowledgement purposes by putting the acknowledgement in them.

- The S-frame is used to perform supervisory functions on the link such as to acknowledge I-frames, request retransmission, and suspension of sending.

- The U-frame implements link control without sequence numbers, they are particularly used to bring a line up from a state where sequence numbers can not be known.

FLAG	ADDRESS	CONTROL	DATA	FCS	FLAG

(a) Frame format

	1	2	3	4	5	6	7	8
I: Information	0	N(S)			P/F	N (R)		
S: Supervisory	1	0	S		P/F	N (R)		
U: Unnumbered	1	1	M		P/F	M		

N (S) = Send sequence number
N (R) = Receive sequence number
S = Supervisory function bits
M = Unnumbered function bits
P/F = Poll/final bit

(b) Control field format

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Information	0	N(S)							P/F	N(R)						
Supervisory	1	0	S		0	0	0	0	P/F	N(R)						

(c) Extended control fields

Figure 5.2 LAPD Frame Structure(From [11])

5.1.1.4 Data Field

The data field is present only in I-frames and some unnumbered frames(datagrams). The length of data field is a multiple integer of 8 bits. The maximum length is not defined by the standard, but is generally limited by each implementation (12144 bits for integrated LAN model from the constraints of the IEEE 802.3 CSMA/CD specification).

5.1.1.5 Frame Check Sequence Field

The Frame Check Sequence(FCS) algorithm is that described in ISO Standard. The FCS field is a 16-bit sequence used to detect any transmission errors occurred in a frame.

5.1.2 LAPD Variables and Functions

The purpose of this section is to define the variables used in the LAPD protocol and also describe some of the functions that are simulated in our LAPD simulation model. The first part of this section defines the protocol variables and the second part describes the LAPD functions.

5.1.2.1 LAPD Variables

The LAPD protocol variables are defined as follows:

1. Send State Variable V(s) - This variable contains the sequence number for the next I-frame to be transmitted. The value of V(s) is incremented by one after the transmission of each I-frame. Its value can not exceed V(a)(see below for V(a) definition) by more than the maximum number of outstanding I-frame k(window size). The value of K may be in the range of $1 <= K <= 7$ for modulo 8 operation and $1 <= K <= 127$ for modulo 128 operation

2. Acknowledge State Variable V(a) - the Acknowledge State Variable identifies the last frame that has been acknowledged by other node. The value V(a) will be updated by the valid N(r) value received from the other node.

$$V(a) <= N(r) <= V(s)$$

3. Send Sequence Number N(s) - this is the sequence number of an I-frame. The value of the V(s) is set equal to the value of the N(s) in the I-frame before transmission.

4. Receive State Variable V(r) - this variable contains the value of the next in sequence I-frame to be received. It is incremented by one with the receipt of an error-free and in sequence I-frame whose N(s) is equal to V(r).

5. Receive Sequence Number N(r) - N(r) indicates that the data link layer entity transmitting the N(r) has correctly received all I-frames numbered up to and including N(r) - 1

.

6. K - the maximum number of outstanding I-frames at any given time .

7. N200 - the maximum number of retransmissions of a frame .

8. T200 - The acknowledge timer which is the maximum length of time allowed for a response . This variable is also known as timeout period. The default value is 1.0 second .

5.1.2.2 LAPD Functions

The various LAPD functions are given as follows:

- Information(I) is used to transfer data across a logical data link connection, sequentially numbered frames.

- Receive Ready(RR) indicates to the other end of a link that the receiver is ready to receive I-frames, it also acknowledges I-frames with a N(s) sequence number up to N(r) - 1. In addition , RR frame is used to clear a busy condition which was set by the transmission of an Receive Not Ready frame.

- Reject(REJ) requests the retransmission of I-frames starting at the one indicated by the value N(r) which is to be the sequence number of the next packet required. It is sent when a station's receiver receives a frame whose N(s) is out of sequence. It is also used to clear a busy RNR condition. The value of N(r) in the REJ frame acknowledges

I-frames numbered up to and including N(r) - 1.

There are other functions defined by LAPD protocol for link set ups, link disconnects , Frame Reject(violation of maximum and minimum frame sizes), and Receive Not Ready(no I-frames can at present be accepted because of buffer shortage). These functions are defined in [1] and are not modeled in our LAPD simulation model, so the description of these functions are omitted.

5.1.3 Protocol Description

This section describes the information transfer procedures for sending and receiving I-frames, and also the Receiving REJ frame procedures and timeout recovery procedures will be defined. These are the primary functions considered in LAN simulation model developed here.

5.1.3.1 Information Transfer

The procedure for sending and receiving I-frames is now described. When the transmitter has an I-frame to send (assuming that an RR is outstanding) it will send it with an N(s) equal to its current value of V(s). A timer is started when the transmission commences. If the value of V(s) is equal to the value of the last N(r) received plus the window size then further transmission of I-frames are suspended until a frame containing an advanced N(r) is received except for possible retransmissions that are required.

When an I-frame is received by node B with a correct Frame Check Sequence (FCS) and a sequence number N(s) equal to receiver's V(r), the information contained in the I-frame is accepted and the data is passed to the upper layer entities. The value in the receiver's V(r) will be advanced to the value contained in the received packet N(s). If an I-frame is ready to be sent out by this node,then node B's transmitter will set the N(r) of the I-frame to be transmitted to the current value of V(r). If there is no I-frames ready for transmission ,then this node would transmit a Receive Ready (RR) Supervisory frame . The RR frame contains N(r) set to the current value of V(r).

If an I-frame with an invalid FCS is received ,it will be discarded . If the received I-frame has a correct FCS, but the value of N(s) in the received frame is not equal to the value of receiver's V(r), then N(r) contained in the I-frame (piggyback acknowledgement) is processed, but the rest of the frame is ignored. The receiving node informs the transmitting station about the out of sequence frame by sending a Reject (REJ) frame with its N(r) one greater than the value of the last I-frame received in sequence. Whenever an information or supervisory frame which contains a N(r) is received, it is considered to be an acknowledgement of all frames up to the value of N(r) - 1. If timeout period expires before receiving an acknowledgement for a transmitted I-frame, procedure for timer recovery condition is followed(see section 5.1.3.3). If a Reject (REJ) supervisory frame is received , the procedure for Receiving REJ frame would be followed(see section 5.1.3.2).

5.1.3.2 Receiving REJ Frame

Refering to figure 5.1,if station A receives a REJ frame from station B, the station A transmitter would set its V(s) to the value contained in the REJ frame and commences to retransmit the I-frame in which REJ frame refers. If other I-frames have been sent with a N(s) greater than the sequence number of the frame in which the REJ frame refers, then these frames must also be retransmitted.

If the station A transmitter is already in the process of sending an I-frame, it may abort the transmission of that I-frame. Also if the transmitter was sending a supervisory frame when the REJ frame was received, then it would complete the transmission of that S-frame. Finally if there was no transmission in progress then the transmission of the rejected(requested) frames starts immediately.

5.1.3.3 Timer Recovery Condition Procedures

If the timeout period expires at any time indicating that the remote end of the link has not acknowledged frames within the timeout period, then the procedure for timeout recovery condition is followed.

A single I-frame or the last I-frame in a sequence of I-frames can not be recovered by REJ function. Also an I-frame may be lost due to transmission errors. Furthermore, it is not allowed to repeat the REJ recovery(assuming one REJ is outstanding) if a retransmitted I-frame is disturbed. Therefore, each combined

station uses a timer to recover I-frames from such situations. since it can happen that an I-frame has been correctly received, but the acknowledgement has not been received, a RR supervisory frame command is issued upon expiration of the timeout prior to retransmission to avoid a duplication of the I-frame already sent. The timeout condition is only cleared by the receipt of a response frame with its N(r) advanced. The value of the transmitter send state variable V(s) is set to the value N(r) contained in the received response supervisory frame. Then if it is necessary, I-frame(s) starting with N(r) sequence number will be retransmitted.

5.2 LAPD Simulation Model

In this section the LAPD simulation model developed here will be outlined. The model is throughly validated and accurately simulates Data Link Control procedures. The validation results are given in section 5.3.

LAPD protocol has been selected as a substitute for IEEE 802.2 LLC to predict the LANs performance. The procedures implemented are identical to the procedures given in IEEE 802.2 LLC which enables us to study LANs more efficiently. By having the primary functions implemented in our simulation model, we are able to gain insight into the detailed behavior of the procedures both during error-free and error recovery operations. The results obtained shows that delay performance is affected by rather complex interactions among the various protocol mechanisms.

In addition to the implemented protocol procedures, model allows transmission errors for I-frames, propagation delay, processing delays for each protocol function, and network latencies. The model also is flexible enough to allow for different distributions message arrivals and message length. It also permits user to change any of the protocol or modeling parameters,e.g. timeout period, window size.The LAPD simulation model does not implement the call set ups, call disconnects, Receive Not Ready(RNR), and Unumbered I-frames(UI). The simulation model implements the LAPD functions previously described(see section 5.1.2.2). LAPD simulation model considers the following modeling assumptions :

1. Setting up the link has been appropriately handled.

2. The link established is available and operational for the time interval required.

3. Infinite buffer capacity for the receivers which eliminates the implementation of function Receive Not Ready (RNR) from the simulation model. Receive Not Ready function is used when receiving end buffer space reaches the critical level.

4. Bit errors occur at independent random times(no burst errors).

5. Error free transmission for supervisory frames

The LAPD simulation model consists of two parts: the first part is the Network Model, where SLAM[12] Network blocks are used to simulate the arrival of packets to the network and the build up of packets at the node queues. The second part is the Discrete Event Model which simulates the LAPD protocol procedures. The LAPD Network Model and the Discrete Event part are given in Appendix 1 of this report.

5.2.1 Variables and File Assignments

In previous sections, we defined the protocol description, variables, and functions. This section defines the variable descriptions and file assignments which are used in implementation of LAPD protocol. By defining these variables, reader can easily relate the actual LAPD protocol to the simulated LAPD, and also be able to understand the integrated LAPD/CSMA-CD which is described in chapter 6.

The variables used to model the LAPD protocol are divided into four types. The first variable type used are the SLAM[12] variables to describe the network topology and define some of the LAPD protocol variables. The second are model variables which are used to model the Link Access Protocol. The third are LAPD variables, and the fourth are measurement variables.

5.2.1.1 SLAM Variables

The SLAM variables include the attributes, TNOW, NCLNR, and global variables. The TNOW variable is the current time of simulation in micro seconds. The NCLNR variable is the file number of the event calendar. The attributes are an array of variables carried with the packet as it passes through each process in the model. The attribute array of each packet has the same definition of a particular attribute, while the actual value for the attribute may be different[2]. The definition of each attribute is listed below.

atrib(1) = packet creation time

atrib(2) = source node

atrib(3) = destination node

atrib(4) = file number of the repeat queue

atrib(5) = packet count number

atrib(6) = packet Send Sequence Number(N(s) Protocol Version)

atrib(7) = packet timeout period

atrib(8) = the communication indicator between Data Link Layer and Application Layer are defined as follows :

atrib(8)=1.0 indicates that LAP is ready to accept I-frames from application layer

atrib(8)=2.0 indicates that LAP is sending an I-frame to the application layer

atrib(9) = packet length in micro seconds

atrib(10) = used for timer resetting purposes and is defined as follows:

atrib(10)=0.0 stop timer for acknowledged I-frames

atrib(10)=1.0 stop timer only for one particular I-frame. This is the case when node is in timer recovery condition.

atrib(10)=2.0 stop timer for all transmitted I-frames. It is used when node is in REJ recovery condition.

atrib(11) = number of bits per packet

atrib(12) = measures LAP packet delay

atrib(13) = not used

atrib(14) = signifies if frame is correct, in error, or out of sequence

atrib(14)=0.0 correct frame

atrib(14)=1.0 frame in error(transmission error)

atrib(14)=2.0 out of sequence frame

atrib(15) = indicates the type of frame(command/response)

atrib(15)=1.0 command frame

atrib(15)=2.0 response frame

atrib(16) = indicates the packet count number that receiver has received plus one

atrib(17) = the packet sequence number that receiver expects to

receive (V(r) protocol version)

atrib(18) = indicates the frame I.D. and is defined as follows:

atrib(18)=0.0 I-frame

atrib(18)=1.0 S-frame (RR frame)

atrib(18)=2.0 S-frame (REJ frame)

atrib(18)=3.0 S-frame (RR frame for timer recovery)

atrib(19) = indicates that the Data Link Layer entity transmitting , has received all I-frames numbered up to including atrib(19) - 1 (N(r) - 1 protocol version)

atrib(20) = timeout event indicator

atrib(20)=0.0 non timeout event

atrib(20)=1.0 timeout event

5.2.1.2 Model Variables

The model variables are to simulate certain aspects of the LAPD protocol. These variables are defined as follows :

- ndsts(i) = 0 node(i) is in normal transmission of I-frame status

- ndsts(i) = 1 node i is going in retransmit status
- ndsts(i) = 2 node i is in retransmit status
- irejsts(i) = 1 node i is in REJ recovery condition
- itmot(i) = 1 node i is in timeout recovery condition
- irxsts(i) = 1 node i is retransmitting
- ichksts(i) = 0 node i is idle
- ichksts(i) = 1 node i is transmitting an I-frame/S-frame
- jchksts(i) = 0 node i is not transmitting an I-frame
- jchksts(i) = 1 node i is transmitting an I-frame
- sfrm(i) = 0 no S-frames waiting for transmission at node i
- sfrm(i) = 1 S- frame waiting for transmission at node i
- ifrm(i) = 1 transmission termination for I-frame in progress for node i
- jfrm(i) = 1 I-frames waiting for transmission at the LAP level
- bfrcnt(i) = 1 number of outstanding frames waiting for retransmission

- ipktrej(i) = sequence number of I-frame causing the REJ condition at node i

- jsnd(i) = temporary storage of packet transmission sequence number for node i

5.2.1.3 LAPD Protocol Variables

The third variable type are the LAPD variables. These variables are particular to the LAPD protocol. The LAPD protocol variables are outlined as follows :

- isnd(i) = The next packet in sequence to be transmitted by node i (V(s) protocol version)

- rcvsvr(i)= the sequence number of the next frame to be received by node i (V(r) protocol version)

- iakvar(i)= it identifies the last frame that has been acknowledged by the node i (V(a) protocol version)

- krcv(i) = The sequence number in which node i expects from node j(secondary node) to receive. This variable is assigned to atrib(19)(N(r) protocol version) of each outgoing I-frame(piggyback acknowledgement). it indicates that LAP has received all I-frames numbered up to and including krcv(i) - 1 .

- iwndsz = the maximum number of outstanding I-frames at any given time(window size)

The timeout period is not defined in terms of node bases, but is defined based on each packet (ATRIB(7)) which enables us to consider variable timeout periods. This condition would be the case , if it is desired to have variable timeouts for exponential message length(different transmission times) or variable timeouts based on the offered load in the network(different acknowledgement times).

5.2.1.4 Measurement Variables

The fourth and final variable type are the measurement variables. These variables are used to measure statistical information on the network. Many of the measurement variables are used to measure performance of the network, such as the throughput. The measurement variables are described as follows :

- pktcnt(i) = number of packets created at node i
- totpktcnt(i) = total number of packets transmitted by node i including retransmissions
- itmoutcnt(i) = total number of timeouts by node i
- irxmtcnt(i) = total number of retransmissions by node i

- icount(i) = total number of frames in error from node i
- iotsqcnt(i) = total number of out sequence frame from node i
- idmgpktcnt(i)= total number of damaged frames(trans. error) from node i
- iabrtpkt(i) = total number of frames aborted during transmission by node i
- ircvcnt(i) = total number of frames received by node i
- infobts(i) = total number of information bits transmitted by node i
- rrfrmcnt(i) = total number of RR frames transmitted by node i
- sfrmcnt(i) = total number of REJ frames transmitted by node i
- rrsts(i) = total number of RR timeout inquiry status frames transmitted by node i

In addition to user collected statistics, the model also provides some delay statistics on the network. The measured statistics by SLAM are as follows:

USR to LAP Q DLY = time from packet creation to arrival at LLC level

AVE. LAP DLY = The average time delay of an I-frame transmitted from one LLC and received successfully by the other LLC(not including application buffer delay)

USR(i) to USR(j) = end-to-end delay between nodes i and j

AVE. USR DLY = average end-to-end delay of all workstations in the network

AVE. BFR HLDNG TM= average time that a message has to be stored in the repeat queue until a positive acknowledgement is received.

AVE. MESSAGE SIZE= average I-frame size

AVE.NUM in SYSTEM= average number of I-frames in the system(end-to-end)

5.2.1.5 File Assingments

There are several types of files in the LAPD simulation model. The first type are the node queues. These are files which are used to store packets in the end user queue. The node queues are really SLAM[12] AWAIT blocks. Each node on the network has a station queue or AWAIT file associated with it, where packets will wait until the LAP is ready to accept the next one(end of transmission).

The second file type are LAP retransmission queues which are used to store packets that are transmitted by LAP but are not yet acknowledged by the secondary node. The size of these retransmission queues are limited to window size. Data Link Layer will stop transmitting I-frames, if the retransmission queues reach the maximum limit.

The third file type is used to store the supervisory frames that are waiting for transmission. This file is used when a node's transmitter is busy and also have a S-frame to send. Node will store the S-frame in the S-frame Queue until transmitter becomes idle.

The fourth type file is called channel queue with zero service time. This file allows only one packet at the time to enter the channel(Discrete Event Model)

The fifth type file is called the event calendar which is numbered as NCLNR. The event calendar is a file that holds the specific events which are due to occur. Events are placed on the calendar in chronological order by SLAM, and at the proper time are executed. In Network Model, SLAM automatically places events on the event calendar. In Discrete Event model, the user places events on the calendar and can remove events from the calendar. The last file type are the statistics collection files. These files store the statistical information being collected.

5.3 Validation of the LAPD Simulation Model

In this section, an experiment which duplicates Bux's analytic model[4,5,6] for HDLC Mean Transfer Time / I-frame Transmission time versus load will be presented. The experiment performed serves as a way to validate the LAPD simulation model. In order to make an appropriate comparisons between LAPD results and HDLC results, only Asynchronous Balanced Mode(ABM) of HDLC is considered for validation. The procedures simulated in LAPD are identical to the procedures given in HDLC ABM.

In Bux's experiments the following configurations are used:

- Line Capacity: 48K bits/sec
- Propagation Delay(Tp): 50m sec
- Packet Length: 5000 bits
- Window size: 8
- Timeout Period: (3*Ti + 2*Tp) sec
- BER: Error Free and 1e-5
- Overhead: 48 bits

Figures 5.3 and 5.4 shows the comparison between Bux's and ones obtained from LAPD simulation model. Figure 5.3 shows an error free comparison between LAPD and Bux's, it can be observed that simulation model closely simulates the LAPD protocol. Also figure 5.4 shows an experiment with the BER=1e-5. It is seen that protocol

performs as it expected which further validates the error recovery procedures of the LAPD simulation model. All simulated results given in figures 5.3 and 5.4 have 95% confidence intervals.

This chapter described the LAPD Protocol and Simulation Model. The program listing and flow charts are given in Appendix 1. The validation of the model presented, and it was seen that the model closely simulates the LAPD Protocol. The LAPD simulation model can be used to do performance studies of Data Link Protocol, and performance/prediction of computer networks specially Local Area Networks. As it explained before LAPD is very compatible to LLC of IEEE/Std 802.3 which in turn can be used to model Local Area Networks.

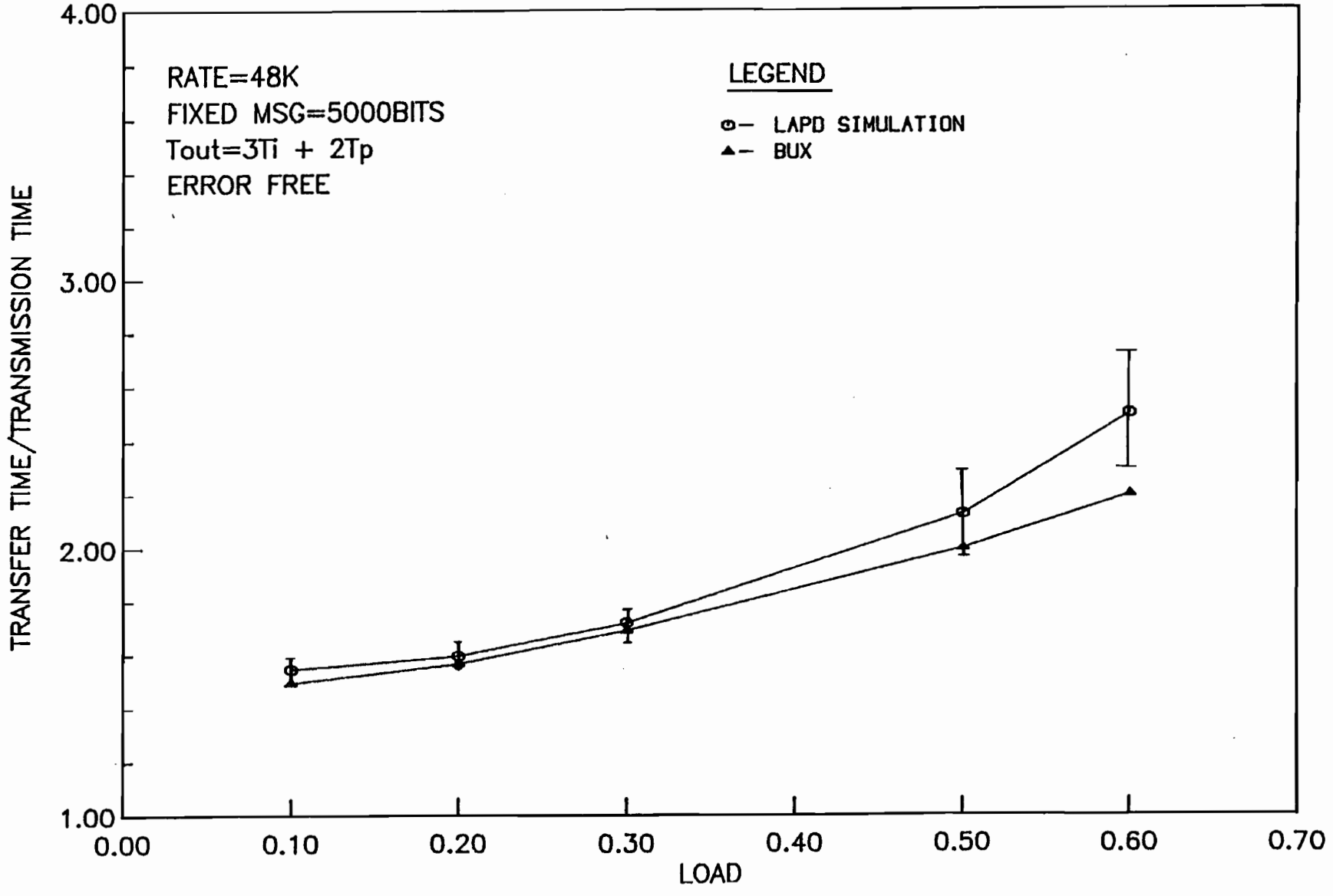


Figure 5.3 LAPD Validation – I

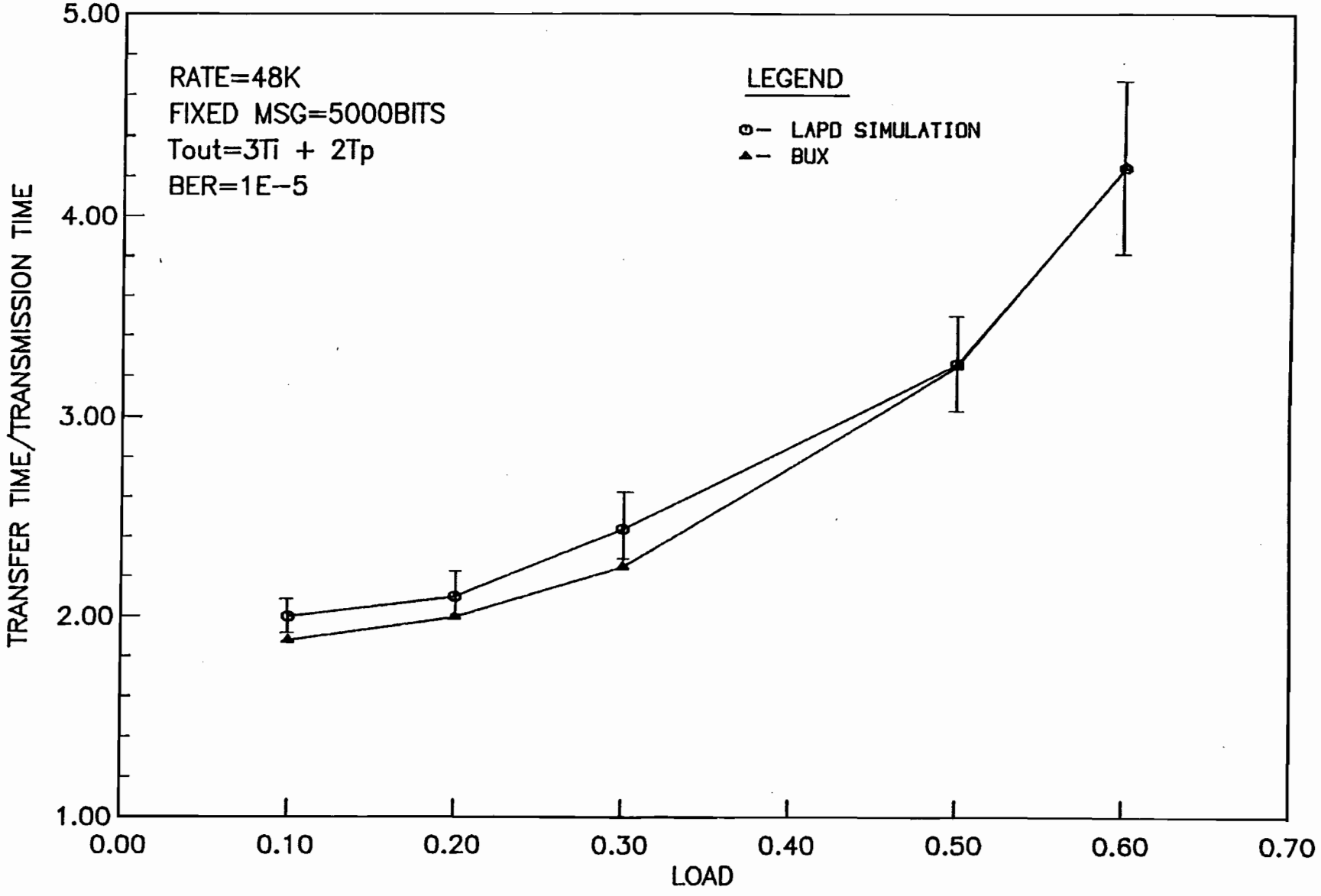


Figure 5.4 LAPD Validation – II

REFERENCES:

1. AT&T Information Systems,' Digital Multiplexed Interface Technical Specification',AT&T Information Systems, April 1985,pp IV-176,IV-259

2. E.M.Friedman,' Computer Simulation Model for a CSMA/CD Local Area Network Utilizing a Multirate Voice Coding Technique for Load Control',Technical Report,TISL - 6731-2,May 1985

3. B.Wood,' A Performance Evaluation of Local Area Networks',Technical Report,TISL - 6731-3,January 1986

4. W.Bux,K.Kummerle,and H.L.Truong,' Data Link-Control Performance: Results Comparing HDLC Operational Modes',Computer Networks, 6-1982,pp 37-51

5. W.Bux and H.L.Truong,' A Queuing Model for HDLC-Controlled Data Links', Flow Control in Computer Networks,IFIP,North-Holland Publishing Company,1979, pp 287,306

6. W.Bux,K.Kummerle,and H.L.Truong,' Balanced HDLC Procedures: A Performance Analysis',IEEE Trans. on Comm,Vol 28,No. 11,1980,pp 1889-1898

7. P.T.Brady,'Performance Of LAPD Protocol With Link Errors and Propagation Delay',ICC 1985,pp 121-125

8. A.J.Weissberger,'Bit Oriented Data Link Controls',Computer Design,March 1983 pp 135-141

9. E.Gelenbe, J.Labetoulle, and G.Pujolle,' Performance Evaluation of the HDLC Protocol',Computer Networks,2-1978,pp 409-415

10. A.U.Shankar and S.Lam ,'An HDLC Protocol Specification and its Verification Using Image Protocols',ACM Trans. on Computer Systems,Vol. 1, No.4,Nov. 1983,pp 331-368

11. W.Stallings,' Data and Computer Communications",Macmillan publishing company, New York 1985,pp 142-152.

12. A.B.Pritsker,' Introduction to Simulation and SLAM II ',Halsted Press Inc., 1984

6.0 The Integrated LAN Model

Chapters 4 and 5 described the main protocols needed to model a local area network, also chapter 3 discussed some basic concepts about local area networks. In order to make an accurate performance analysis of LANs, we must consider all possible aspects of the network. The first step would be to create a model that fully considers the major protocols, decision latencies, and processing times involved . As it was described in chapter 3, a local area network protocols consists of 3 layers: Physical layer, Media Access Control(MAC) layer, and Logical Link Control(LLC) layer. This chapter describes the integration of Link Access Protocol(LAP-D) described in chapter 5, as the IEEE 802.2 LLC and Medium Access Protocol(CSMA/CD)[1], as the IEEE 802.3 MAC into a single combined simulation model of a LAN. This chapter is divided into 4 sections:the simulation model,validation, and, performance measurements.

6.1 The Simulation Model

The developed integrated LAN simulation model is designed to predict the performance of local area networks. This model integrates the Medium Access Control Protocol (CSMA/CD) and Logical Link Control Protocol (LAP-D) into a single combined simulation model. Also the model allows for the specification of the processing times for each protocol function in the MAC and LLC protocol. The developed model is fully validated and accurately

simulates a local area network.

The integrated LAN model permits simulation of two or more detailed workstations. The workstations are primarily used either to generate a network background load for transport stations , or to be used fully as the transport stations. The simulated model can support as many as 99 workstations.

To develop an integrated model, the driver files of SLAM[11] Network Model of CSMA/CD simulation model of[1], and LAPD simulation model(chapter 5) are combined and modified to serve as a local area network. Figure 6.1 shows the driver file of integrated model SLAM[11] driver file. In addition the Discrete Event part of the model are combined to complete the integration of MAC and LLC layers. The developed model is highly modular in which each protocol function of MAC and LLC is implemented independently which gives the analyst the flexibility to modify any part of the protocol. This gives us the necessary tool to make further performance studies of local area networks. It is observed that IEEE 802 Standard specifies one Logical Link Control Standard, but three Media Access Control Standards. The integrated model has the flexibility that one MAC(CSMA/CD) can be replaced with the other MAC(e.g.Token Ring). In order to obtain the highest modeling efficiency of integrating other MACs to LLC(LAP-D), it is recommended to use SLAM[11] Discrete Event Modeling approach to define the MAC protocol. Figure 6.2 shows a flow chart of integrated CSMA/CD and LAPD protocols. It is observed from the flow

chart that MAC and LLC are distinctly separated and defined. Also it can be noted that integrated model incorporates the processing times involved in MAC and LAP protocols.

```
GEN,U OF K TISL,LAP MAP INTEGRATION MODEL,4/5/86,7,YES,YES;
LIMITS,47,42,1000;
;
;-------------------------------------------------------------------
;
;     XX(1) - XX(22) ARE THE AVERAGE MESSAGE LENGTHS FOR NODES 1 - 22
;
INTLC,XX(1)=4096.00,
      XX(2)=4096.00,
      XX(3)=4096.00,
      XX(4)=4096.00,
      XX(5)=4096.00,
      XX(6)=4096.00,
      XX(7)=4096.00;
INTLC,XX(8)=4096.00,
      XX(9)=4096.00,
      XX(10)=4096.00,
      XX(11)=4096.00,
      XX(12)=4096.00,
      XX(13)=4096.00,
      XX(14)=4096.00,
      XX(15)=4096.00;
INTLC,XX(16)=4096.00,
      XX(17)=4096.00,
      XX(18)=4096.00,
      XX(19)=4096.00,
      XX(20)=4096.00,
      XX(21)=4096.00,
      XX(22)=4096.00,
;-------------------------------------------------------------------
;
;    SET SIMULATIONS RUN AND TIMING PARAMETERS
;
INTLC,XX(25)=1.287E+7;   XX(25) = TOTAL LENGTH OF THE SIMULATION RUNS
;
INTLC,XX(26)= 22.0;     XX(26) = TOTAL NUMBER OF ACTIVE NODES(MAX=22)
;
INTLC,XX(27)=7.0;      XX(27) = NUMBER SIMULATION SUBRUNS
;
;-------------------------------------------------------------------
;
;     XX(50) - XX(71) ARE THE AVERAGE INTERARRIVAL TIMES (IN MICRO SEC)
;                     FOR NODES 1 - 22 RESPECTIVELY
;
INTLC,XX(50)=1.287E+5,
      XX(51)=1.287E+5,
      XX(52)=1.287E+5,
      XX(53)=1.287E+5,
      XX(54)=1.287E+5,
      XX(55)=1.287E+5;
INTLC,XX(56)=1.287E+5,
      XX(57)=1.287E+5,
```

```
53        XX(58)=1.287E+5,
```

Figure 6.1 Integrated LAN Model SLAM Driver File

```
54          XX(59)=1.287E+5,
55          XX(60)=1.287E+5,
56          XX(61)=1.287E+5,
57          XX(62)=1.287E+5,
58          XX(63)=1.287E+5,
59          XX(64)=1.287E+5;

60    INTLC,XX(65)=1.287E+5,
61          XX(66)=1.287E+5,
62          XX(67)=1.287E+5,
63          XX(68)=1.287E+5,
64          XX(69)=1.287E+5,
65          XX(70)=1.287E+5,
66          XX(71)=1.287E+5,
67          XX(72)=1.287E+5;
68    ;-----------------------------------------------------------------
69    ;    XX(75) - XX(88) SPECIFY THE NETWORK PARAMETERS AND TOPOLGY
70    ;
71    ;    IT IS ASSUMED HERE THAT THE PORT ON THE BUS ARE EQUAL SPACED
72    ;    AND THAT THE NODE-TO-BUS LINKS ARE OF EQUAL LENGTH
73    ;    ( THE INTLC.F ROUTINE CAN BE MODIFIED TO MODEL GENERAL TOPOLGIES )
74    ;
75    INTLC,XX(75)=1.0;       XX(75) = LINE RATE IN MBPS
76    ;
77    INTLC,XX(76)=16.0;      XX(76) = MAXIMUM NUMBER OF COLLISIONS
78    ;
79    INTLC,XX(77)=10.0;      XX(77) = BACKOFFLIMIT
80    ;
81    INTLC,XX(78)=96.0;      XX(78) = INTERFRAME SPACING IN MICRO SEC
82    ;
83    INTLC,XX(79)=32.0;      XX(79) = JAM TIME IN MICRO SEC
84    ;
85    INTLC,XX(80)=512.0 ;    XX(80) = SLOT TIME IN MICRO SEC
86    ;
87    INTLC,XX(81)=12144.0;   XX(81) = MAXIMUM PACKET SIZE IN BITS
88    ;
89    INTLC,XX(82)=512.0;     XX(82) = MINIMUM PACKET SIZE IN BITS
90    ;
91    INTLC,XX(83)=256.0;     XX(83) = NUMBER OF OVERHEAD BITS/PACKET
92    ;
93    INTLC,XX(84)=0.0033;    XX(84) = END-TO-END BUS DELAY IN MICRO SEC
94    ;
95    INTLC,XX(85)=0.8333;    XX(85) = NODE-TO-BUS ONE WAY DELAY IN MICRO SEC
96    ;
97    INTLC,XX(86)=8.0;       XX(86) = HOST ECHO PACKET SIZE IN BITS
98    ;
99    INTLC,XX(87)=0.888;     XX(87) = PROBABILITY THAT HOST WILL TX ECHO
100   ;
101   INTLC,XX(88)=22.0;      XX(88) = NUMBER OF TERMINAL NODES ON THE NETWORK
102   ;
103   INTLC,XX(89)= 0.0;      NOT USED
104   ;
```

```
105  INTLC,XX(90)= 0.0;      NOT USED
106  ;
107  INTLC,XX(110)=8.0;      XX(110)= MAX   OF OUTSTANDING FRAMES
108  ;
109  INTLC,XX(111)=1.0E+6;   XX(111)= TIME OUT PERIOD IN MICRO SEC
110  ;
111  INTLC,XX(112)=0.0;      XX(112)=PROBABILITY OF A PACKET BEING IN ERROR
112  ;
113  INTLC,XX(113)=1.0E-6;   XX(113)= PROBABILITY OF A BIT BEING IN ERROR
114  ;
115  INTLC,XX(114)=5.3E+11; XX(114)= IF NEEDED USED FOR TRACING AND DEBUGGING
116  ;                               PURPOSES
117  ;
118  INTLC,XX(115)=0.0;      XX(115)= VAR TO COLLECT STAT. ON AVE.   IN THE SYSTEM
119  ;
120  INTLC,XX(116)=0.00E+3; NOT USED
121  ;
122  INTLC,XX(117)=0.20E+3; XX(117) =  ACKPKTS PROCESSING TIME IN MICRO SEC
123  ;
124  INTLC,XX(118)=1.85E+3; XX(118) =  BMOVE   PROCESSING TIME IN MICRO SEC
125  ;
126  INTLC,XX(119)=0.40E+3; XX(119) =  ENQUEUE PROCESSING TIME IN MICRO SEC
127  ;
128  INTLC,XX(120)=0.55E+3; XX(120) =  DEQUEUE PROCESSING TIME IN MICRO SEC
129  ;
130  INTLC,XX(121)=0.10E+3; XX(121) =  GT COMM PROCESSING TIME IN MICRO SEC
131  ;
132  INTLC,XX(122)=0.10E+3; XX(122) =  GV RESP PROCESSING TIME IN MICRO SEC
133  ;
134  INTLC,XX(123)=0.30E+3; XX(123) =  INF PKT PROCESSING TIME IN MICRO SEC
135  ;
136  INTLC,XX(124)=0.10E+3; XX(124) =  RCV MSG PROCESSING TIME IN MICRO SEC
137  ;
138  INTLC,XX(125)=0.15E+3; XX(125) =  RECEIVE PROCESSING TIME IN MICRO SEC
139  ;
140  INTLC,XX(126)=0.35E+3; XX(126) =  SEND    PROCESSING TIME IN MICRO SEC
141  ;
142  INTLC,XX(127)=0.10E+3; XX(127) =  SND SPR PROCESSING TIME IN MICRO SEC
143  ;
144  INTLC,XX(128)=0.10E+3; XX(128) =  SND MSG PROCESSING TIME IN MICRO SEC
145  ;
146  INTLC,XX(129)=0.0;      XX(129) =  NOT USED
147  ;
148  INTLC,XX(130)=0.0;      XX(130) =  NOT USED
149  ;
150  INTLC,XX(131)=0.10E+3; XX(131) =  SENSING PROCESSING TIME IN MICRO SEC
151  ;
152  INTLC,XX(132)=0.40E+3; XX(132) =  COLLSN  PROCESSING TIME IN MICRO SEC
153  ;
154  INTLC,XX(133)=0.00E+3; XX(133) =  DEFFRNG PROCESSING TIME IN MICRO SEC
155  ;
156  INTLC,XX(134)=0.20E+3; XX(134) =  CRC CHK PROCESSING TIME IN MICRO SEC
```

```
157  ;
158  ;---------------------------------------------------------------------
159  ;
160  TIMST,NNQ(1),AVE NUM IN QUE1;
161  TIMST,NNQ(2),AVE NUM IN QUE2;
162  TIMST,XX(115),AVE NUM IN SYSTM;
163  STAT,1,    ACCESS DELAY;
164  STAT,2,   CHANNEL DELAY;
165  STAT,3, SR TO MAP Q DLY;
166  STAT,4,   AVE MAP DELAY;
167  STAT,5,USR TO LAP Q DLY;
168  STAT,6,  AVE. LAP DELAY;
169  STAT,7,USR1  USR3   DLY;
170  STAT,8,USR2  USR4   DLY;
171  STAT,9,USR3  USR1   DLY;
172  STAT,10,USR4  USR2   DLY;
173  STAT,11,USR5  USR6   DLY;
174  STAT,12,USR6  USR5   DLY;
175  STAT,13,USR7  USR10  DLY;
176  STAT,14,USR8  USR12  DLY;
177  STAT,15,USR9  USR11  DLY;
178  STAT,16,USR10 USR7   DLY;
179  STAT,17,USR11 USR9   DLY;
180  STAT,18,USR12 USR8   DLY;
181  STAT,19,USR13 USR22  DLY;
182  STAT,20,USR14 USR15  DLY;
183  STAT,21,USR15 USR14  DLY;
184  STAT,22,USR16 USR20  DLY;
185  STAT,23,USR17 USR18  DLY;
186  STAT,24,USR18 USR17  DLY;
187  STAT,25,USR19 USR21  DLY;
188  STAT,26,USR20 USR16  DLY;
189  STAT,27,USR21 USR19  DLY;
190  STAT,28,USR22 USR13  DLY;
191  STAT,29,AVE  USR MSG DLY;
192  STAT,30,AVE.BFR HLDNG TM;
193  STAT,31,AVE.MESSAGE SIZE;
194  ;
195  NETWORK;
196  ;
197  ;     VARIABLE DEFINITIONS:
198  ;
199  ;       ATRIB(1) = PACKET CREATION TIME
200  ;       ATRIB(2) = NODE IDENTIFICATION
201  ;       ATRIB(3) = PROPAGATION LEFT MARKER, IF LEFT PROP IS FINISHED THEN
202  ;                  ATRIB(3)=0
203  ;       ATRIB(4) = PROPAGATION RIGHT MARKER, IF RIGHT PROP IS FINISHED THEN
204  ;                  ATRIB(4)=0
205  ;       ATRIB(5) = IF ATRIB(5)=0 THEN EVENT BEING SCHEDULED IS PROPAGATION,
206  ;                  IF ATRIB(5)=1 THEN EVENT BEING SCHEDULED IS TRANSMISSION
207  ;       ATRIB(6) = COUNTER FOR THE NUMBER OF ATTEMPTS TO TRANSMIT
208  ;       ATRIB(7) = END OF PROPAGATION LEFT MARKER, IF ATRIB(7)=0 THEN
```

```
209  ;                    FINISHED PROP LEFT
210  ;          ATRIB(8) = END OF PROPAGATION RIGHT MARKER, IF ATRIB(8)=0 THEN
211  ;                    FINISHED PROP RIGHT
212  ;          ATRIB(9) = PACKET LENGTH IN MICROSECONDS
213  ;          ATRIB(10)= AMOUNT OF TIME LEFT AFTER THE COLLISION
214  ;                    DISCRIMINATION PERIOD
215  ;          ATRIB(11)= NUMBER OF BITS PER PACKET
216  ;          ATRIB(12)= MARKS THE TIME TRANSMISSION IS ATTEMPTED, GETS RESET EACH
217  ;                    TIME A RETRANSMISSION IS ATTEMPTED
218  ;          ATRIB(13)= WHILE PACKET IS IN THE AWAIT NODE ATRIB(13)=0, SET TO 1
219  ;                    WHEN PACKET LEAVES THE QUEUE, USED TO COLLECT STATISTICS
220  ;                    ON THE AMOUNT OF TIME A PACKET IS QUEUED, TQUEUD= ATRIB(14
221  ;                    MINUS ATRIB(1)IF ATRIB(13)=0
222  ;          ATRIB(14)= MARKS THE TIME WHEN A PACKET LEAVES THE NODE QUEUE
223  ;                    (THE AWAIT), USED TO COLLECT STATISTICS ON THE TIME
224  ;                    REQUIRED TO ACCESS THE NETWORK, TACCSS=ATRIB(12)-ATRIB(14)
225  ;          ATRIB(15)= IDENTITY OF TRANSMITTING PORT
226  ;          ATRIB(16)= INDICATES OF LOGIC FUNCTIONS ARE FROM PROPAGATED EVENTS
227  ;          ATRIB(17)= FIRST ACCESS ATTRIBUTE, IF THE PACKET IS TRYING TO TRANSMI
228  ;                    FOR THE FIRST TIME THEN ATRIB(17) IS SET TO 1 FROM 0, ON
229  ;                    SUBSEQUENT ATTEMPTS TO TRANSMIT ATRIB(17) IS INCREMENTED
230  ;          ATRIB(18) - ATRIB(24)   NOT USED
231  ;          ATRIB(25)= DESTINATION ADDRESS OF A PACKET
232  ;          ATRIB(26)= FILE   OF THE REPEAT QUEUE
233  ;          ATRIB(27)= PACKET COUNT NUMBER
234  ;          ATRIB(28)= PACKET SEND SEQUENCE NUMBER ( N(s)   PROTOCOL VERSION )
235  ;          ATRIB(29)= PACKET TIMEOUT PERIOD
236  ;          ATRIB(30)= THE COMMUNICATION INDICATOR BETWEEN DATA LINK LAYER
237  ;                    AND APPLICATION LAYER IS DEFINED AS FOLLOWS:
238  ;                    ATRIB(30) = 1.0 INDICATES THAT LAP IS READY TO ACCEPT
239  ;                    I - FRAMES FROM APPLICATION LAYER
240  ;                    ATRIB(30) = 2.0 INDICATES THAT LAP IS SENDING AN I - FRAME
241  ;                    TO THE APPLICATION LAYER
242  ;          ATRIB(31)= SIGNIFIES IF PACKET IS CORRECT,IN ERROR,OR OUT OF SEQUENCE
243  ;                    ATRIB(31) = 0.0 CORRECT PACKET
244  ;                    ATRIB(31) = 1.0 PACKET IN ERROR (TRANSMISSION ERROR)
245  ;                    ATRIB(31) = 2.0 OUT OF SEQUENCE PACKET
246  ;          ATRIB(32)= USED FOR TIMER RESETTING PURPOSES AND IS DEFINED AS
247  ;                    FOLLOWS:
248  ;                    ATRIB(32) = 0.0 STOP TIMER FOR ALL ACKNOWLEDGED FRAMES
249  ;                    ATRIB(32) = 1.0 STOP TIMER ONLY FOR ONE PARTICULAR I - FRA
250  ;                    THIS IS THE CASE WHEN NODE IS IN TIMER RECOVERY CONDITION
251  ;                    ATRIB(32) = 2.0 STOP TIMER FOR ALL TRANSMITTED FRAMES . IT
252  ;                    IS USED WHEN NODE IS IN REJ RECOVERY CONDITION
253  ;          ATRIB(33)= INDICATES THE TYPE OF A FRAME (COMMAND/RESPONSE)
254  ;                    ATRIB(33) = 1.0   COMMAND FRAME
255  ;                    ATRIB(33) = 2.0   RESPONSE FRAME
256  ;          ATRIB(34)= MEASURES LAP DELAY
257  ;          ATRIB(35)= MEASURES MAP DELAY
258  ;          ATRIB(36)= INDICATES IF A NODE IS A
259  ;                    TRANSPORT STATION/BACKGROUND STATION:
260  ;                    ATRIB(36) = 0.0    TRANSPORT STATION
```

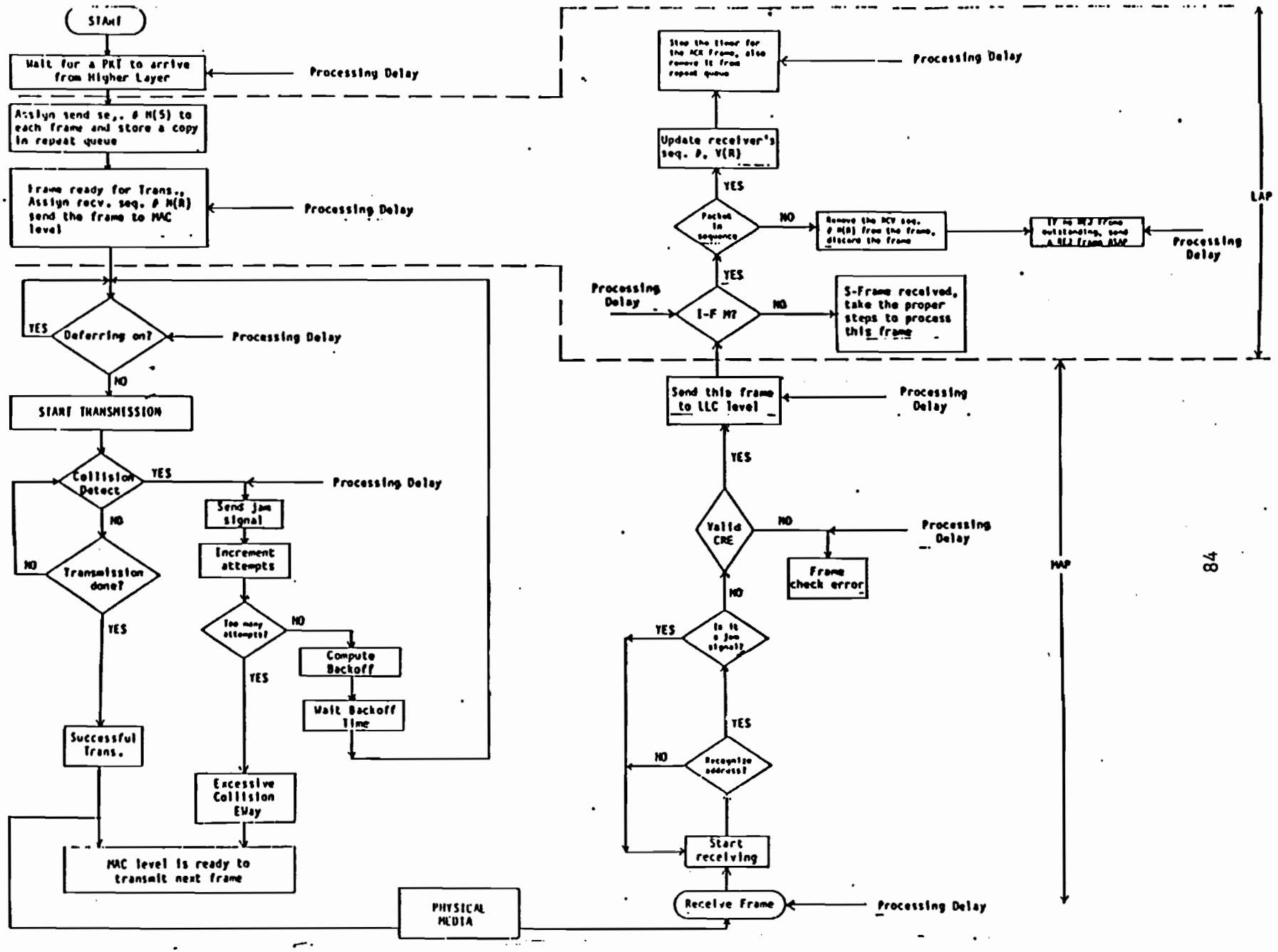
```
;                   ATRIB(36) = 1.0    BACKGROUND STATION
;       ATRIB(37)= NOT USED
;       ATRIB(38)= MARKS THE PACKET COUNT NUMBER THAT A RECEIVER HAS RECEIVED
;                  PLUS ONE
;       ATRIB(39)= THE PACKET SEQUENCE NUMBER THAT RECEIVER EXPECTS TO RECEIV
;                  ( V(r) - PROTOCOL VERSION )
;       ATRIB(40)= INDICATES THE PACKET I.D. AND IS DEFINED AS FOLLOWS:
;                  ATRIB(40) = 0.0 I - FRAME
;                  ATRIB(40) = 1.0 S - FRAME ( RR FRAME USED FOR
;                                ACKNOWLEDGEMENT)
;                  ATRIB(40) = 2.0 S - FRAME ( REJ FRAME )
;                  ATRIB(40) = 3.0 S - FRAME ( RR FRAME USED FOR TIMER
;                                RECOVERY CONDITION )
;       ATRIB(41)= INDICATES THAT THE DATA LINK LAYER ENTITY TRANSMITTING
;                  THE ATRIB(41) HAS RECEIVED ALL I - FRAMES NUMBERED UP
;                  TO AND INCLUDING ATRIB(41) - 1 ( N(r) - 1 PROTOCOL
;                  VERSION )
;       ATRIB(42)= TIMEOUT EVENT INDICATOR
;                  ATRIB(42)= 0.0 NON TIMEOUT EVENT
;                  ATRIB(42)= 1.0 TIMEOUT EVENT
;
;     XX(25) = THE THE TOTAL SIMULATION TIME OVER ALL RUNS USED IN CALC
;              OUTPUT STATISTICS
;
;       FILES   1-22 RESERVED FOR THE AWAIT NODE AT EACH STATION
;       FILES  23-44 RESERVED FOR THE REPEATE QUEUE AT EACH STATION
;       FILE   45    CHANNEL QUEUE NODE
;       FILE   46    SUPERVISORY FRAMES WAITING FOR TRANSMISSION
;       FILE   47    DEFER FILE
;       FILE   48    EVENT CALENDAR
;
        RESOURCE/NODE1,1/NODE2,2/NODE3,3/NODE4,4/NODE5,5/NODE6,6/
        NODE7,7/NODE8,8/NODE9,9/NODE10,10/NODE11,11/NODE12,12/NODE13,13/
        NODE14,14/NODE15,15/NODE16,16/NODE17,17/NODE18,18/NODE19,19/
        NODE20,20/NODE21,21/NODE22,22;
;
;
;
;
;-------------------------------------------------------------------
;
;
;
;-------------------------------------------------------------------
;     MODEL OF THE NUMBER OF STATIONS ON THE ETHER
;     ===========================================
;
;-------------------------------------------------------------------
;
      CREATE,EXPON(XX(50),1),,1;CREATE PACKETS AT NODE 1
      ASSIGN,ATRIB(2)=1.0,
             ATRIB(25)=3.0,
```

```
313               ATRIB(26)=ATRIB(2) + XX(26),
314               ATRIB(11)=USERF(2),
315               ATRIB(9)=ATRIB(11)/XX(75),
316               ATRIB(36)=0.0,
317               ATRIB(29)=XX(111);
318         ASSIGN,XX(115)=XX(115)+1;
319         AWAIT(1),ALLOC(1);
320         ACT,,,CHNL;                    IMMEDIATE BRANCH TO THE CHANNEL
321   ;
322   ;-------------------------------------------------------------------
323   ;
324         CREATE,EXPON(XX(51),5),1000.5,1;
325         ASSIGN,ATRIB(2)=2.0,
326               ATRIB(25)=4.0,
327               ATRIB(26)=ATRIB(2) + XX(26),
328               ATRIB(11)=USERF(2),
329               ATRIB(9)=ATRIB(11)/XX(75),
330               ATRIB(36)=1.0,
331               ATRIB(29)=XX(111);
332         ASSIGN,XX(115)=XX(115)+1;
333         AWAIT(2),ALLOC(2);
334         ACT,,,CHNL;
335   ;
336   ;-------------------------------------------------------------------
337   ;
338          CREATE,EXPON(XX(52),4),2000.2,1;
339          ASSIGN,ATRIB(2)=3.0,
340                 ATRIB(25)=1.0,
341                 ATRIB(26)=ATRIB(2) + XX(26),
342                 ATRIB(11)=USERF(2),
343                 ATRIB(9)=ATRIB(11)/XX(75),
344                 ATRIB(36)=0.0,
345                 ATRIB(29)=XX(111);
346          ASSIGN,XX(115)=XX(115)+1;
347          AWAIT(3),ALLOC(3);
348          ACT,,,CHNL;
349   ;
350   ;-------------------------------------------------------------------
351   ;
352          CREATE,EXPON(XX(53),3),3000.4,1;
353          ASSIGN,ATRIB(2)=4.0,
354                 ATRIB(25)= 2.0,
355                 ATRIB(26)=ATRIB(2) + XX(26),
356                 ATRIB(11)=USERF(2),
357                 ATRIB(9)=ATRIB(11)/XX(75),
358                 ATRIB(36)=1.0,
359                 ATRIB(29)=XX(111);
360          ASSIGN,XX(115)=XX(115) + 1;
361          AWAIT(4),ALLOC(4);
362          ACT,,,CHNL;
363   ;
```

```
                .
                .
                .
                .
                .
                .
                .
                .
                .
;
        CREATE,EXPON(XX(69),3),25000.5,1;
        ASSIGN,ATRIB(2)=20.0,
               ATRIB(25)= 16.0,
               ATRIB(26)=ATRIB(2) + XX(26),
               ATRIB(11)=USERF(2),
               ATRIB(9)=ATRIB(11)/XX(75),
               ATRIB(36)=1.0,
               ATRIB(29)=XX(111);
        ASSIGN,XX(115)=XX(115) + 1;
        AWAIT(20),ALLOC(20);
        ACT,,,CHNL;
;
;-------------------------------------------------------------------------
;
      CREATE,EXPON(XX(70),1),26000.5,1;CREATE PACKETS AT NODE 21
      ASSIGN,ATRIB(2)=21.0,
             ATRIB(25)=19.0,
             ATRIB(26)=ATRIB(2) + XX(26),
             ATRIB(11)=USERF(2),
             ATRIB(9)=ATRIB(11)/XX(75),
             ATRIB(36)=1.0,
             ATRIB(29)=XX(111);
      ASSIGN,XX(115)=XX(115)+1;
      AWAIT(21),ALLOC(21);
      ACT,,,CHNL;                        IMMEDIATE BRANCH TO THE CHANNEL
;
;-----------------------------------------------------------------------
;
      CREATE,EXPON(XX(71),5),40000.5,1;
      ASSIGN,ATRIB(2)=22.0,
             ATRIB(25)=13.0,
             ATRIB(26)=ATRIB(2) + XX(26),
             ATRIB(11)=USERF(2),
             ATRIB(9)=ATRIB(11)/XX(75),
             ATRIB(36)=1.0,
             ATRIB(29)=XX(111);
      ASSIGN,XX(115)=XX(115)+1;
      AWAIT(22),ALLOC(22);
      ACT,,,CHNL;
;
;-----------------------------------------------------------------------
;
```

```
618  ;      COMBINE STATIONS TO FORM CHANNEL
619  ;      ================================
620  ;
621  CHNL   QUEUE(45);                      DUMP ENTITIES INTO QUEUE
622  ;
623         ACT;                            IMMEDIATE BRANCH TO CHANNEL MODEL
624  ;
625         GOON,1;
626  ;
627         ACT,,ATRIB(36).LE.0,LAP;
628         ACT,,ATRIB(36).GT.0,MAP;
629  ;
630  LAP    EVENT,11;
631         TERM;
632  ;
633  MAP    EVENT,1;
634         TERM;
635  ;
636  ;
637         END;
638  ;
639  INIT,0.0,1.287E+7;
640  MONTR,CLEAR,2.574E+6;
641  ;
642  SIMULATE;
               .
               .
               .
               .
               .
670  FIN;
```

Figure 6.2 Integrated LAN Model Flow Diagram



6.1.1 Integrated LAN Model Processing Times

Chapter 3 described the importance of implementing processing times in local area network simulation. The model developed here allows for the explicit specification of the processing times for each protocol function. These processing times are measured in microseconds and are defined in the next two sections. The complete description of these processing times are obtained from [2,3]. The processing times used in the integrated model are defined in terms of SLAM[11] global variables described in section 6.1.2.2.

The processing times used for the integrated LAP-D/CSMA-CD model are taken from [2] (LAP processing times only). The processing times measurement performed in [2] are on a VAX 11/780 processor under UNIX. These processing times are shown in SLAM[11] driver file in figure 6.1. The choice to use the VAX 11/780 processing times might not be a good approximations for processing times in popular and common LANs. The reason for such dilemma is because there is a lack of information about actual processing times in popular LANs. It should be noted that by applying more appropriate LAN processing times to the LAP-D/CSMA-CD integrated LAN model, more accurate results can be obtained. The software for the integrated LAP-D/CSMA-CD model with the associated flow charts is given in Appendix 2 of this report.

6.1.1.1 Logical Link Control Processing Times

The Logical Link Control processing functions are currently implemented in software in which processing times depends heavily on the speed of packet processing at the network interface[2]. The LLC processing times are defined as follows:

SND COMM = time to process commands sent to the lower layer entity. Among the commands processed are commands to send data on the network and message packetization.

RCV COMM = time to process commands received from a lower layer entity. Among the commands processed are commands to receive data from the network and message depacketization.

BMOVE = a function for block movement of data within the memory ,packetizing ,and depacketizing.

TRANSMITTER

SEND = time to decide what type of frame is waiting for transmission or to be transmitted.

SNDSPRVS = time to prepare a supervisory frame

INFO PKT = time to prepare an I-frame for transmission : assign a frame sequence number and piggyback the acknowledgment.

SND MSG = time to transfer an I-frame or a S-frame to the media Access Layer.

RECEIVER

RCV MSG = time needed to transfer an I-frame or a S-frame from the Media Access Layer to the Logical Link Control Layer

RECEIVE = time to process each frame received from the network according to its type.

ACK PKT = time to process the acknowledgment received either in terms of piggybacked acknowledgement, RR frame, or REJ frame and remove the acknowledged packet from the repeat buffer.

INFO PKT = time to process the received information frame : check for the correct sequence number.

Enqueue/Dequeue = They arre queue management functions which are in terms of primitives to access queues

It should be noted that the integrated LAN model is sensitive to timing parameters, and since LLC processing times can be chosen to be very long(in order of seconds), it is responsibility of the experimenter to adjust the necessary timing parameters in the model to accommodate the long processing times,i.e.timeout period,end user queue length,etc.

6.1.1.2 The Medium Access Control Processing Times

The Media Access Control Protocol normally implemented in hardware which requires modest amount of processing times and its implementation is as fast as medium[2]. For instance, the INTEL 82588 LAN Controller designed to support one of the most common Media Access Controls CSMA/CD which is used for PC Networks of AT&T's StarLAN and IBM. The MAC processing times can be listed as follows:

Sensing = Hardware processing time for sensing the channel

Collision = Processing time to calculate the backoff time and to send a jam signal

Deferring = Processing time needed to defer a frame

CRC Check = Processing time to calculate the CRC value

6.1.2 Variables and File Assignments

The integrated LAN model combines the variables and files of CSMA/CD defined in[1,4] and LAPD described in chapter 5. This model follows the same filing structure as given in CSMA/CD and LAPD. In addition to variables described for LAP and MAC, there several variables defined in the integrated LAN simulation model. These variables are classified as SLAM[11], user, and measurement variables.

The SLAM[11] variables in the integrated model includes attributes and global variables. The definition of SLAM[11] variables in the model are given below in the following fashion:

"-" indicates MAC variables

"+" indicates LLC variables

"*" indicates a variable that is specific to the LAN integrated model

6.1.2.1 SLAM Attribute Variable Type

The attribute variables defined in the integrated simulation model are as follows:

- atrib(1) = packet creation time
- atrib(2) = node identification
- atrib(3) = propagation left marker, if left prop is finished then atrib(3)=0
- atrib(4) = propagation right marker, if right prop is finished then atrib(4)=0
- atrib(5) = if atrib(5)=0 then event being scheduled is propagation, if atrib(5)=1 then event being scheduled is transmission

- atrib(6) = counter for the number of attempts to transmit
- atrib(7) = end of propagation left marker, if atrib(7)=0 then finished prop left
- atrib(8) = end of propagation right markar, if atrib(8)=0 then finished prop right
- atrib(9) = packet length in microseconds
- atrib(10) = amount of time left after the collision discrimination period
- atrib(11) = number of bits per packet
- atrib(12) = marks the time transmission is attempted, gets reset each time a retransmission is attempted
- atrib(13) = while packet is in the wait node atrib(13)=0, set to 1 when packet leaves the queue, used to collect statistics on the amount of time a packet is queued, tqueued=atrib(14) minus atrib(1) if atrib(13)=0
- atrib(14) = marks the time when a packet leaves the node queue(the wait), used to collect statistics on the time required to access the network, taccss=atrib(12)-atrib(14)
- atrib(15) = identity of transmission port

- atrib(16) = indicates of logic functions are from propagated events

- atrib(17) = first access attribute, if the packet is trying to transmit for the first time then atrib(17) is set to 1 from 0, on subsequent attempts to transmit atrib(17) is incremented

- atrib(18)---->atrib(24) not used

+ atrib(25) = destination node

+ atrib(26) = file number of the repeat queue

+ atrib(27) = packet count number

+ atrib(28) = packet Send Sequence Number(N(s) Protocol Version)

+ atrib(29) = packet timeout period

+ atrib(30) = the communication indicator between Data Link Layer and Application Layer are defined as follows :

    atrib(30)=1.0 indicates that LLC is ready to accept I-frames from application layer

    atrib(30)=2.0 indicates that LLC is sending an I-frame to the application layer

+ atrib(32) = used for timer resetting purposes and is defined as follows:

  atrib(32)=0.0 stop timer for acknowledged I-frames

  atrib(32)=1.0 stop timer only for one particular I-frame. This is the case when node is in timer recovery condition.

  atrib(32)=2.0 stop timer for all transmitted I-frames. It is used when node is in REJ recovery condition.

+ atrib(31) = signifies if frame is correct, in error, or out of sequence

  atrib(31)=0.0 correct frame

  atrib(31)=1.0 frame in error(transmission error)

  atrib(31)=2.0 out of sequence frame

+ atrib(33) = indicates the type of frame(command/response)

  atrib(33)=1.0 command frame

  atrib(33)=2.0 response frame

+ atrib(34) = measures LLC delay

+ atrib(35) = measures MAC delay

* atrib(36) = indicates if a node is a transport station/background station

    atrib(36) = 0.0 - transport station

    atrib(36) = 1.0 - background station

+ atrib(37) = not used

+ atrib(38) = indicates the packet count number that receiver has received plus one

+ atrib(39) = the packet sequence number that receiver expects to receive (V(r) protocol version)

+ atrib(40) = indicates the frame I.D. and is defined as follows:

    atrib(40)=0.0 I-frame

    atrib(40)=1.0 S-frame (RR frame)

    atrib(40)=2.0 S-frame (REJ frame)

    atrib(40)=3.0 S-frame (RR frame for timer recovery)

+ atrib(41) = indicates that the Data Link Layer entity transmitting , has received all I-frames numbered up to including atrib(41) - 1 (N(r) - 1 protocol version)

+ atrib(42) = timeout event indicator

    atrib(42)=0.0 non timeout event

    atrib(42)=1.0 timeout event

6.1.2.2 SLAM Global Variables

SLAM[11] global variables used to define the CSMA/CD, LAPD, integrated, and modeling variables that are necessary to model a local area network. These variables are defined as follows:

"." modeling variables

"-" CSMA/CD variables

"+" LAPD variables

"*" integrated variables

. XX(1) - XX(24) = Average Message Length for nodes 1 - 24

. XX(25) = simulation run time in microseconds

. XX(26) = total number of active nodes (max = 24)

. XX(27) = number of simulation runs

. XX(28)---->XX(49) not used

. XX(50)- XX(74) = average interarrival times(in microseconds) for nodes 1 - 24 respectively

. XX(75) = Line rate in Mbps

- XX(76) = maximum number of collisions

- XX(77) = backoff limit

- XX(78) = interframe spacing in microseconds

- XX(79) = jam time in microseconds

- XX(80) = slot time in microseconds

- XX(81) = maximum packet size in bits

- XX(82) = minimum packet size in bits

- XX(83) = number of overhead bits/packet

- XX(84) = end-to-end bus delay in microseconds

- XX(85) = node-to-bus one way delay in microseconds

- XX(86)--->XX(109) not used

+ XX(110) = maximum number of outstanding frames(window size)

+ XX(111) = timeout period in microseconds

+ XX(112) = not used

+ XX(113) = probability of a bit being in error

+ XX(114) = if needed used for tracing and debugging purposes

+ XX(115) = variable to collect statistics on the average number of I-frames in the system

* XX(116) = not used

* XX(117) = ACKPKT processing time in microseconds

* XX(118) = BLOCK MOVE processing time in microseconds

* XX(119) = ENQUEUE processing time in microseconds

* XX(120) = DEQUEUE processing time in microseconds

* XX(121) = GT COMM processing time in microseconds

* XX(122) = GV RESP processing time in microseconds

```
*  XX(123) = INFO PKT processing time in microseconds
*  XX(124) = RCV MSG processing time in microseconds
*  XX(125) = RECEIVE processing time in microseconds
*  XX(126) = SEND provessing time in microseconds
*  XX(127) = SEND SPRVS processing time in microseconds
*  XX(128) = SND MSG processing time in microseconds
*  XX(129)--->XX(130) not used
*  XX(131) = SENSING processing time in microseconds
*  XX(132) = COLLSN processing time in microseconds
*  XX(133) = DEFFRNG processing time in microseconds
*  XX(134) = CRC CHK processing time in microseconds
```

6.2 Validation of The Integrated LAP-D/CSMA-CD LAN Model

Chapters 4 and 5 discussed the separate validation of CSMA/CD protocol and LAPD protocol. This section describes the validation of the Integrated LAP-D/CSMA-CD LAN Model.

Up to now, there has not been any analytical studies or simulations that considers the performance of integrated IEEE 802.3 (CSMA/CD) and IEEE 802.2 LLC (LAPD) in which we can compare the integrated model simulation results. Therefore the validation of

integrated LAN model is dependent on the fact that CSMA/CD and LAPD simulation models are both validated and simulate accurately each of these protocols. Because there are not any measurements or analytical models available to validate the integrated LAN model, two approaches were developed to confirm the performance of integrated LAN simulation model. These two approaches are now described.

6.2.1 LAP-D/CSMA-CD Validation Approach-I

The first validation approach uses the property of the CSMA/CD networks in which packet transfer time is almost equal to packet transmission time at the low load. This phenomenon is explained by the fact that number of collisions at the low load is minimal in CSMA/CD and this would have little effect on the delay versus throughput performance of the CSMA/CD networks. By knowing this fact , the integrated LAN model should behave like a LLC protocol is present at the low loads(less than 0.30). This approach can be used to validate our integrated LAP-D/CSMA-CD LAN model by comparing the simulated results to Bux's[5] of analytical results of HDLC ABM and LAPD validation results(chapter 5). The results given consider a 95% confidence interval. Figures 6.3 and 6.4 show the validation of integrated simulation model compared to results obtained by Bux[5] and LAPD simulation results for error free and BER=1e-5 cases. Tables 1 and 2 shows the simulated CSMA/CD and LAPD parameters respectively for comparing to Bux's results. It is observed that the integrated results are in close agreement with the results

obtained from LAPD and Bux's models. It should be noted that as the load increases the CSMA/CD protocol becomes more dominant in performance of LAP-D/CSMA-CD model. This explained by the fact that when load increases, the collisions in the channel increase which in turn leads to greater backoff times and finally greater packet transfer time. The effect of such delay clearly can be observed on LAP-D/CSMA-CD validation curves in figures 6.3 and 6.4.

Another factor that effected validation results is timeout period. The delay caused by MAC retransmissions due to collisions has a negative impact on the optimum timeout period which is commonly set equal to time of transmitting 3 I-frames. This clearly show that the timeout period period (3 * Ti) would be too short for the integrated LAP-D/CSMA-CD model, therefore timeouts would occur in normal I-frame transmission which causes a greater transfer time for the integrated LAP-D/CSMA-CD model. Therefore, the timeout period 3*Ti suggested by Bux[6] would not be valid for the integrated Logical Link Control and Media Access Control Protocols.

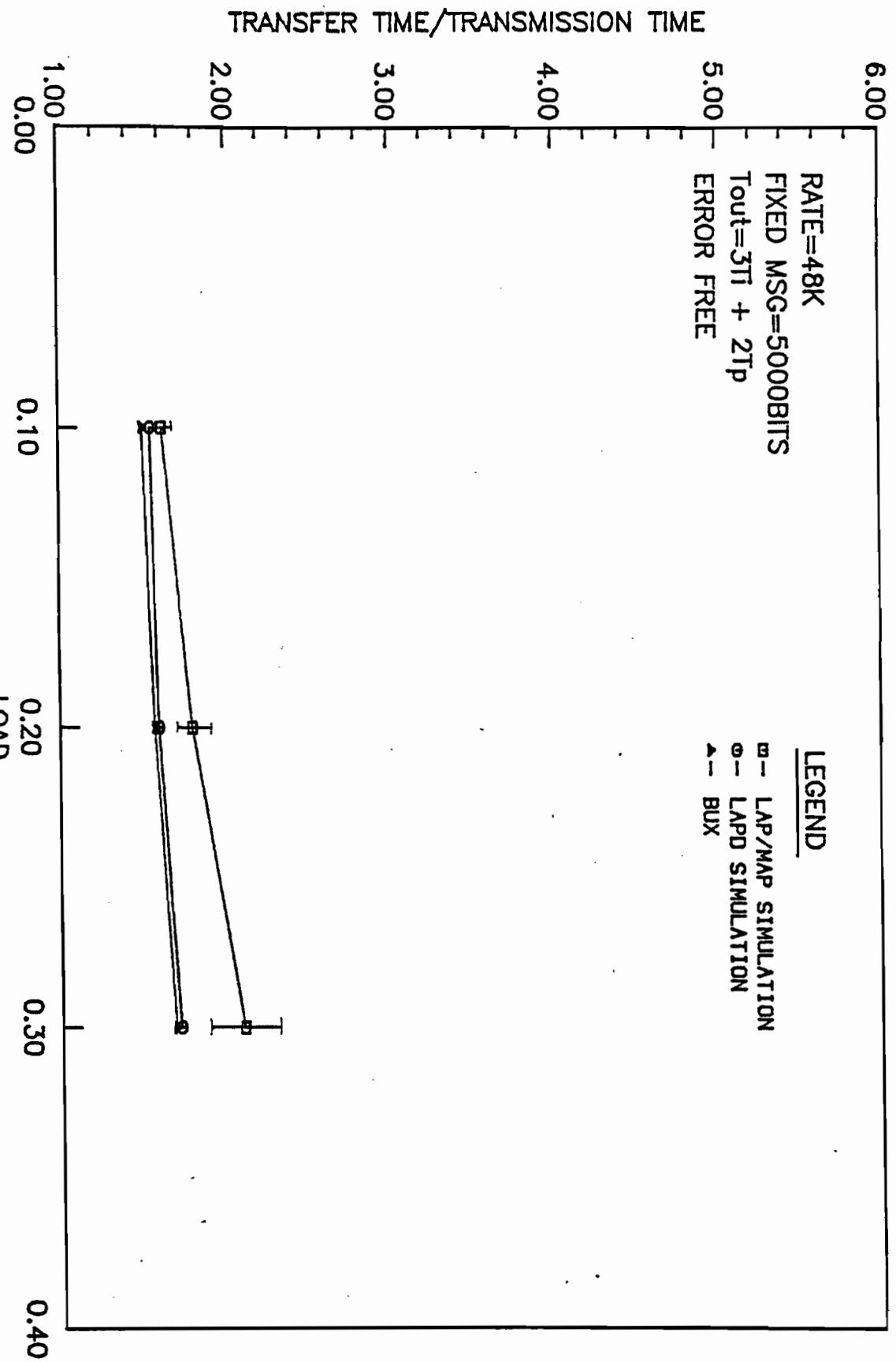


Figure 6.3 LAP-D/CSMA-CD LAN Validation - I

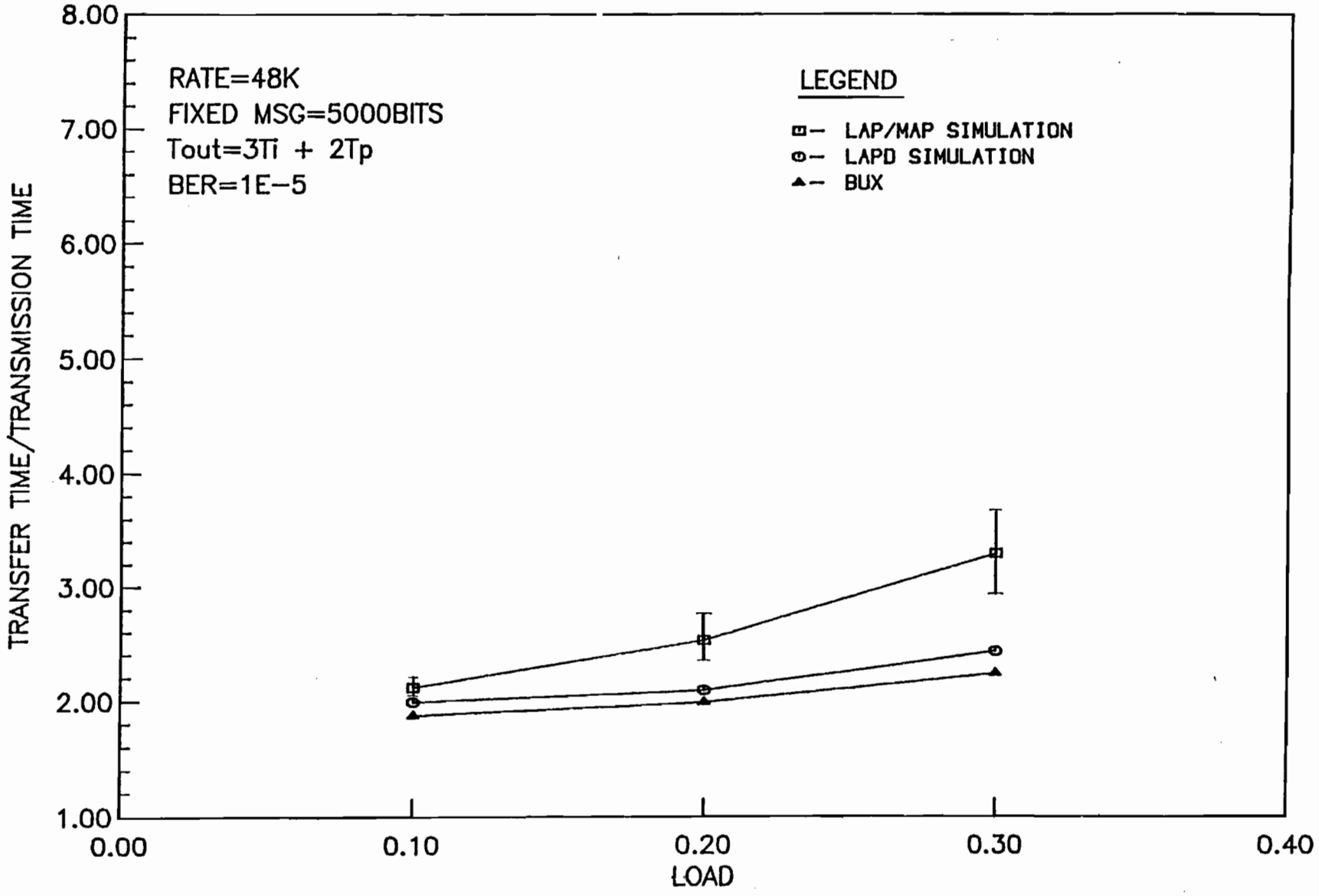


Figure 6.4 LAP-D/CSMA-CD LAN Validation - II

6.2.2 LAP-D/CSMA-CD Validation Approach-II

The second validation approach is to employ Little's formula to validate the LAP-D/CSMA-CD integrated simulation model. The formula called Little's formula after the individual who first proved it [7]. Little's formula says simply that a queuing system with average arrival rate LAMBDA and mean delay time E(T) through the system has an average queue length of E(n) given by the expression:

$$\text{LAMBDA} * E(T) = E(n)$$

The relation among these quantities are shown in figure 6.5. The relation given in Little's formula is very general and is valid for all types of queuing systems,i.e.,M/M/1,M/G/1,etc. The complete description and proof of Little's formula is given in [7,8]. This formula can also be written as

$$E(T) = E(n) / \text{LAMBDA}$$

which means if we know average queue length and arrival rate, then mean time delay through the system can be obtained.

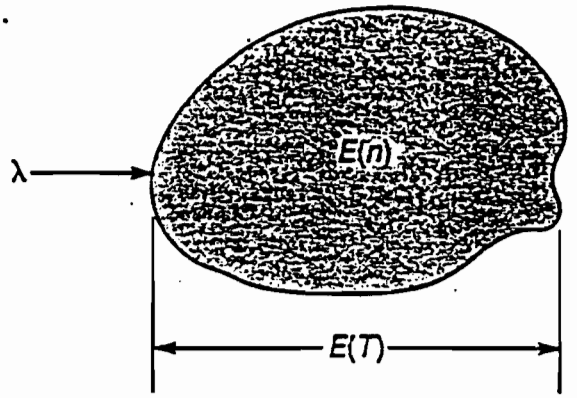


Figure 6.5 The Environment of Little's Formula (From[8])

We have applied Little's result to validate our LAP-D/CSMA-CD integrated model. The simulated average end-to-end delay is to be validated using Little's result. To accomplish this, we must know the average packet arrival rate to the "system", and average queue length of the "system" to calculate average delay. The arrival rate is theoretically found by using given values of offered load, average message length, and line rate. Also to obtain average queue length of the system, we incorporate a SLAM[11] time-persistent variable in which measures average number of entities (I-frames) in the system for a given time interval(simulation run time). By the use of Little's formula, average end-to-end delay is calculated by dividing the average number in the system by arrival rate. This number is then compared to results obtained from simulation by measuring end-to-end delay of each I-frame passing through the system.

Figures 6.6 and 6.7 shows a comparison between LAP-D/CSMA-CD integrated simulation results and results obtained from Little's formula. The parameters used for Link Access protocol are the same as Bux's parameters given in table 2,and Medium Access Protocol(CSMA/CD) parameters given in table 1. Figures 6.6 and 6.7 measures ratio of average transfer time to average transmission time versus offered load. It can be observed that LAP-D/CSMA-CD simulation results are in close agreement with results obtained from Little's formula. This approach further validates the integrated LAP-D/CSMA-CD simulation model.

To further validate our LAN model, the MAC and LLC parameters are modified to be compatible to one of most commonly used LANs. Tables 3 and 4 shows the modified parameters for results given in figure 6.8. Figure 6.8 shows delay versus offered load of comparison between simulation results and Little's formula results. It is noted that again simulation results are in close agreement with Little's results.

Section 6.2 showed that the integrated LAP-D/CSMA-CD LAN simulation model is valid and accurately simulate a combined Logical Link Control Protocol and Medium Access Control Protocol for the study of local area networks.

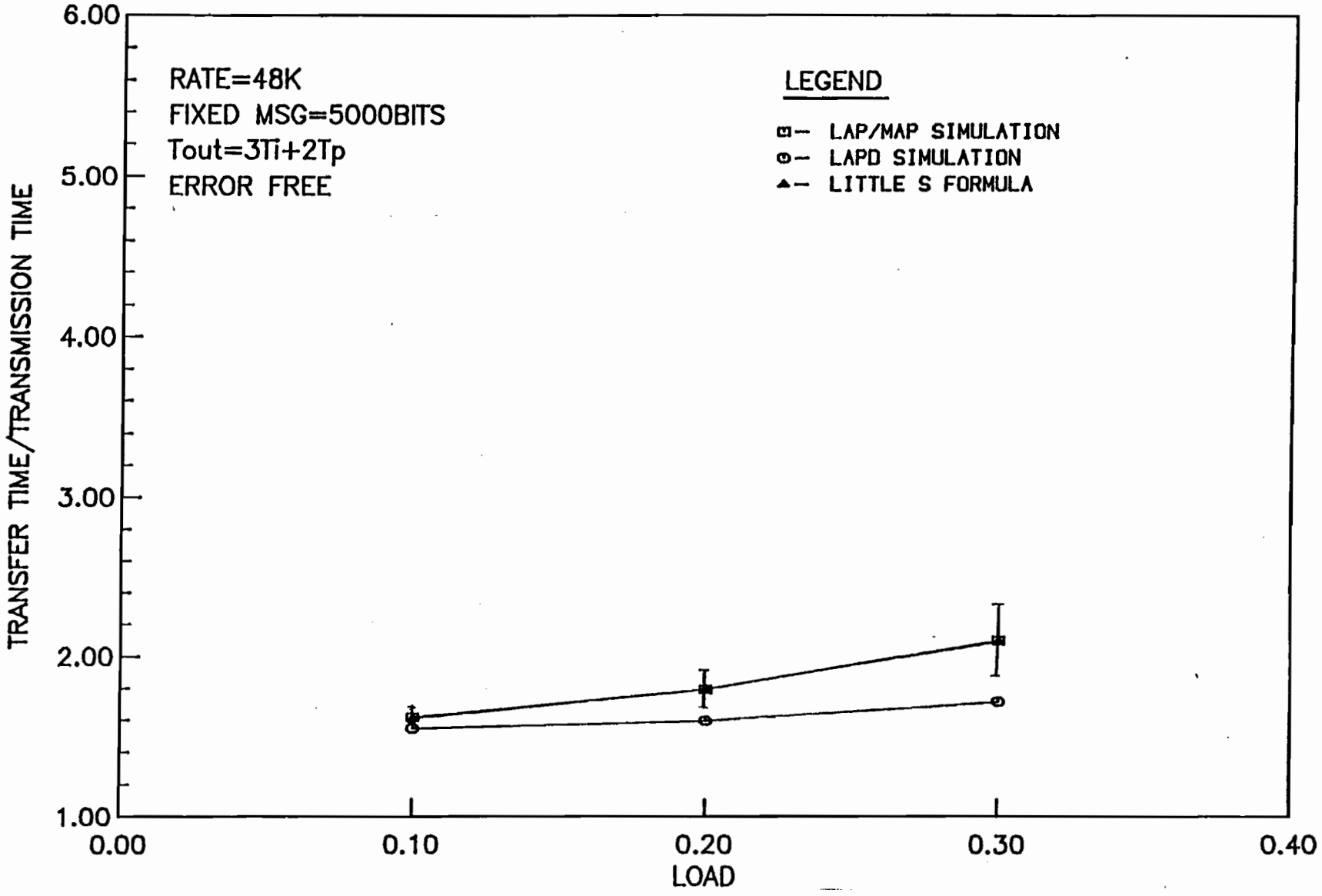


Figure 6.6 LAP-D/CSMA-CD LAN Validation - III

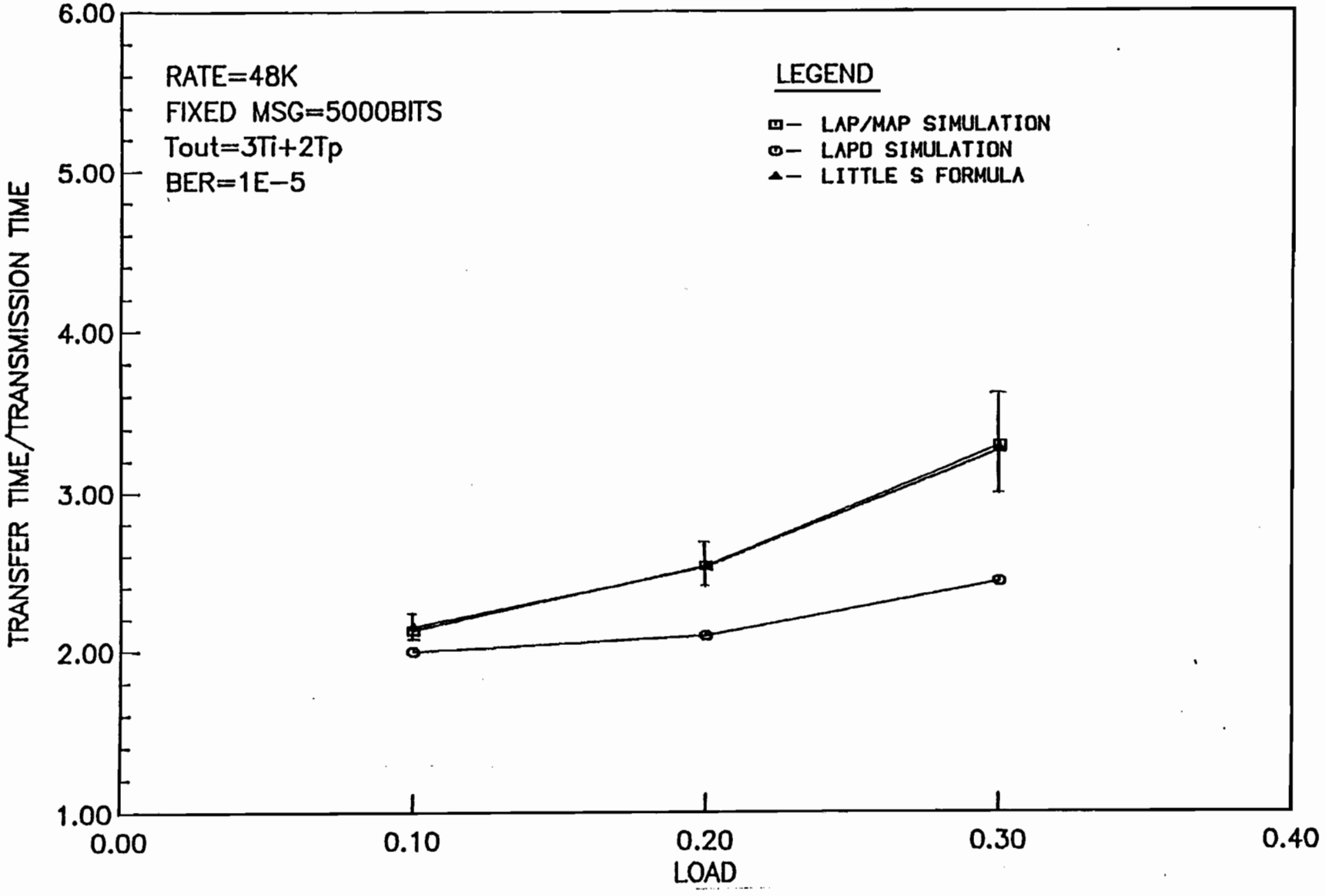


Figure 6.7 LAP-D/CSMA-CD LAN Validation - IV

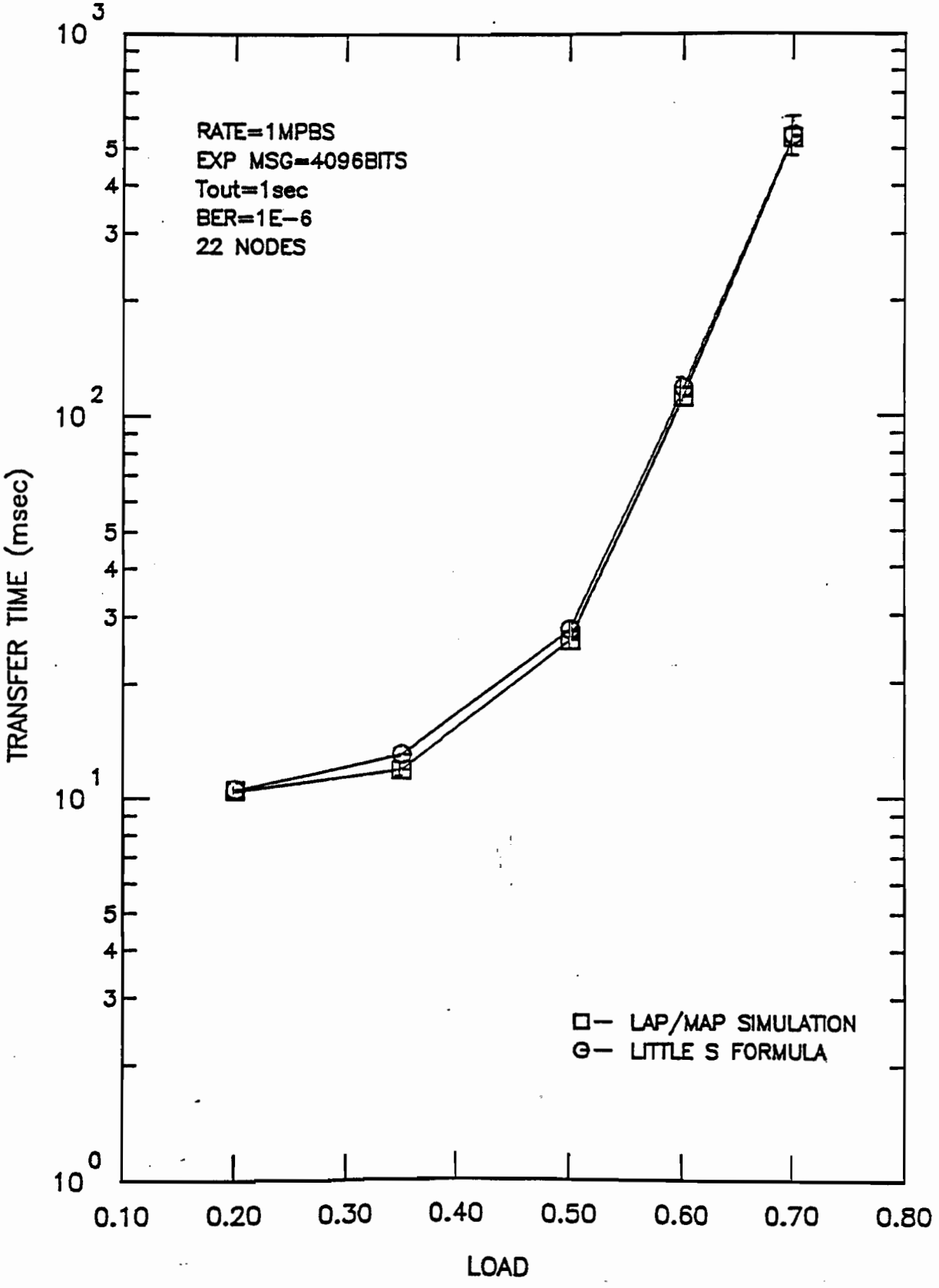


Figure 6.8 LAP-D/CSMA-CD LAN Validation - V

6.3 Performance Measures of LAP-D/CSMA-CD Integrated LAN Model

This section describes the performance LAN. The performance analysis considers the effect of timeout period, method of background load generation, and window size/timeout period on the integrated simulation model. The first section discusses the performance of LAP-D/CSMA-CD simulation under different timeout periods , the next section considers the performance of a 22-node local area network with and without background load. The last section evaluates the effect of window size on the performance of LANs.

The experiments performed in this section have the same MAC parameters but different LLC parameters. The MAC parameters used is given in table 3, and LLC parameters are defined based on the experiment performed and are given in each associated section. In addition to the LLC and MAC parameters, we incorporated processing times for all experiments performed for LLC and MAC layers. The LLC processing times used in the performance study are fully described in [2]. Table 5 shows the processing times for LLC and MAC used in integrated LAN model.

Table 6.1 Simulated CSMA/CD Parameters for Comparing to Bux's Results

Line Rate	48Kbps
Slot Time	10667 Usec(512 bits)
Interframe Gap	2000 Usec(96 bits)
Jam Time	667 Usec(32 bits)
Attempt Limit	16
Backoff Limit	10
Max.Pkt.Size	N/A
Min.Pkt.Size	56 bits
Overhead	48 bits
Node-to-Bus Delay	0.8333 Usec
End-to-End Bus Delay	0.0033 Usec

Table 6.2 Logical Link Control Parameters For Comparing to Bux's

Ave.Pkt.Size	5000 bits
Timeout Period	3Ti +2Tp
Window Size	8
Prob.Bit in Error	1e-5
Prob.Pkt.in Error	4.87e-2
Tp=Processing Time	50 msec

Ti=Transmission Time of an I-Frame

Table 6.3 Simulated CSMA/CD Parameters For Performance Studies

Line Rate	1Mbps
Slot Time	512 Usec
Interframe Gap	96 Usec
Jam time	32 Usec
Attempt Limit	16
Backoff Limit	10
Max.Pkt.Size	12144 bits
Min.Pkt.Size	512 bits
Levels 1 2 Overhead	256 bits
Node-to-Bus Delay	0.8333 Usec
End-to-End Bus Delay	0.0033 Usec

Table 6.4 Logical Link Control Parameters For LAP-D/CSMA-CD Validation

Ave.Pkt.Size	4096 bits
Timeout Period	1.0 sec
Window Size	8
Prob.Bit in Error	1e-6
Ave. Prob.Pkt.in Error	4.342e-3

Table 6.5 Integrated LAN Processing Times

Logical Link Control Protocol Processing Times

ACKPKTS = 0.20E+3 usec

BMOVE = 1.85E+3 usec

ENQUEUE = 0.40E+3 usec

DEQUEUE = 0.55E+3 usec

GT COMM = 0.10E+3 usec

GV RESP = 0.10E+3 usec

INF PKT = 0.30E+3 usec

RCV MSG = 0.10E+3 usec

RECEIVE = 0.15E+3 usec

SEND = 0.35E+3 usec

SND SPR = 0.10E+3 usec

SND MSG = 0.10E+3 usec

Medium Access Control Protocol Processing Times

SENSING = 0.10E+3 usec

COLLSN = 0.30E+3 usec

DEFFRNG = 0.10E+3 usec

CRC CHK = 0.20E+3 usec

6.3.1 Performance Analysis - Timeout Period Study

In section 6.2.1 LAP-D/CSMA-CD validation results showed the crucial effect of timeout period on the delay versus load performance of integrated LAN model. When a node's timer expires before receiving an acknowledgment for a transmitted I-frame, then this node (node A) sends a RR supervisory command frame to the other node (node B) to inquire about node B's current receiver sequence number, then node B immediately transmits a response RR supervisory frame to node A, assuming node B is not already transmitting , and informs node A about its receiver sequence number. Then node A decides if there is any frames to be retransmitted. It is seen that timeout recovery procedure imposes at least 2 S-frame transmission time delay on the I-frames waiting for transmission, in case of StarLAN this delay would be as much as 1 milisecond assuming that these S-frame would get successfully transmitted at the first attempt. It is observed that , if timeout period is too short, then in normal I-frame transmission, unnecessary delay is imposed on the I-frames waiting for transmission. The effect of timeout period can easily be observed on high loads because of CSMA/CD characteristics at these loads. Timeout period can be a crucial factor at higher loads because of the following reasons:

First I/S-frame transfer time is considerably greater at higher loads which in turn increases the chance of getting a timeout. Second timeout recovery becomes very slow due to the fact that a node which is in timeout recovery status has to wait minimum of 2 S-frame transmission time before it can start transmitting I-frames. This waiting time is considerably large for highly loaded networks. Third factor is when number of timeouts increase, then instead of sending I-frames, station sends S-frames to recover the timeout , which directly effects the station's throughput(useful information bits).

On the other hand if the timeout period is too long, then the delay performance becomes worse. This is because timeout is the only means of detecting certain errors. If timeout is too long, detection and error recovery is thereby slowed down which makes the overall transfer time longer. The MAC and LLC parameters for this experiment are given in tables 3 and 6 respectively.

To perform the timeout period performance analysis of integrated LAP-D/CSMA-CD model, a LAN with 10 nodes, fixed message length(fixed transmission time), and exponential interarrival times is modeled. Also chose 4 different timeout periods for this experiment which allows us to observe the effect of different timeouts on the load and delay performance. The first timeout period is chosen to be 40 milisecond (7Ti + 2Tp)(Tp = Processing times) which is considered to be relatively short timeout period for LANs that have CSMA/CD as the Medium Access Protocol. The next timeout period considered is 100 msec(18Ti + 2Tproc) which is a relatively moderate timeout period that accommodates low(0.10) and medium(0.5) loads without much degradation in delay and throughput. The third timeout period is 1 second which is the LAPD default timeout period. This timeout period is considered to be long for lightly loaded LANs. Finally the fourth timeout period is considered to be 3.0 second. This timeout period is extremely long and totally unsuitable for networks with transmission errors at low and medium loads.

Figure 6.9 shows the delay performance comparisons of different timeout periods. It is observed that 40 msec timeout period is an optimum value for the low load in terms of delay and throughput performance. Eventhough there are some transmission errors(BER=1e-5), the transfer time is very close to packet transmission time, this shows that timeout and error recovery are performed with

the least amount of delays. On the other hand this timeout period will not be suitable for medium and high loads. Referring to figure 6.9, it shows the delay performance of a LAN with 100 msec for the timeout period. This timeout period outperforms the other 3 timeout periods and can accommodate both low and high loads. Figure 6.9 clearly shows the delay performance degradation of low loads for timeout periods of 1 and 3 seconds. It is also observed that at high load of 0.75 regardless of what timeout period is, the average packet end-to-end delay is within 95% confidence interval for all timeouts. Clearly what makes the difference is the network's throughput. When timeout is short for a highly loaded CSMA/CD network, there would be a lot of timeouts occurring in normal I-frame transmission which causes the end user to transmit fewer number of I-frames. This effect is clearly shown in table 7 which shows a comparison between different timeout periods and loads based on the number of useful I-frames transmitted by a LAN on a given simulation run time. This value is approximately 2500 I-frames.

Table 6.6 Logical Link Control Parameters For TMOUT PRD Study

Ave.Pkt.Size.....4096 bits

Timeout Period.....Variable

Window Size.....8

Prob.Bit in Error.....1e-5

Ave.Prob.Pkt.in Error.....4.258e-2

Table 6.7 Load VS. I-Frames Transmitted

	USEFUL I-FRAMES TRANSMITTED			
Tout / load	40.0 msec	100.0 msec	1.0 sec	3.0 sec
0.25	2410	2401	2392	2384
0.50	2324	2393	2401	2384
0.75	1784	1964	2016	1890

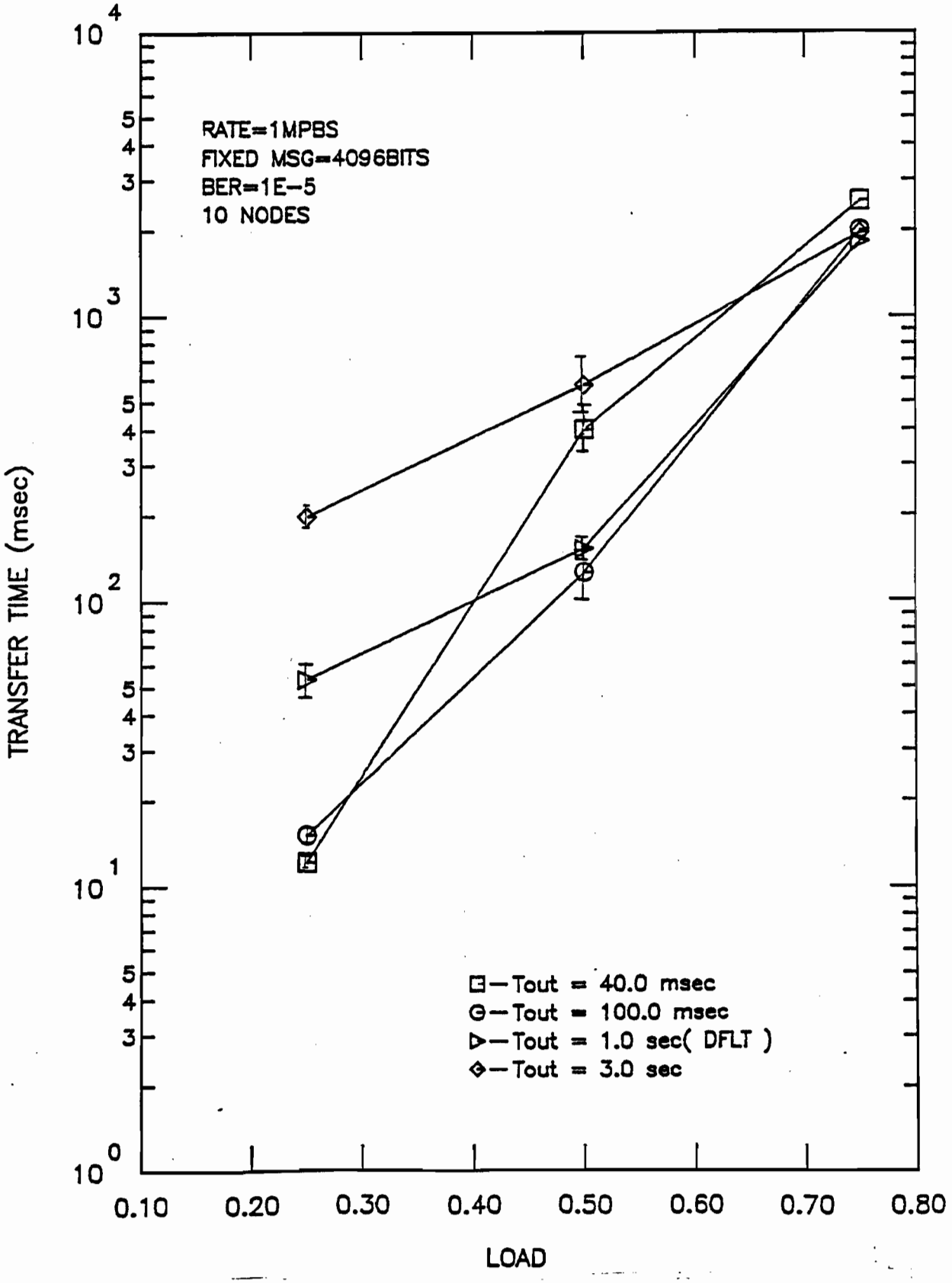


Figure 6.9 LAP-D/CSMA-CD LAN Timeout Performance Study

This section discussed the effect of different timeout periods on the integrated LAN model. It was shown that because of dynamic behavior of CSMA/CD at high loads, we can not achieve an optimum delay and throughput , if we incorporate the same timeout periods for both low and high loads. To resolve such complication, it is the best that timeout period be adaptive and should continuously adjust the timeout period based on the load on the network or the largest possible acknowledgment delays. It should be noted that if a LAN can achieve the probability bit error rate specified by [9], one would expect variations to be much smaller in magnitude, allowing the use of a fixed timeout period. This fact is shown in section 6.3.3 where we analyze the effect of extended window size and timeout period. If a LAN can not achieve the bit error rate(1e-8), it becomes a necessity to use a dynamic timeout period in order to optimize LAN's delay and throughput performance. Also if the number of discarded I-frames by MAC layer increases(mostly occurs for highly loaded networks with more than 16 nodes in the network), then again it is suggested to have a reasonable timeout period in order to recover the discarded I-frames.

6.3.2 Performance Analysis - Transport Load VS. Background Load Study

This section describes the delay versus load performance of LANs in which all stations are modeled as transport stations. Also a simulation model which uses only a pair of transport stations with the additional stations to generate a network background load. This experiment provide us the necessary information about the LAN performance as seen at transport stations with and without the network background load. The purpose of the transport stations is to provide a detailed simulation of LAN protocols including Medium Access Control Protocol and Logical Link Control Protocol. The simulated LAN model consists of 22 nodes

with exponential interarrival times and exponential message length. The MAC and LLC parameters are given in tables 3 and 8 respectively.

Some have proposed to use simulation models of two primary nodes simulated in detail to operate as transport stations with a set of background nodes to generate the network background load in order to measure LAN performance as seen at transport stations to characterize the network[10]. The simulated LAN model developed here ,shows that the delay versus load performance of all where transport stations are modeled is somewhat different from background station case. This clearly indicates that simulation model in [10] totally ignores the performance effect of other transport stations contributed to the network. Therefore, an accurate performance prediction can not be obtained.

Figure 6.10 shows a delay versus load comparison between all transport stations and the 2 transport stations with 20 background stations to generate the network background load. The experiment was performed in an error free environment(no transmission errors), but timeout and error recovery procedures are activated if acknowledgments are not received within the timeout period or I-frames are discarded by the MAC layer due to excessive collisions. It is observed that at the low loads performance of transport stations versus background stations is the same because of the following reasons: First, at low loads number of collisions are minimal and the limit for excessive collisions(16) would not be reached and therefore no I-frames will be discarded by the MAC layer. Second the I-frames and the S-frames would get through the channel before timer expires which implies no delay because of timeout recovery. On the other hand as the load increases, the performance gap between the all transport stations model and background stations becomes greater. This is

explained by the fact that as the load increases the acknowledgment delay increases which causes some timeouts, therefore an additional delay would be imposed on the packets that are waiting for transmission. Also at high loads not only the number of timeouts increases but also the probability that an I-frame suffers more than 16 collisions in the channel increases which it causes to be discarded by the MAC layer. This phenomenon initiates the error and timeout recovery procedures(retransmission of discarded frame(s)). It is reminded if a packet be discared from one of the background stations no recovery is done for the discarded frame. In addition the background station model does not consider the effect of supervisory frames on the delay and throughput performance of the network. It is noted that for transmitting a S-frame in a highly loaded CSMA/CD network, the delay could be considerable and therefore there would be a delay imposed on the Iframes waiting for transmission at the LLC level or user level. The clarity of above discussion is shown in figure 6.10 in which for loads greater than 0.50 the performance difference between transport stations and background stations becomes greater.

Figure 6.11 shows an experiment with BER=1e-6 or packet error rate of 4.34e-3. This bit error rate is a moderate and theoretically does not match the BER=1e-8 specified by[9]. The goal is to show that as the number of packets in error increases, the gap performance between transport stations and background stations becomes wider. To obtain these results , total of 5500 I - frames was transmitted by the network(excluding LAP retransmissions) which is the same for both error free and error cases. It is observed that if the number of frames discarded by MAC layer increases, this would have the same effect on the delay performance as having a LAN model with transmission errors because in both cases the timeout and error recovery procedures would be used to recover the discarded

frame(s). The results described in this experiment considered a 95% confidence interval.

This section showed that, we can not assume the performance of all transport stations to be the same as two transport stations with 20 background stations for a LAN operating at moderate and high loads. The experiment has shown that it is a valid assumption to say that these models have the same performance at low loads. Also is seen that as the number of nodes in the network increases, the performance difference between transport stations and the background stations becomes greater.

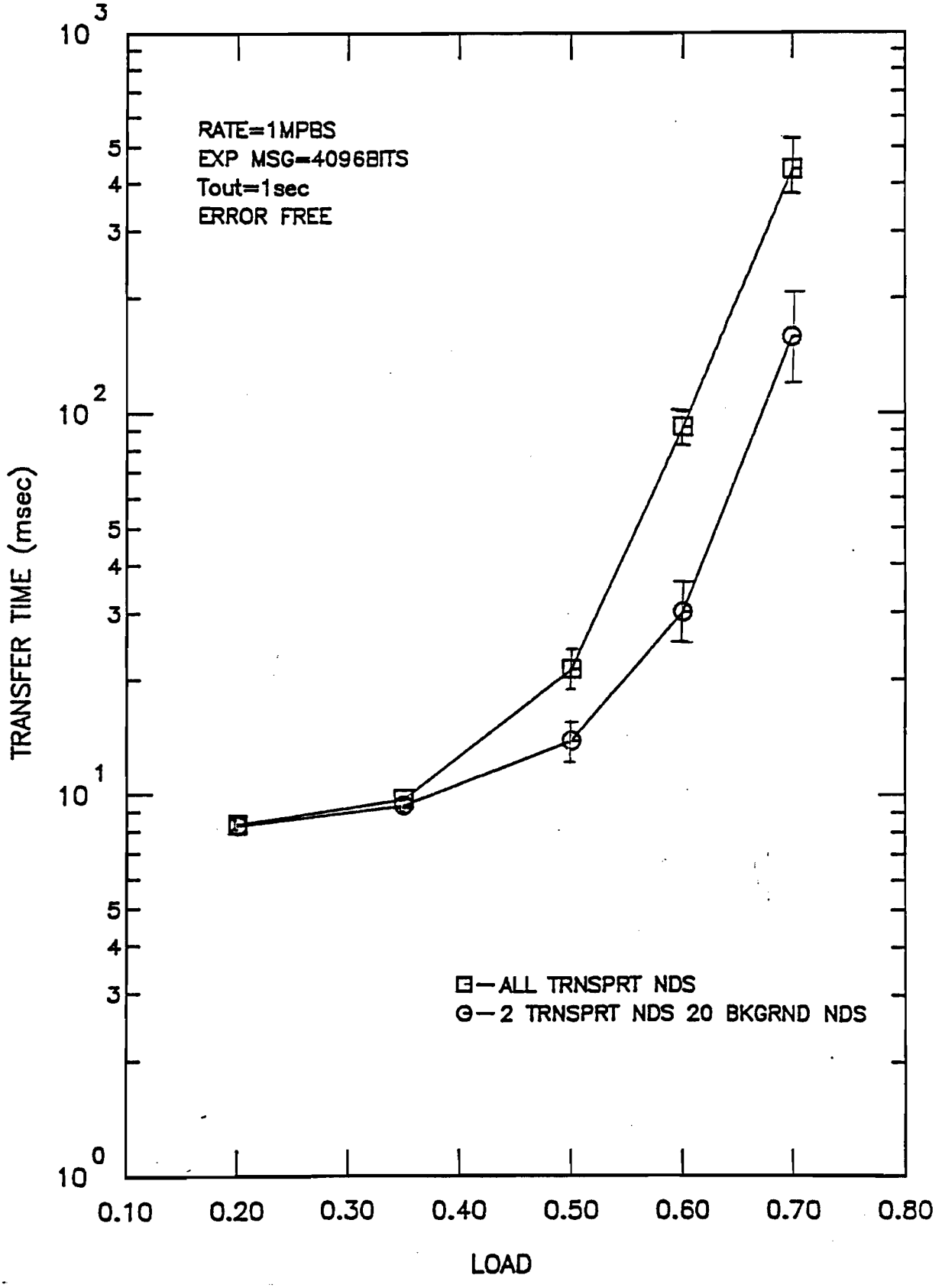


Figure 6.10 LAP-D/CSMA-CD LAN TRNSPRT-BCKGRND Station Performance Study - I

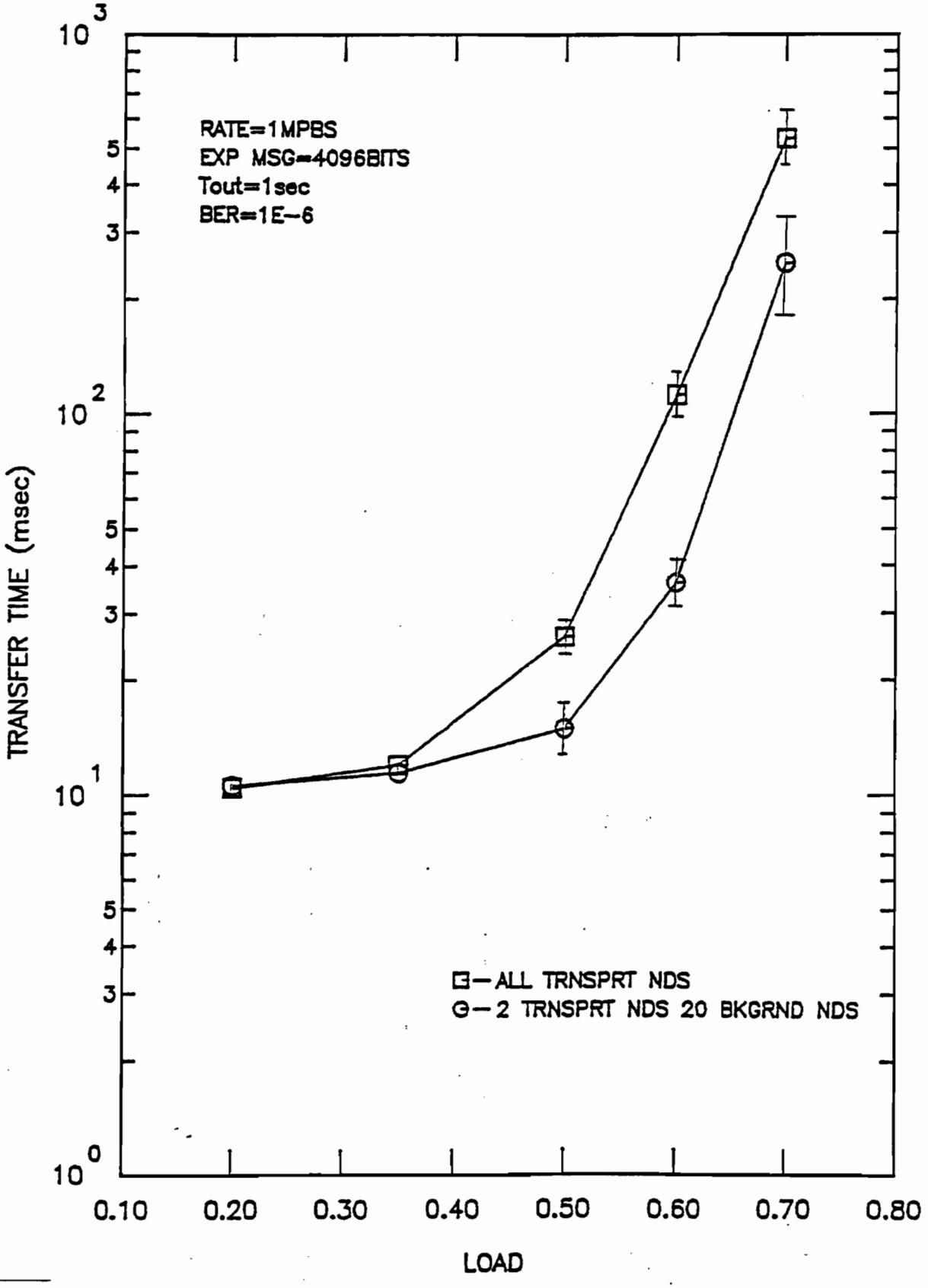


Figure 6.11 LAP-D/CSMA-CD LAN TRNSPRT-BCKGRND Station Performance Study - II

Table 6.8 Logical Link Control Parameters For TRNSPRT/BKGRND Study

Ave.Pkt.Size	4096 bits
Timeout Period	1 sec
Window Size	8
Prob.Bit in Error	Error Free
Ave.Prob.Pkt In Error	Error Free

Table 6.9 Logical Link Control Parameters For WNDW SZ/TMOUT PRD Study

Ave.Pkt.Size	4096 bits
Timeout Period	Variable
Window Size	Variable
Prob.Bit in Error	Error Free
Ave.Prob.Pkt.in Error	Error Free

6.3.3 Performance Study - Extended Window/Timeout Study

This section considers an experiment to vary the window size and timeout period in order to improve the delay versus load performance of a LAN with 22 nodes all operating as transport stations in an error free environment(compatible with BER=1e-8 specified by [9]) with exponential message length and interarrival times. For this experiment the selection of window size and timeout period are mainly based on speculation and previous simulation experiments. The MAC and LAP parameters for this experiment are given in tables 3 and 9 respectively.

Figure 6.12 shows a performance comparison between LANs operating at different timeout periods and window sizes. It was observed that the performance remained unchanged for loads less than 0.5 in an error free environment regardless of window size and timeout period. At low loads, there is no delay in acknowledgment of transmitted I-frames, therefore there would be no timeouts and window never reaches its maximum specied size.

Figure 6.12 shows that for window size of 32 and timeout period of 3 seconds, the lowest delay is achieved as compare with the other two curves. Also it is learned that window size alone cannot improve the performance for the same timeout periods because 1 second timeout period is short for high loads and regardless of window size, a station can not transmit many I-frames because of limited timeout period. This fact is demonstrated by an example. At a load 0.70 for a 22 node network with the parameters given in tables 3 and 9, the message interarrival times for each node is 1.287e+5 microseconds. If we divide the timeout period(1e+6 microseconds) by interarrival times, we observe that a station on the average can transmit about 7-8 I-frames before timer expires,

assuming no acknowledgments received for the transmitted I-frames which is a reasonable assumption at high loads for LANs having CSMA/CD as the Medium Access Protocol. This explains the reason for getting an improvement for expanded window size(32) and increased timeout period (3 seconds). The experiments have shown that beyond this window size and timeout period the delay performance tends to get worse, but it still outperformed the other two cases(W=8 and W=32 and To=1s). This is caused by the limitation of CSMA/CD's attempt limit. In this case when an I-frame was discarded by MAC layer, the error recovery was mostly performed by the REJ procedure, therefore for each case of I-frame recovery, a station on the average had to retransmit as many as 24 I-frames which clearly degraded the delay and throughput performance.

The study performed can be valid only if we are dealing with error free or almost error free(1e-8) LANs.In section 6.3.1, we showed the effect of long timeout periods for a LAN with transmission errors. Also for extended window size tend to adversely effect the delay versus load performance of a LAN with considerable transmission errors. This is caused by the number of retransmissions due to increased number of outstanding I-frames. This experiment shows a tradeoff between improved delay and stations buffer size(window).

This section described the performance analysis of the integrated LAP-D/CSMA-CD LAN model. It was observed that in order to optimize delay and throughput performance of a LAN with transmission errors, an adaptive timeout period is required. Also it was seen that delay performance of a network that all stations operating as transport stations is somewhat different than delay performance of a network under same conditions but not all stations are

operating as transport stations. Finally, it was shown that for appropriate window size and timeout period, LAN's delay versus throughput performance can be improved considerably.

Chapter 6 discussed the integrated LAP-D/CSMA-CD LAN simulation model developed here. This model is unique in sense that can predict the actual performance of local area networks specified by IEEE 802 standards. This chapter described the simulation model and processing times used in Link Access Protocol and Medium Access Protocol. Also discussed the different variable type involved in the integrated LAN model. Finally this chapter discussed the performance study of LAP-D/CSMA-CD model.

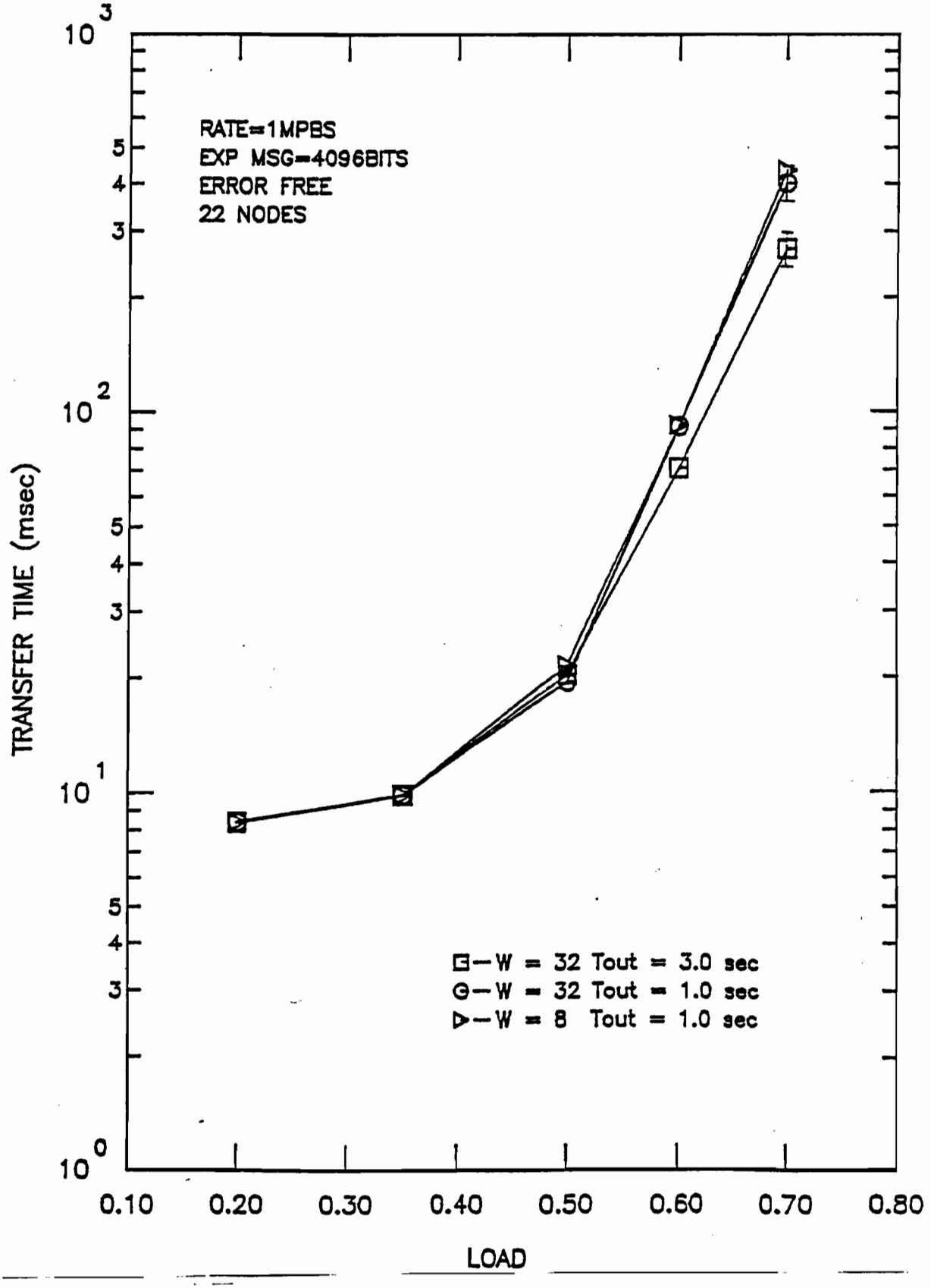


Figure 6.12 LAP-D/CSMA-CD LAN Window Size/Timeout Period Performance Study

References:

1. B.Wood,'A Performance Evaluation of Local Area Networks',Technical Report TISL-6731-3,1985

2. H.Kanakia and F.Tobagi,' Performance Measurement of a Data Link Protocol',ICC 86,pp 894-898

3. W.Zwaenepoel,'Protocols for Large Data Transfers Over Local Area Networks',Proc. of 9th Data Comm. Sympo.,British Columbia, Sept. 1985, pp 22-32

4. E.M.Friedman,' Computer Simulation Model for a CSMA/CD Local Area Network Utilizing a Multirate Voice Coding Technique for Load Control' ,Technical Report,TISL-6731-2,May 1985

5. W.Bux,K.Kummerle,and H.L.Troung,' Data Link-Control Performance: Results Comparing HDLC Operational Modes',Computer Networks,6-1982, pp 37-51

6. W.Bux and H.L.Troung,' A Queuing Model for HDLC-Controlled Data Links',Flow Control in Computer Networks,IFIP,North-Holland Publishing Company,1979,pp 287-306

7. D.C.Little,' A Proof of the Queuing Formula L = LAMBDA * W ', Operations Research,Vol.9, No.3,1961, pp 383-387

8. M.Schwartz,'Telecommunications Networks',Addison-Wesley Publishing Company,1987,pp 21-65

9. ANSI/IEEE 802.3,' Local Area Networks CSMA/CD',IEEE Inc.,1985

10. K.Mills,M.Wheatley and S.Heatley,' Prediction of Transport Protocol Through Simulation', IEEE Communication Society Workshop on Computer-Aided Modeling, Analysis and Design of Communication Link and Networks, May 6-8, University of Kansas, Lawrence Kansas, 1986

11. A.B.Pritsker,' Introduction to Simulation and SLAM II',Halsted Press Inc.,1984

CONCLUSION

In order to make improvements in performance of LANs, we must be able to predict the LAN performance accurately as a function of the network parameters. The simulation model developed here is capable of predicting the actual performance of LANs. This model integrated LAP and MAC into a single combined simulation model of a LAN. In addition model allows the explicit specification of the processing times for each protocol function.

The performance analysis using this model showed that with correct processing times and network latencies an accurate prediction of actual LAN performance can be obtained. This model further demonstrated that if LANs operating with moderate transmission errors which is not compatible with specified bit error rate by IEEE 802 Standards(1e-8), it is crucial to have a dynamic timeout period in order to optimize delay and throughput performance.

Some have proposed to use simulation models composed of two primary nodes simulated in detail to operate as transport stations with a set of background nodes to generate the network background load, in order to measure the performance of LANs. Our study showed that LANs with all transport stations have higher delays at medium and high loads than LANs with a combination of transport stations and background stations.

Finally, an experiment was performed to improve delay and throughput performance of LANs. This study considered extended window size and timeout period in an error free environment which meets the IEEE 802 Standard bit error rate requirements(BER=1e-8).

APPENDICES

A COMBINED LAP-D/CSMA-CD SIMULATION MODEL FOR THE STUDY OF LOCAL AREA NETWORKS

by

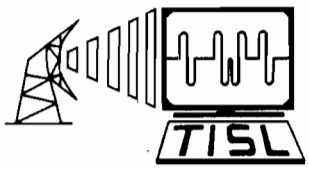
Mohammed S. Kashefipour
Graduate Research Assistant

Telecommunications and Information Sciences Laboratory
University of Kansas
Lawrence, Kansas 66045

Technical Report TISL-6731-4

Supported by
National Science Foundation
and
AT&T Information Systems

Principal Investigator: Dr. Victor S. Frost



TELECOMMUNICATIONS AND INFORMATION SCIENCES LABORATORY
THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.
2291 Irving Hill Drive - Campus West Lawrence, Kansas 66045

Appendix 1

The purpose of this appendix is to describe the routines that make up the Discrete Event portion of LAPD simulation model and a description of SLAM Network Model of LAPD.

As we mentioned is previous chapters, the LAN Model developed here is written in simulation language called SLAM[2](Simulation Language for Alternative Modeling). For this effort the sole interface between the user and simulation is the SLAM driver file as shown in figure A1.1. Details of pertinent SLAM statements will be discussed here. For further details see [2]. The model as represented in the SLAM driver file contains five segments:

1. Parameter definition
2. Statistics file definition
3. Traffic generation
4. Entry into the network
5. Execution parameters

In figure A1.1 lines 4-76 contain the parameter definitions. The traffic timing, simulation run, and network parameters are set using the SLAM INTLC statement and SLAM user xx(.) array. The INLTC statement is used to assign initial values to SLAM variables, in this case the xx(.) array.

The details of the xx(.) array are not significant only the meaning of each element. The meaning of these variables will be discussed in the following sections.

Lines 78-88 inform SLAM processor that statistics will be collected for each node. The statistics of interest are the average number of I-frames in the system, average transmission time of I-frames,average transmission time of all frames, user to LAP queuing delay, average LAP delay, and average user-to-user delay for all the stations in the network.

The generation of traffic is accomplished in lines 161-184. The attribute array, atrib(.), is described first. Each packet which is generated has an attribute array associated with it. This array describes the state of the packet as it progresses through the system. The user of this model need not be concerned with most of values of the attribute array. However, atrib(2) holds source identity, atrib(9) holds packet length in microseconds, atrib(11) holds packet length in bits, atrib(3) holds the destination identity, atrib(4) holds the queue number of the retransmission buffer, and atrib(7) holds the packet timeout period.

After the packet was generated (SLAM CREATE statement), it is queued in the wait node where it is stored until the seizure of resource became available through SLAM ALLOC function. Then packets would enter the network where the specifics of LAPD-D protocol is implemented. This done in lines 186-196.

The last aspect of the model deals with SLAM execution parameters i.e., how the simulation is to run. this is done in lines 1,11-21, and 198-203.

To simulate the rest of LAPD protocol, the Discrete Event approach is used. The network elements that remain to be simulated include transmitting and receiving I-frames and S-frames, retransmission of errored I-frames, and timeout recovery. The LAPD simulation model provides a reliable communications path between two devices. Packet is transmitted from one node and received error free by the other node.

The Discrete Event model consists of ten events and five subroutines. The events are performing the LAPD protocol functions. The subroutines are used to handle receiving/sending of acknowledgements, removing acknowledged I-frames from the retransmission buffer, stop timer for acknowledged or suspended I-frames, and resource allocation for the transmitting nodes. The relationship between events and user subroutines is shown in tables A1.1 and A1.2.

```
GEN,U OF K TISL,LAPD SIMULATION MODEL,4/5/86,5,YES,YES;
LIMITS,7,20,100;
;
;-------------------------------------------------------------------------
;
;     XX(1) - XX(2) ARE THE AVERAGE MESSAGE LENGTHS FOR NODES 1 - 2
;
INTLC,XX(1)=5000.00,
      XX(2)=5000.00;
;-------------------------------------------------------------------
;
;    SET SIMULATIONS RUN AND TIMING PARAMETERS
;
INTLC,XX(25)=5.3325E+8; XX(25) = TOTAL LENGTH OF THE SIMULATION RUN
;
INTLC,XX(26)= 2.0;      XX(26) = TOTAL NUMBER OF ACTIVE NODES( MAX = 22 )
;
INTLC,XX(27)=5.0;       XX(27) = NUMBER SIMULATION SUBRUNS
;
;-------------------------------------------------------------------
;
;     XX(50) - XX(71) ARE THE AVERAGE INTERARRIVAL TIMES ( IN MICRO SEC )
;                     FOR NODES 1 - 2 RESPECTIVELY
;
INTLC,XX(50)=1.667E+5,
      XX(51)=1.667E+5;
;-------------------------------------------------------------------
;   XX(75) - XX(128) SPECIFY THE NETWORK PARAMETERS AND TOPOLOGY
;
;
INTLC,XX(75)=0.016;     XX(75) = LINE RATE IN MBPS
;
INTLC,XX(81)=12144;     XX(81) = MAXIMUM PACKET SIZE IN BITS
;
INTLC,XX(82)=56.0;      XX(82) = MINIMUM PACKET SIZE IN BITS
;
INTLC,XX(83)=48.0;      XX(83) = NUMBER OF OVERHEAD BITS/PACKET
;
; NOTE THAT IF XX(110) IG GREATER THAN 8 THEN SET XX(83) TO
; 56.0 BITS INSTEAD OF 48.0 BITS
;
INTLC,XX(110)=3.0;      XX(110)= MAX   OF OUTSTANDING FRAMES(BY DEFAULT)
;
INTLC,XX(111)=0.4155E6; XX(111)=TIMEOUT PERIOD (3 TI + 2 TP)
;
INTLC,XX(112)=0.0;      XX(112)=PROBABILITY OF A PACKET BEING IN ERROR
;
INTLC,XX(113)=1.0E-6;   XX(113)= PROBABILITY OF A BIT BEING IN ERROR
;
INTLC,XX(114)=1.0E+10;  XX(114)= IF NEEDED USED FOR TRACING AND DEBUGGING
;                                PURPOSES
```

Figure A1.1 LAPD SLAM Network Model Driver File

```
52  ;
53  INTLC,XX(115)=0.0;       XX(115)= VAR TO COLLECT STAT. ON AVE.   IN THE SYSTEM
54  ;
55  INTLC,XX(116)=50.00E+3;  XX(116)= ONE WAY PROPAGATION DELAY
56  ;
57  INTLC,XX(117)=0.00E+3;  XX(117)= ACKPKTS PROCESSING TIME IN MICRO SEC
58  ;
59  INTLC,XX(118)=0.00E+3;  XX(118)= BMOVE   PROCESSING TIME IN MICRO SEC
60  ;
61  INTLC,XX(119)=0.00E+3;  XX(119)= ENQUEUE PROCESSING TIME IN MICRO SEC
62  ;
63  INTLC,XX(120)=0.00E+3;  XX(120)= DEQUEUE PROCESSING TIME IN MICRO SEC
64  ;
65  INTLC,XX(121)=0.00E+3;  XX(121)= GT COMM PROCESSING TIME IN MICRO SEC
66  ;
67  INTLC,XX(122)=0.00E+3;  XX(122)= GV RESP PROCESSING TIME IN MICRO SEC
68  ;
69  INTLC,XX(123)=0.00E+3;  XX(123)= INF PKT PROCESSING TIME IN MICRO SEC
70  ;
71  INTLC,XX(124)=0.00E+3;  XX(124)= RCV MSG PROCESSING TIME IN MICRO SEC
72  ;
73  INTLC,XX(125)=0.00E+3;  XX(125)= RECEIVE PROCESSING TIME IN MICRO SEC
74  ;
75  INTLC,XX(126)=0.00E+3;  XX(126)= SEND    PROCESSING TIME IN MICRO SEC
76  ;
77  INTLC,XX(127)=0.00E+3;  XX(127)= SND SPR PROCESSING TIME IN MICRO SEC
78  ;
79  INTLC,XX(128)=0.00E+3;  XX(128)= SND MSG PROCESSING TIME IN MICRO SEC
80  ;
81  ;-----------------------------------------------------------------
82  ;
83  TIMST,NNQ(1),AVE NUM IN QUE1;
84  TIMST,NNQ(2),AVE NUM IN QUE2;
85  TIMST,XX(115),AVE NUM IN SYSTM;
86  STAT,1,TRNS TIME I FRMS;
87  STAT,2,TRNS TM ALL FRMS;
88  STAT,5,USR TO LAP Q DLY;
89  STAT,6,AVE. LAP DELAY;
90  STAT,7,USR1 TO USR2 DLY;
91  STAT,8,USR2 TO USR1 DLY;
92  STAT,29,AVE USR MSG DLY;
93  STAT,30,AVE.BFR HLDNG TM;
94  STAT,31,AVE.MESSAGE SIZE;
95  ;
96  NETWORK;
97  ;
98  ;     VARIABLE DEFINITIONS:
99  ;
100 ;       ATRIB(1) = PACKET CREATION TIME
101 ;       ATRIB(2) = NODE IDENTIFICATION
102 ;       ATRIB(3) = DESTINATION ADDRESS OF A PACKET
103 ;       ATRIB(4) = FILE   OF THE REPEAT QUEUE
```

```
104  ;        ATRIB(5) = PACKET COUNT NUMBER
105  ;        ATRIB(6) = PACKET SEND SEQUENCE NUMBER ( N(s)   PROTOCOL VERSION )
106  ;        ATRIB(7) = PACKET TIMEOUT PERIOD
107  ;        ATRIB(8) = THE COMMUNICATION INDICATOR BETWEEN DATA LINK LAYER
108  ;                   AND APPLICATION LAYER IS DEFINED AS FOLLOWS:
109  ;                   ATRIB(8) = 1.0 INDICATES THAT LAP IS READY TO ACCEPT
110  ;                   I - FRAMES FROM APPLICATION LAYER
111  ;                   ATRIB(8) = 2.0 INDICATES THAT LAP IS SENDING AN I - FRAME
112  ;                   TO THE APPLICATION LAYER
113  ;        ATRIB(9) = PACKET LENGTH IN MICROSECONDS
114  ;        ATRIB(10)= USED FOR TIMER RESETTING PURPOSES ANS IS DEFINED
115  ;                   AS FOLLOWS:
116  ;                   ATRIB(10)= 0.0 STOP TIMER FOR ALL ACKNOWLEDGED FRAMES
117  ;                   ATRIB(10)= 1.0 STOP TIMER ONLY FOR ONE PARTICULAR
118  ;                   I - FRAME . THIS IS THE CASE WHEN NODE IS IN TIMER
119  ;                   RECOVERY CONDITION
120  ;                   ATRIB(10)= 2.0 STOP TIMER FOR ALL TRANSMITTED FRAMES .
121  ;                   IT IS USED WHEN NODE IS IN REJ RECOVERY CONDITION
122  ;        ATRIB(11)= NUMBER OF BITS PER PACKET
123  ;        ATRIB(12)= MEASURES LAP DELAY
124  ;        ATRIB(13)= NOT USED
125  ;        ATRIB(14)= SIGNIFIES IF PACKET IS CORRECT,IN ERROR,OR OUT OF SEQUENCE
126  ;                   ATRIB(14) = 0.0 CORRECT PACKET
127  ;                   ATRIB(14) = 1.0 PACKET IN ERROR (TRANSMISSION ERROR)
128  ;                   ATRIB(14) = 2.0 OUT OF SEQUENCE PACKET
129  ;        ATRIB(15)= INDICATES THE TYPE OF A FRAME (COMMAND/RESPONSE)
130  ;                   ATRIB(15) = 1.0   COMMAND FRAME
131  ;                   ATRIB(15) = 2.0   RESPONSE FRAME
132  ;        ATRIB(16)= MARKS THE PACKET COUNT NUMBER THAT A RECEIVER HAS RECEIVED
133  ;                   PLUS ONE
134  ;        ATRIB(17)= THE PACKET SEQUENCE NUMBER THAT RECEIVER EXPECTS TO RECEIV
135  ;                   ( V(r) - PROTOCOL VERSION )
136  ;        ATRIB(18)= INDICATES THE PACKET I.D. AND IS DEFINED AS FOLLOWS:
137  ;                   ATRIB(18) = 0.0 I - FRAME
138  ;                   ATRIB(18) = 1.0 S - FRAME (RR FRAME USED FOR ACKNOWLEDGEME
139  ;                   ATRIB(18) = 2.0 S - FRAME (REJ FRAME )
140  ;                   ATRIB(18) = 3.0 S - FRAME (RR FRAME USED FOR TIMER
141  ;                               RECOVERY CONDITION )
142  ;        ATRIB(19)= INDICATES THAT THE DATA LINK LAYER ENTITY TRANSMITTING
143  ;                   ATRIB(19) HAS RECEIVED ALL I - FRAMES NUMBERED UP
144  ;                   TO AND INCLUDING ATRIB(41) - 1 ( N(r) - 1 PROTOCOL
145  ;                   VERSION )
146  ;        ATRIB(20)= TIMEOUT EVENT INDICATOR
147  ;                   ATRIB(20)= 0.0 NON TIMEOUT EVENT
148  ;                   ATRIB(20)= 1.0 TIMEOUT EVENT
149  ;
150  ;      XX(25) = THE THE TOTAL SIMULATION TIME OVER ALL RUNS USED IN CALC OUTPU
151  ;               STATISTICS
152  ;
153  ;        FILES   1-2  RESERVED FOR THE AWAIT NODE AT EACH STATION
154  ;        FILES   3-4 RESERVED FOR THE REPEATE QUEUE AT EACH STATION
155  ;        FILE    5    CHANNEL QUEUE NODE
```

```
;       FILE    6    SUPERVISORY FRAMES WAITING FOR TRANSMISSION
;       File    7    NOT USED
;       FILE   48    EVENT CALENDAR
;
       RESOURCE/NODE1,1/NODE2,2;
;-------------------------------------------------------------------------
;
      CREATE,EXPON(XX(50),1),,1;CREATE PACKETS AT NODE 1
      ASSIGN,ATRIB(2)=1.0,
             ATRIB(3)=2.0,
             ATRIB(4)=ATRIB(2) + XX(26),
             ATRIB(11)=USERF(1),
             ATRIB(9)=ATRIB(11)/XX(75),
             ATRIB(7)=XX(111);
      ASSIGN,XX(115)=XX(115)+1;
      AWAIT(1),ALLOC(1);
      ACT,,,CHNL;                      IMMEDIATE BRANCH TO THE CHANNEL
;
;-------------------------------------------------------------------------
;
      CREATE,EXPON(XX(51),5),5000.5,1;
      ASSIGN,ATRIB(2)=2.0,
             ATRIB(3)=1.0,
             ATRIB(4)=ATRIB(2) + XX(26),
             ATRIB(11)=USERF(1),
             ATRIB(9)=ATRIB(11)/XX(75),
             ATRIB(7)=XX(111);
      ASSIGN,XX(115)=XX(115)+1;
      AWAIT(2),ALLOC(2);
      ACT,,,CHNL;
;
;-------------------------------------------------------------------------
;
;     COMBINE STATIONS TO FORM CHANNEL
;     ================================
;
CHNL  QUEUE(5);                            DUMP ENTITIES INTO QUEUE
;
      ACT;                                 IMMEDIATE BRANCH TO CHANNEL MODEL
;
      EVENT,1;                             GATEWAY TO DISCRETE EVENT PROGRAM
;
      END;
;
INIT,0.0,5.3325E+8;
MONTR,CLEAR,1.0665E+8;
;
SIMULATE;
MONTR,CLEAR,1.04E+8
;
FIN;
```

Table A1.1 Event Relationship in the Discrete Event Model

Event Code Called	Event Name	Schedules To	Subroutines
1	sndifrm	trnsmt	colct *
		sndsfrm	filem *
			schdl *
2	trnsmt	laptrns	schdl *
		timeout	copy *
		applclyr	
3	laptrns	laprcv	colct *
		trnsmt	schdl *
		sndsfrm	
		applclyr	
4	laprcv	rcvifrm	schdl *
		rcvsfrm	
5	rcvifrm	sndsfrm	schdl *
		applclyr	colct *
sndrcvack			
			filem *
6	sndsfrm	laptrns	copy *
			rmove *
			schdl *
7	rcvsfrm	timeout	schdl *
		sndsfrm	filem *
		rexmit	srch
rmvackfrm			
8	timeout	rexmit	schdl *
		sndsfrm	srch
rmvackfrm			
9	rexmit	trnsmt	copy *
			schdl *
10	applclyr	-	free *

* => SLAM Subroutine

Table A1.2 User Subroutines in Discrete Event Model

Subroutine Name	Schedules To	Call To
sndrcvack	sndsfrm	schdl *
		srch
		rmvackfrm
rmvackfrm	applclyr	rmove *
		colct *
		copy *
srch	-	rmove *
		copy *
alloctrsc	-	seize *

* => SLAM Subroutine

In the following text, each of the events and subroutines will be described. It is assumed that the reader has a basic knowledge of computer programming in FORTRAN. In addition, it is assumed that the user has a basic understanding of how Discrete Event Models are implemented in SLAM[2]. The actual routines that we are about to discuss are shown at the end of Appendix 1.

The first portion of the Discrete Event Model that we will consider is the PARAMS.DAT file. This file is not a routine, but it is simply a section of code that appears at the beginning of each FORTRAN subroutine in the model via the statement "include params.dat". The purpose of this section of this code is to establish all global variables, and provide a FORTRAN COMMON block for storage of these common variables. This process is shown in figure A1.2

The next part of the Discrete Event Model that we will discuss is the INITLAP subroutine. This subroutine is called by INTLC which in turn called by SLAM[2] before the beginning of each simulation run. Its purpose is to set some of the initial conditions that are needed to simulate LAPD protocol. First convert user parameters that were defined in the SLAM[2] Driver File as XX variables to FORTRAN user defined variables. This allows us to refer to these variables by symbolic variable names instead of XX array elements. Next, we proceed to set the initial conditions on certain elements in the attribute array. Finally, the user defined variables which are used to simulate LAPD protocol are initialized. This routine is shown in figure A1.3.

The next part of the Discrete Event Model that we consider is the EVENT file. Here is where most of the routines that simulate the LAPD protocol. The first routine in the EVENT file is the EVENT subroutine. This subroutine is required by all SLAM Discrete Event Models. Its purpose is to allow subroutines to be called by their event number as opposed to their subroutine name. Note that the EVENT subroutine only deals with the event routines, not the user subroutines.

The first event in the EVENT file is the SNDIFRM event. This event is scheduled from the SLAM Network Model. Here packets are routed from user to Data Link Layer. This layer is to perform some LAP functions on this packet then give it to physical layer for

transmission. This event performs the following functions: First collect statistics for time that this packet was queued in the application layer. Next assign a sequence number to this packet based on sender's sequence number , and also for retransmission purposes store a duplicate of this packet in the retransmission buffer which in simulation defined based on the source node and maximum number of nodes in the network.For example, for a 2-node network The next step would to send this packet to lower part of data link layer for piggybacking and other LAP functions, but before doing that certain conditions must be met before doing so. First if this node is in retransmit status increment the number of outstanding I-frames to be transmitted. This condition can only occur when application layer does not know the current status of data link layer. It is noted that when data link layer is in retransmit condition would not accept any I-frames from the higher layers. If source is idle and no S-frames waiting for transmission then another section of data link layer will be invoked for piggybacking and other functions.

The next event is the TRNSMT event. This event performs the main functions for retransmission of I-frames, piggybacking of acknowledgements, window rotation and timer settings. This event is scheduled from SNDIFRM, REXMIT, and LAPTRNS events. Every time this event is scheduled certain logic is performed regardless of where this event is scheduled. First define source , destination, and retransmission buffer file number. Next is to check if this node is

beginning to retransmit, if so the sender's sequence number is not incremented since it has been already reset to requested I-frame sequence in REXMIT Event. The Next step would be to search the retransmission buffer for the packet specified by the send sequence number. After this entry was found, data link layer assigns a piggyback acknowledgement to this outgoing I-frame, and set the transmitter busy. If source retransmitting decrement number of outstanding I-frames transmitted. For each outgoing I-frame start a timer as specified by LAP. If an acknowledgment not received for this frame within the specified time(timeout period), then procedure for timeout condition will be followed. Once the proper processing time is done, then schedule the transmission of this frame to begin after the processing time. Figure A1.5 shows the flow diagram for TRANSMT Event.

The next event is LAPTRNS which is scheduled by TRNSMT and SNDSFRM events. This event is basically implements the end of transmission and collects statistics for transmitted frame. If this event is scheduled from TRNSMT, then source is transmitting an I-frame, and also if this event is scheduled from SNDSFRM, then source is transmitting a S-frame. If a S-frame was being transmitted then propagation event LAPRCV will be scheduled, but if I-frame was being transmitted, we check if this frame should be suspended because of timer recovery condition or REJ recovery condition. if so do not schedule the propagation event, otherwise schedule the LAPRCV propagation event for this I-frame.

Since this event is scheduled for the end of transmission, then for transmitting the next frame, LAP follows the following logic. First check if any S-frame waiting for transmission, if so then schedule the SNDSFRM Event after the processing delay in order to transmit the waiting S-frame. Next check if source is in retransmit status and has I-frames outstanding for transmission and is not in time recovery condition then immediately schedule TRNSMT Event to continue the retransmission process. If source has no S-frames waiting and is not retransmitting and is not in timer recovery condition and has no I-frames waiting in data link layer for transmission, then request higher layer entity to send I-frame to data link layer, This action is performed by scheduling APPLCLYR Event after appropriate processing time delay. Figure A1.6 shows a flow diagram for LAPTRNS Event.

The next event LAPRCV simulates a destination of a given node. This event is scheduled by LAPTRNS event. This performs two primary functions. First it determines if an I-frame or a S-frame is received. If a S-frame is received, based on the modeling assumption made in chapter five, this frame is considered to be error free(no transmission error). But if it an I-frame is received, then this event randomly determines if this I-frame is in error based on the channel bit error rate and frame size. If this frame contains an error , it will be discarded. The second function performed is that based on the frame type an appropriate part of data link layer will be scheduled after some processing delays.

This event schedules RCVIFRM for error free (no transmission error) I-frame and schedule RCVSFRM for received S-frame. Figure A1.7 shows a flow diagram of LAPRCV Event.

The next event discussed is the RCVIFRM Event. This event simulates a part of destination's data link layer which process the received I-frames. This event is scheduled by LAPRCV Event. When this event receives an I-frame, first checks if this frame is already been received by this node, in another words make sure no duplicate frames are processed. Next check if the received I-frame has a send sequence number equal to receiver's sequence number. If an out of sequence I-frame is received, then proceed to send a REJ frame to the transmitting node if no REJ frames are outstanding. But if receiver receives an in sequence I-frame, then it would update its receive state variable(next I-frame to be received). Remove the acknowledgement from this I-frame and send the data to the application layer by scheduling APPLCLYR Event after the processing delay.

To process the received piggybacked acknowledgement, the RCVIFRM Event calls SNDRCVACK subroutine to perform such operation. In addition SNDRCVACK subroutine decides the method of acknowledging the I-frame received by this node(piggybacking or sending RR S-frame). The decision login for RCVIFRM Event is given in figure A1.8.

The next event id SNDSFRM Event. This event is scheduled from SNDIFRM, TRNSMT , RCVIFRM, and LAPTRNS events and SNDRCVACK Subroutine.. This event simulates the source of a node which transmits only S-frames. A supervisory frame can be a RR frame which is used to acknowledge I-frame(s), a REJ frame which indicates that a receiver has received an out of sequence I-frame, and finally a RR command I-frame for timer recovery condition. Since it is a necessity to send REJ or RR timer recovery condition frames, and it happens that the transmitter might be busy when these supervisory frames are to be transmitted, the source node always stores these supervisory frames in the S-frame buffer before transmission.

If a supervisory frame to be transmitted, the source checks on the type of supervisory frame. If the supervisory frame is a RR frame which acknowledges a received I-frame, then this node would packetizes a S-frame as specified by the LAPD protocol then would hand over this S-frame to lower part of data link layer for transmission. But if the supervisory frame is a REJ frame or RR timer recovery condition, then the source would search the S-frame buffer for a S-frame which matches the source's ID. When this frame is found, it will be packetized and will be sent to lower part of data link layer for transmission. Figure A1.9 shows a flow diagram of SNDSFRM Event.

The next event is RCVSFRM Event. This event implements a node's receiver in which processes the received S-frames and based on the type of S-frame received different procedures are followed. If a RR frame is received, the acknowledgement is processed by first calling SRCH subroutine to stop the timer for the acknowledged I-frame, and then call RMVACKFRM subroutine to remove the acknowledged I-frame form the retransmission buffer.

If the received S-frame is a REJ frame, first we check if the source is in timer recovery condition, if so then the REJ frame will be ignored. But if source is not in timer recovery condition, the REJ frame is processed immediately. The node that receives a REJ frame should retransmit the requested I-frame and any other frames outstanding. However before retransmission certain procedures are followed. First suspend the transmission of any I-frame in progress. Second call SRCH subroutine to stop the timer for all transmitted I-frames(outstanding I-frames). Note that the I-frames transmitted after the out of sequence packet are discarded by the receiving node, therefore the transmitting node has to retransmit the entire window. Then call RMVACKFRM to remove the acknowledged I-frames from the retransmission buffer. Then schedule the REXMIT Event to start the procedure of retransmission.

The third type of S-frame that could be received by a node is the RR timer recovery condition Command/Response frame. If a node receives a command RR S-frame, then this node should immediately send a response RR S-frame to the transmitting node as it specified

by LAPD protocol. But if a node receives response RR S-frame, then the TIMEOUT Event will be scheduled to process the received RR timer recovery condition S-frame. Figure A1.10 shows a flow diagram of RCVSFRM Event.

The next event discussed is TIMEOUT Event. This event is scheduled from RCVSFRM and TRNSMT events. This event implements the timer recovery procedures specified by LAPD protocol. This event is invoked when a transmitting node does not receive an acknowledgement for a transmitted I-frame before the timeout period expires. Since it can happen that an I-frame has been received correctly, but the acknowledgement has been sent and not received by the transmitting node , a RR Command S-frame is sent to the other node prior to retransmission to avoid a duplication of the I-frame already sent. Therefore when this event is scheduled, first it checks if the received frame is an I-frame or a response timer recovery condition S-frame. If it is an I-frame then checks whether or not the source node is already in timer recovery condition, if not then start the procedure to send a RR command S-frame to the receiving node. But before doing so, if any I-frames are in progress then suspend the transmission of this frame and call SRCH subroutine to stop the timer for the suspended I-frame. Then begin to send A RR command frame. If a node is in timer recovery condition, the expiration of timer for the transmitted I-frames will be ignored.

The TIMEOUT Event also is scheduled from RCVSFRM Event. This is the case when this node is in timer recovery condition and receives a response RR S-frame for a previously transmitted command S-frame. When TIMEOUT Event receives a S-frame, it process the acknowledgement contained in S-frame by calling SRCH and RMVACKFRM subroutines to stop the timer and remove the I-frame(s) from the retransmission buffer for the acknowledged I-frame(s) respectively. Then based on the acknowledgement sequence number received, this node checks if the other node's receiver sequence number is less than or equal to the transmitting node send sequence number, then transmit all the outstanding unacknowledged I-frames by scheduling the REXMIT Event. But if the other node's receive sequence number is greater than send sequence number of transmitting node , then the timeout has occurred in error free transmission environment because of short timeout period with respect to acknowledgement delay and channel load, and the node take would take any actions. The TIMEOUT Event logical flow diagram is given in figure A1.11.

The next event is REXMIT Event. This event is scheduled from TIMEOUT and RCVSFRM Events. This event is invoked when a request for retransmission is needed. The retransmission occurs when an I-frame has been lost or discarded because of transmission errors or not having the correct sequence number. The retransmission mechanism ensures that end user always work with error free data. The REXMIT event logic starts by setting the node status in retransmit, then based on the information given about the requested

frame sequence number, this node would search the retransmission buffer to locate the entry that has the same sequence number as the requested frame sequence number. Then source would reset its send sequence number to the requested I-frame sequence number. Next if transmitter is idle, then the retransmission process would start immediately, otherwise as soon as source returns idle, start the retransmission of outstanding I-frames. The flow diagram of REXMIT Event is given in figure A1.12.

The last event implemented is APPLCLYR Event. This event is scheduled by the following events and subroutines: LAPTRNS, TRNSMT, REXMIT, TIMEOUT, RCVIFRM, and RMVACKFRM. This event simulates an end user in which receives commands a data packets from data link layer and also sends data packets to data link layer. Based on the commands given by data link layer, the end user sends I-frames to data link layer by releasing packets from the wait node, or stop sending I-frames to data link layer by not allocating any resources to the wait node. Application layer also receives error free data packets from data link layer which in turn sent from other node on the network. This event also collects statistics on average end-to-end delay, and decrement number of packets in the system which enables us to measure average number of packets in the system. Figure A1.13 shows a flow diagram of APPLCLYR Event.

So far we described the events that implemented the major portion of LAPD protocol. Now discuss the subroutines involved in implementation of LAPD protocol.

The first subroutine to be considered is SNDRCVACK. This subroutine is called from RCVIFRM Event. This subroutine takes the necessary action for receiving and sending acknowledgements when node is in normal I-frame transmission and retransmission status. When a node receives an error free I-frame, it contains a piggybacked acknowledgement which is used to acknowledge all I-frames transmitted by this node. The received acknowledgement is processed by first calling SRCH subroutine to stop timer for the acknowledged I-frame(s)(remove timeout event from the event calendar). Second call RMVACKFRM subroutine to remove the acknowledged I-frame(s) from the retransmission buffer.

In addition to receiving acknowledgements, a node also sends acknowledgements either in form of piggybacking or sending a supervisory frame. The procedure of sending acknowledgements slightly differs based on the node's status. A source node status is either in normal I-frame transmission, I-frame retransmission, and timer recovery condition. If a node is in timer recovery condition, no acknowledgements will be sent since the timer recovery condition itself is a form of acknowledgement and sending another S-frame becomes redundant. Now describe the procedure for sending acknowledgements in normal I-frame transmission and I-frame retransmission.

In normal I-frame transmission in order for a node to acknowledge a received I-frame using a supervisory frame(not piggybacking) the following conditions must me met:

1. The received I-frame must be error free and in sequence

2. There should not be any I-frames for transmission

3. The source node should not be already transmitting. This would indicate either a piggyback acknowledgement is being sent or a supervisory frame is being sent.

If these conditions are met then send a supervisory frame to acknowledge a received I-frame. Otherwise use piggybacking as the method for acknowledgement. It should be noted that at low loads, there would be a greater chance to use a supervisory frame as the form of acknowledgement.

On the other hand if a node is in retransmit status certain conditions must be met before using a supervisory frame to acknowledge an I-frame. These conditions are as follows:

1. The received I-frame must be error free

2. If the node is idle. Then this means that this node has retransmitted all the outstanding I-frame and is waiting for acknowledgement from the other node for the transmitted I-frames. In this case send a supervisory frame to acknowledge the received I-frame.

3. If the transmitter is busy, then check if this is the last outstanding I-frame being transmitted. If so then as soon as transmitter returns idle, transmit a S-frame to acknowledge the received I-frame. It should be noted that when node is in retransmit status and has transmitted all outstanding I-frames, it would not transmit and more I-frames until the acknowledgement is received for the transmitted I-frame(s). Note that when no I-frames are sent, the only form of acknowledgement is by using supervisory frames. If transmitter is busy and have I-frames outstanding then no supervisory frame will be sent since the subsequent I-frames transmitted would acknowledge the received I-frame (piggybacking).

Figure A1.14 shows a flow diagram of SNDRCVACK Subroutine.

The next user subroutine that we will consider is RMVACKFRM Subroutine. This subroutine is called from SNDRCVACK subroutine, RCVSFRM event, and TIMEOUT event. This subroutine implements the logic of removing acknowledged I-frames from the retransmission buffer. When this subroutine is called, the sequence number of the received acknowledgement will be given to it. Then the retransmission buffer is searched to remove I-frame(s) which have sequence number(s) less than or equal to acknowledgement sequence number. This routine also collects statistics on buffer holding time for the I-frames. In addition, RMVACKFRM subroutine checks if

its source node is in retransmit status and all the retransmitted I-frames has been acknowledged, then it would reset the retransmission variables and schedules the APPLCLYR Event to send more I-frames to data link layer. Figure A1.15 shows a flow diagram of RMVACKFRM Subroutine.

The next subroutine is SRCH Subroutine. This subroutine is called from SNDRCVACK subroutine, RCVSFRM Event, and TIMEOUT Event. This subroutine implements the logic of stopping the timer for transmitted I-frames based on some given conditions. There are tree conditions that are considered : I-frames acknowledgements, REJ recovery condition, and timer recovery condition.

Based on the given condition, search the event calendar and locate the timeout event(s) that matches the given condition. If the given condition is for acknowledgement of I-frames, then the event calendar is searched for the timeout events that are belong to this node and has I-frame sequence number less than or equal to the received acknowledgement sequence number and remove such events from the event calendar. In reality this is similar to stopping the timer for the acknowledged I-frame(s).

If SRCH subroutine is called because of a REJ recovery condition, then the procedure is to stop the timer for all outstanding transmitted I-frames from this node. Search the event calendar for the timeout events that have the same source identity and remove them from the event calendar. The reason is that when a

node is in REJ recovery condition, it would retransmit all the outstanding I-frames.

The last condition that SRCH subroutine could be called for it is the timer recovery condition. If a node is going in timer recovery condition and if at the same time transmitting an I-frame, the transmission of I-frame will be suspended and in result of this the timer should be stopped. To stop the timer the TIMEOUT Event calls SRCH Subroutine to remove the timeout event from the event calendar for the suspended I-frame. This logic is implemented by searching the event calendar for the timeout event based on the node's source identity and the sequence number of suspended I-frame. The logic flow diagram for SRCH Subroutine is given in figure A1.16.

This concludes our discussion on the routines that make up the EVENT file. The remaining routines in the Discrete Event Model are the OTPUTLAP routine, ALLOCTRSC function, and USERF function. The OTPUTLAP routine is mechanism that allows us to report statistics that are not repeated in the SLAM SUMMARY REPORT. This routine is shown at the end of this Appendix.

The ALLOCTRSC function(ALLOC - SLAM[2]) is a standard FORTRAN user function. This function is called from SLAM Network Model every time an entity arrives to the wait node. This function basically perform the resource allocation for a given node. For resource allocation this function considers several conditions: first, node must have resource available. Second, node should not

be in timer recovery condition or retransmit status. Third the window must not be at its maximum limit. If these conditions are met, then a unit of resource will be seized. On the other hand one data packet will be branched from Network Model to Discrete Event Model. Figure A1.17 shows a flow diagram of ALLOCTRSC Function.

The next function implemented is USERF Function. USERF is used to implement packet length restrictions on packets created in the SLAM Network Model. Here, packet lengths are calculated from one the two rules or modes. In each case, the packet length is calculated and the overhead bits are added. Then minimum and maximum packet length restrictions are enforced. These two modes correspond to 1) fixed length packets, 2) exponential length packets. Figure A1.17 shows a flow diagram of USERF Function.

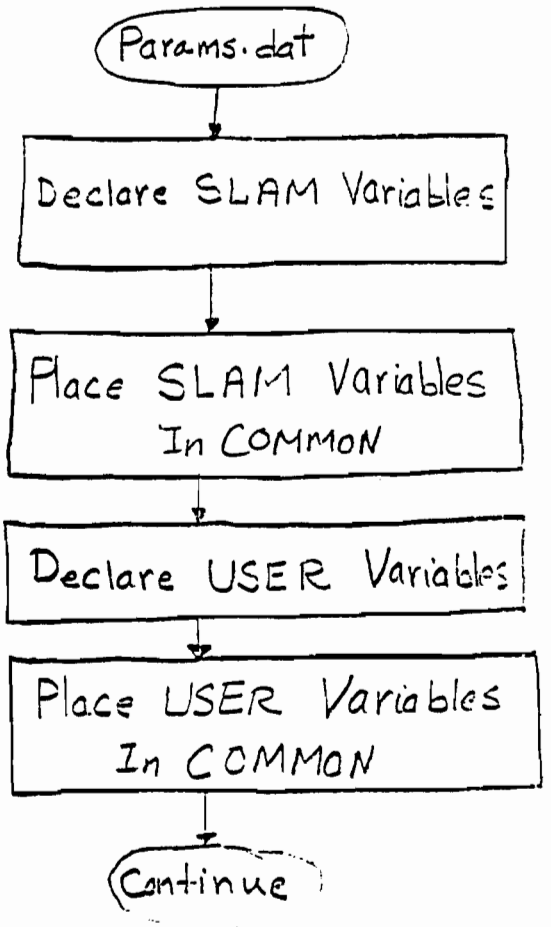


Figure A1.2 PARAMS.DAT Flow Diagram

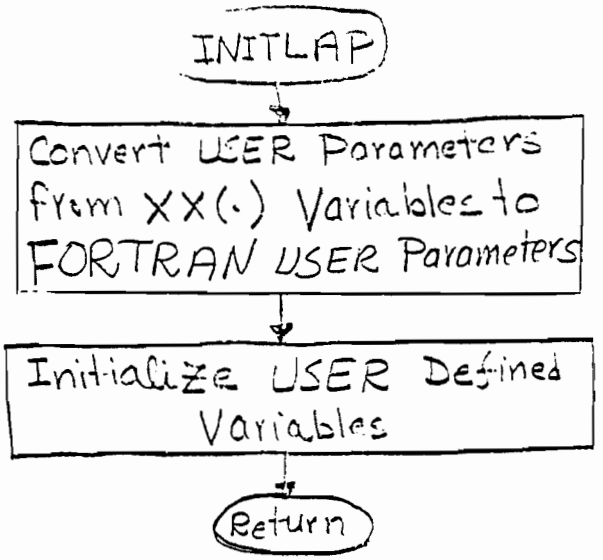


Figure A1.3 INITLAP Subroutine Flow Diagram

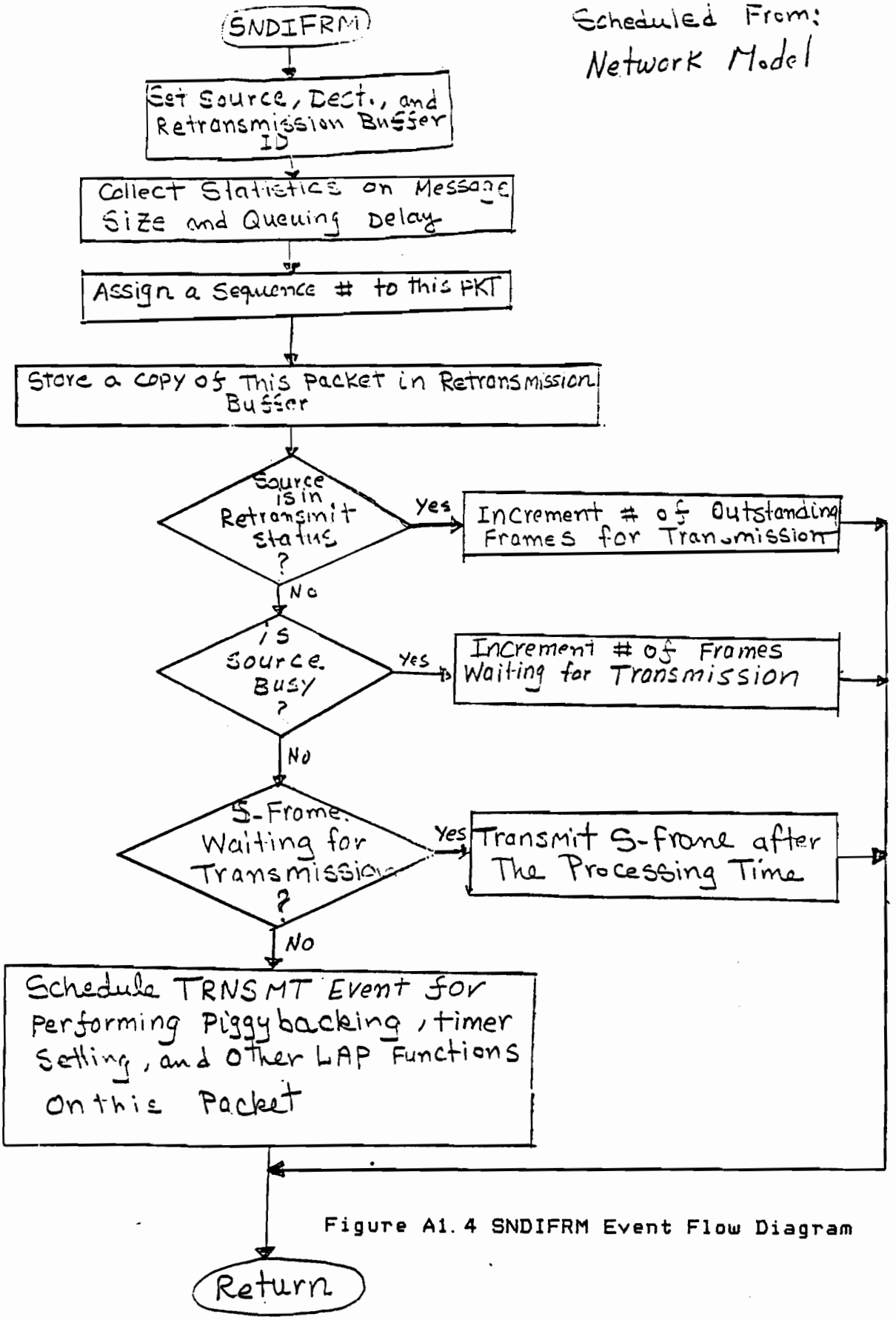


Figure A1.4 SNDIFRM Event Flow Diagram

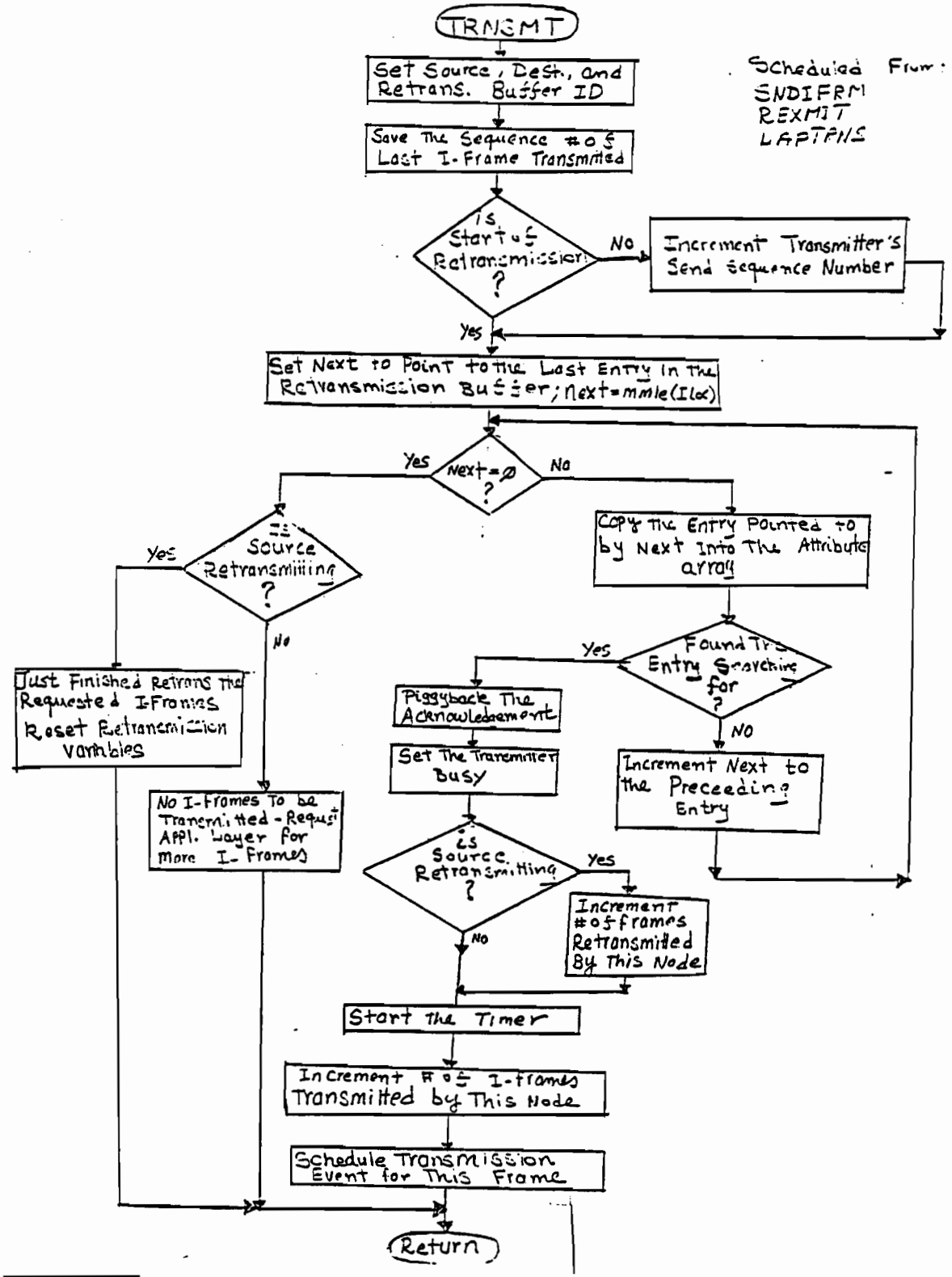


Figure A1.5 TRNSMT Event Flow Diagram

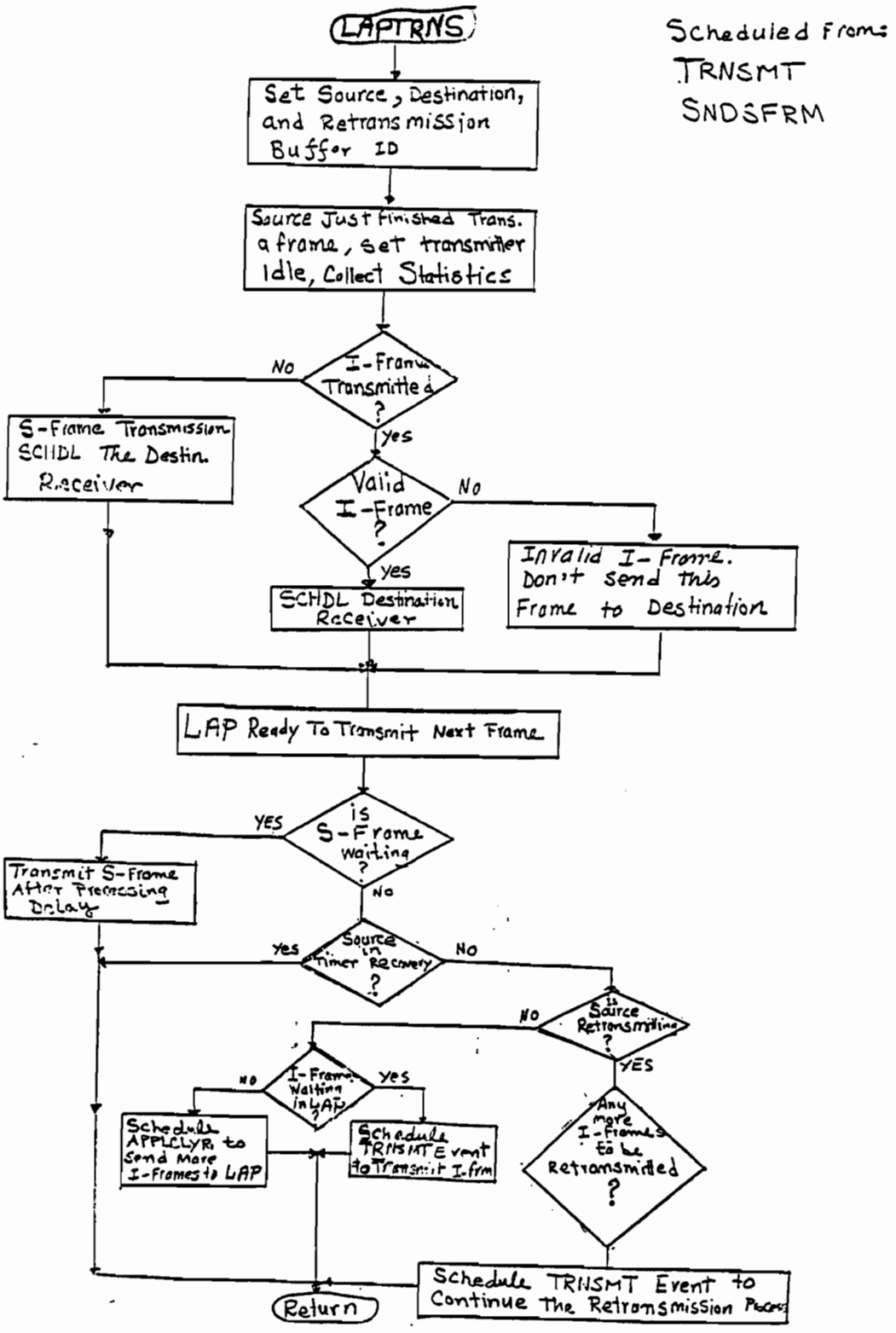


Figure A1.6 LAPTRNS Event Flow Diagram

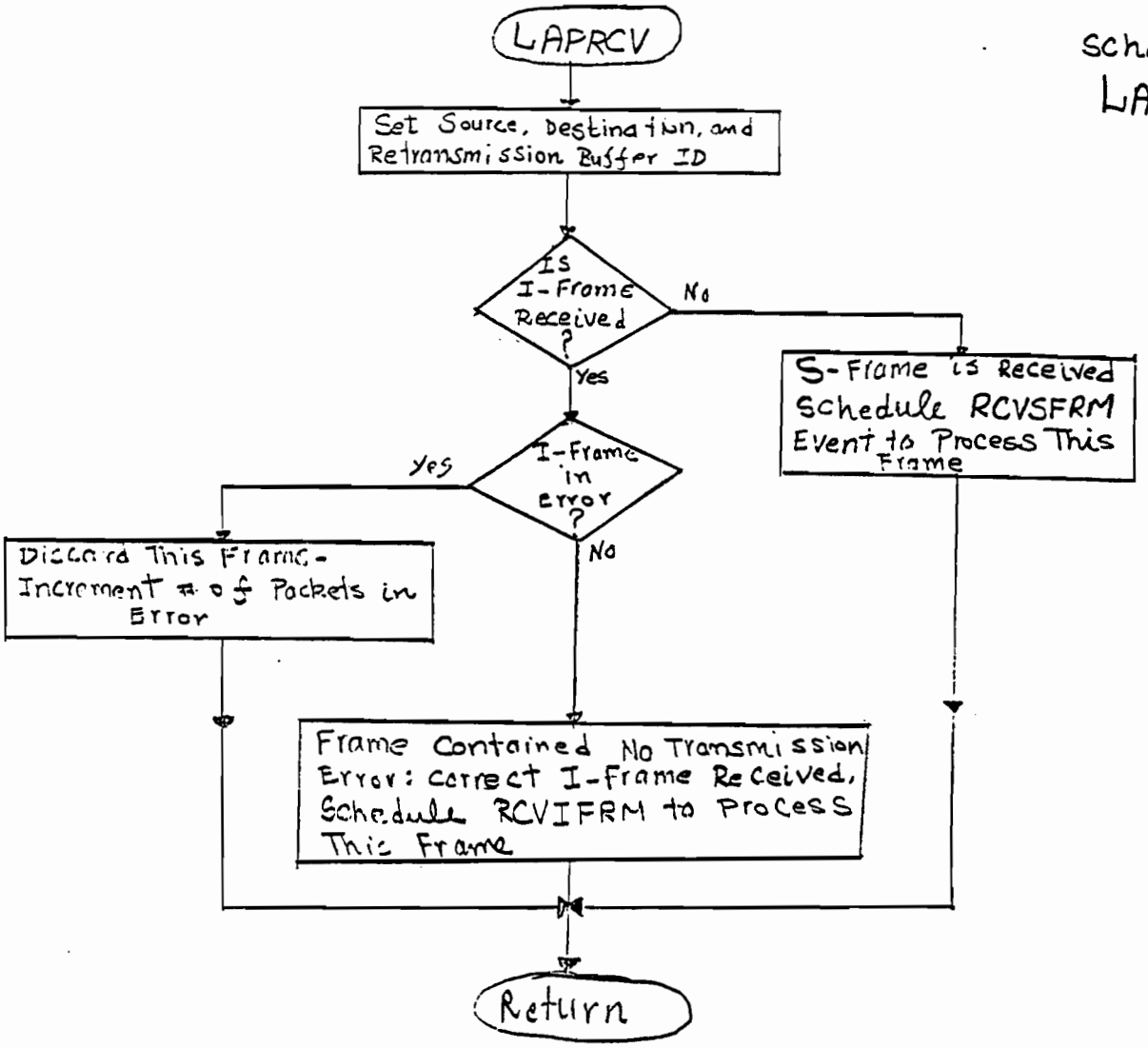


Figure A1.7 LAPRCV Event Flow Diagram

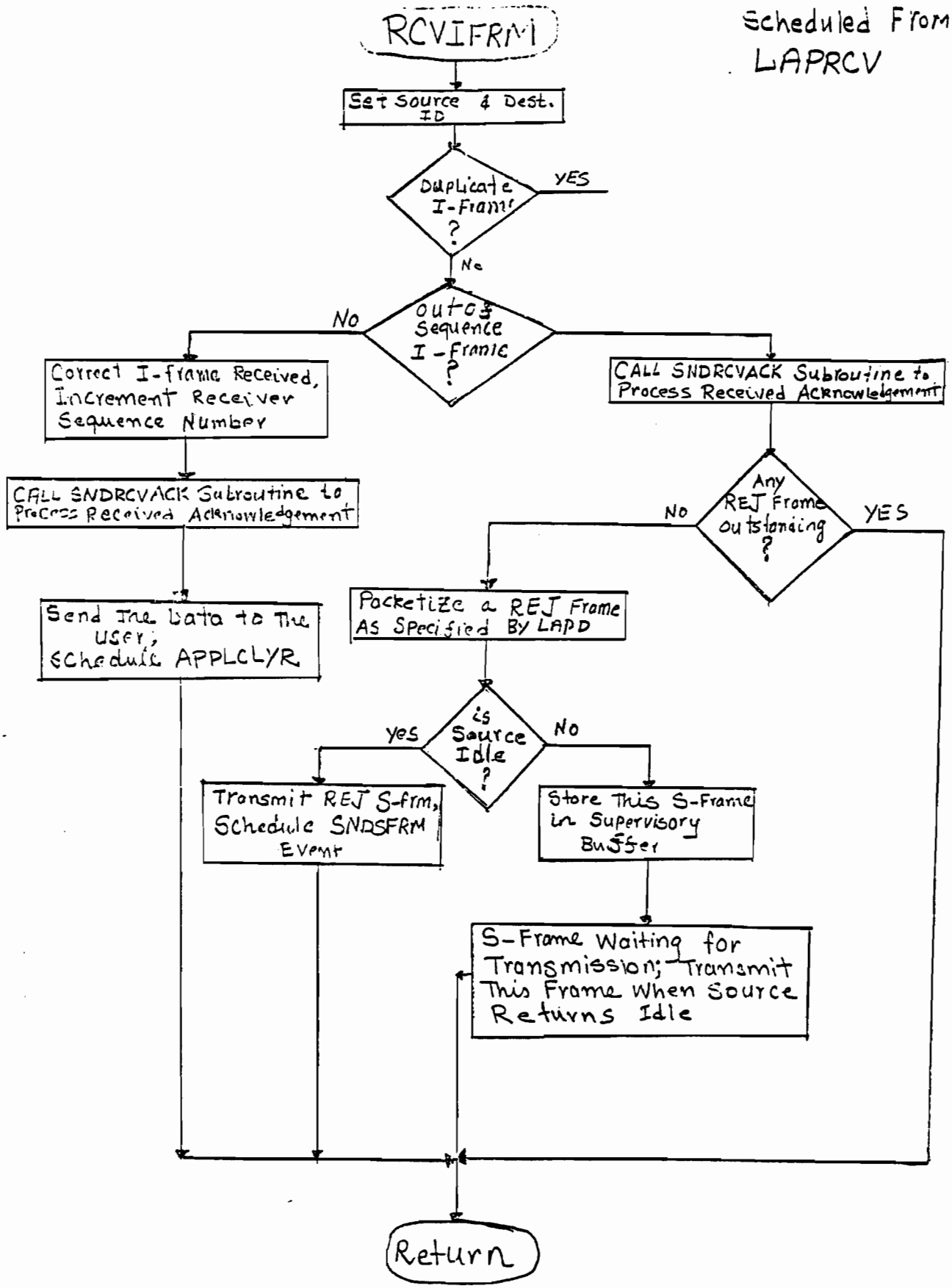


Figure A1.8 RCVIFRM Event Flow Diagram

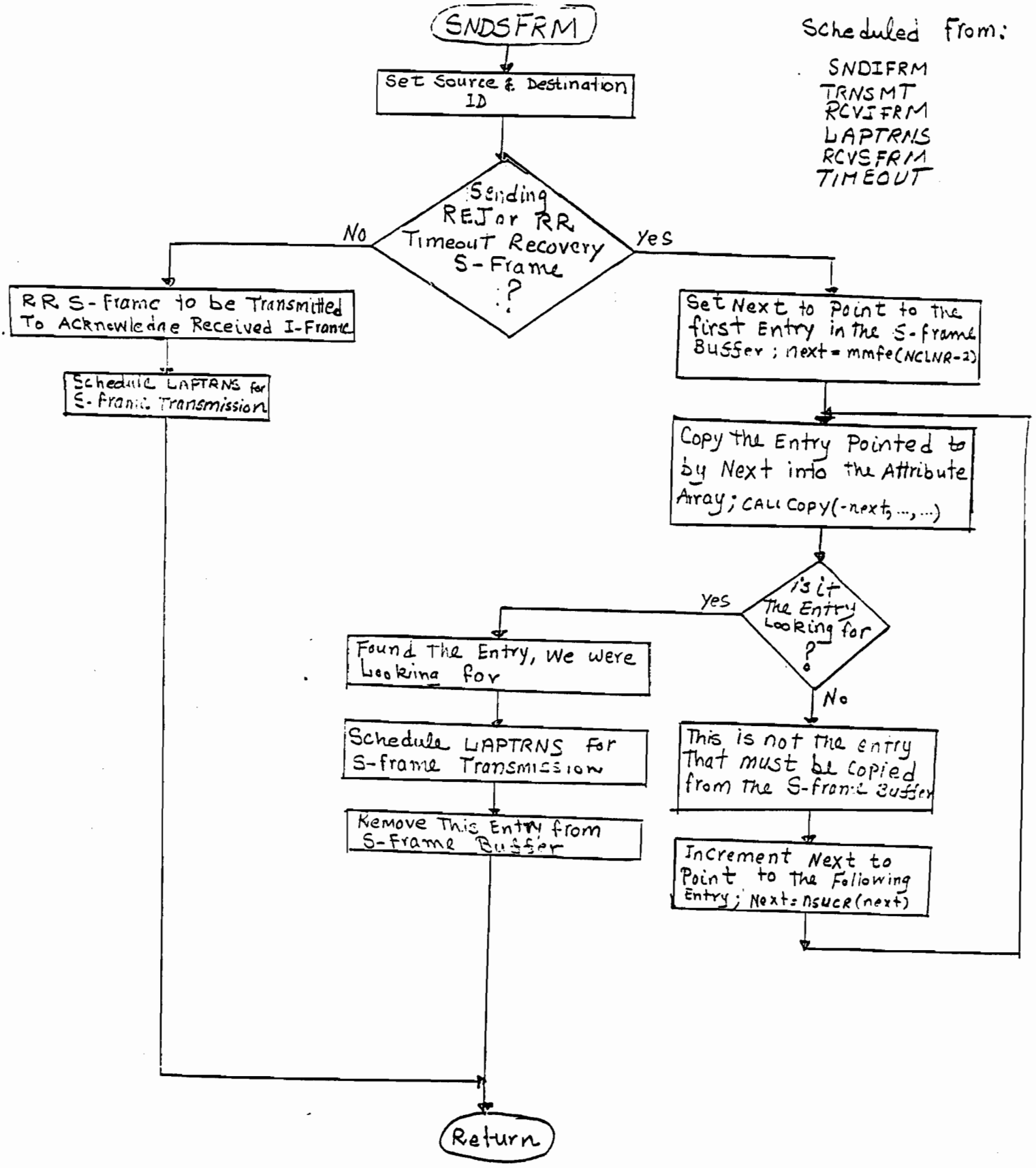


Figure A1.9 SNDSFRM Event Flow Diagram

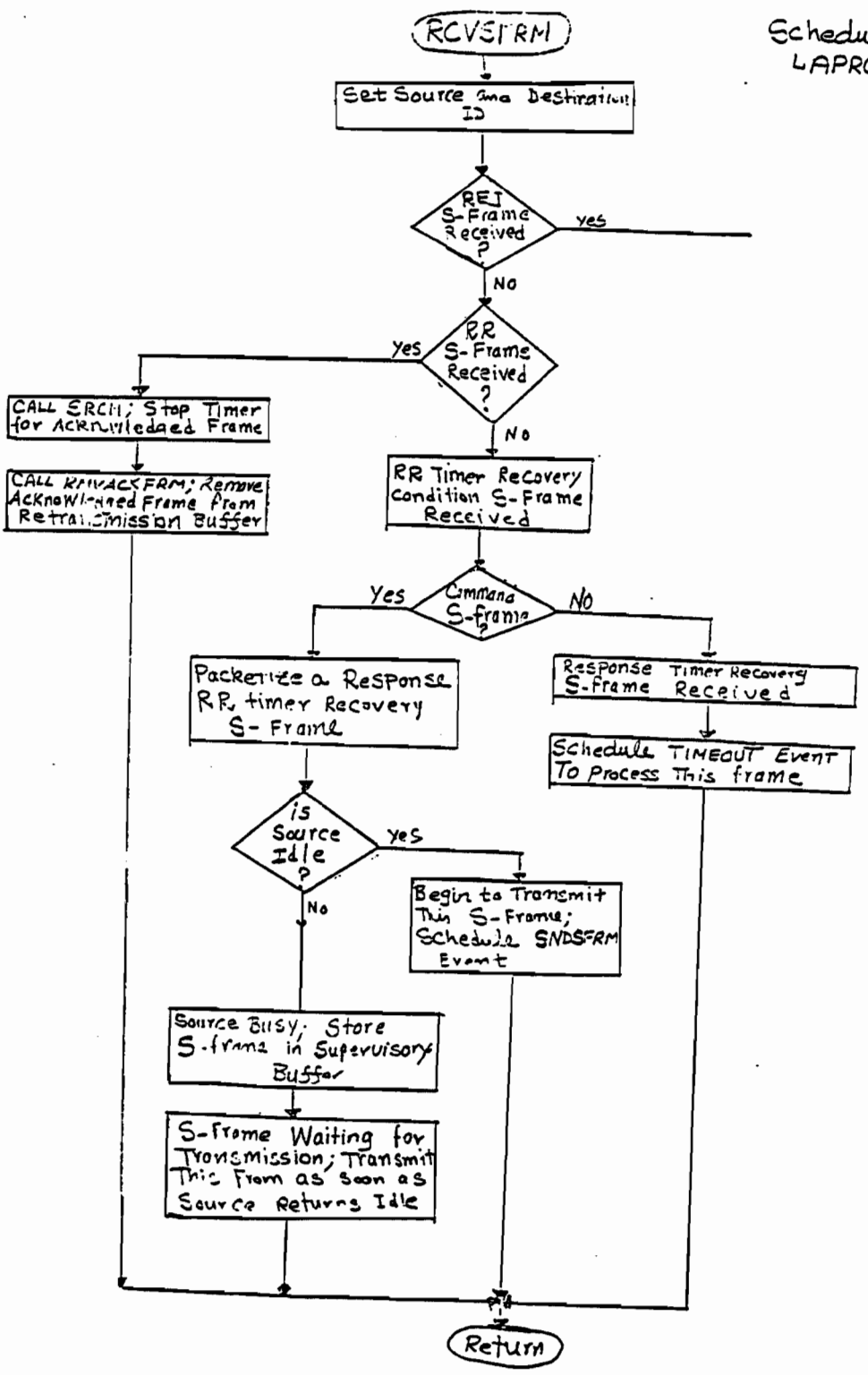


Figure A1.10 RCVSFRM Event Flow Diagram

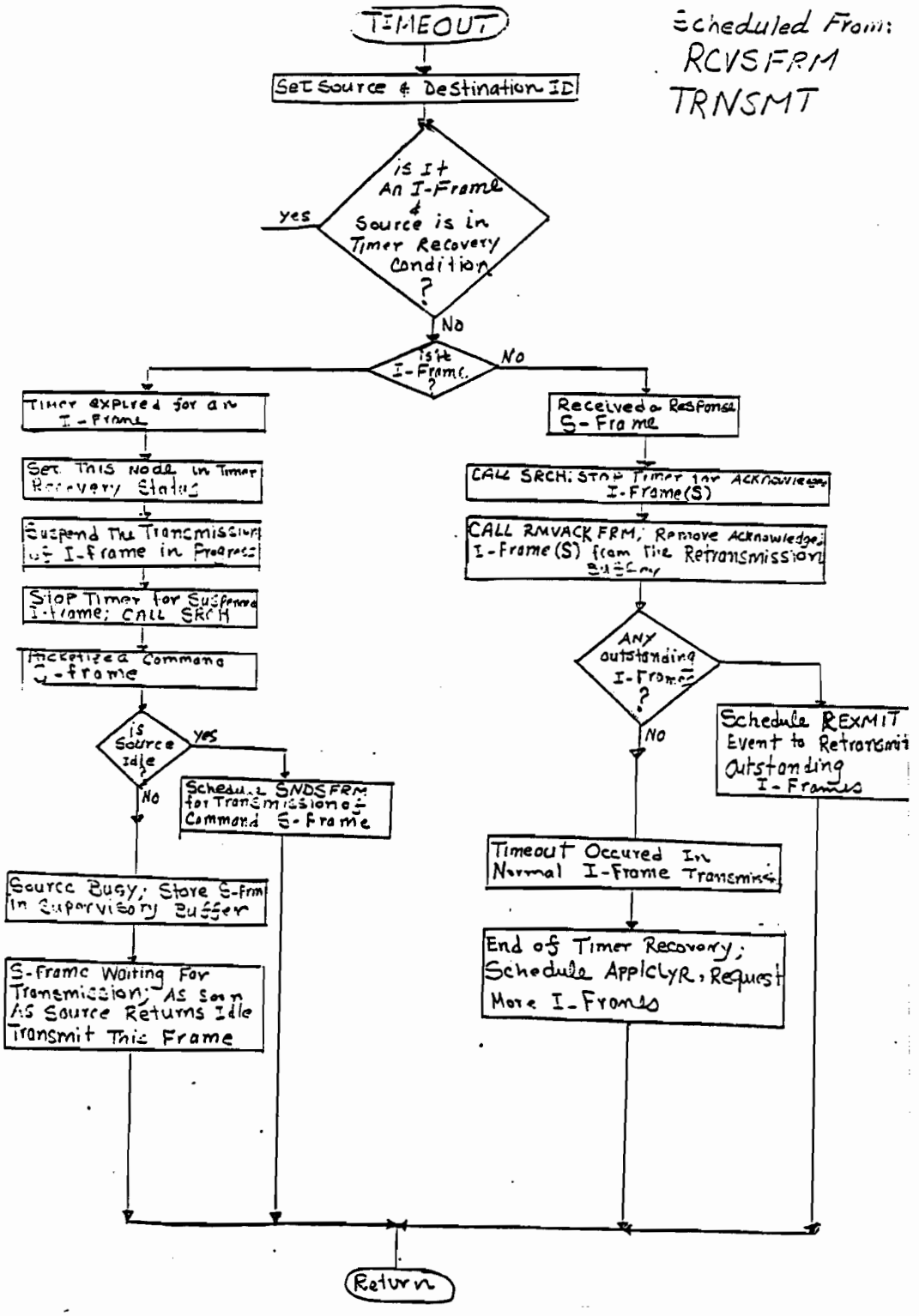


Figure A1.11 TIMEOUT Event Flow Diagram

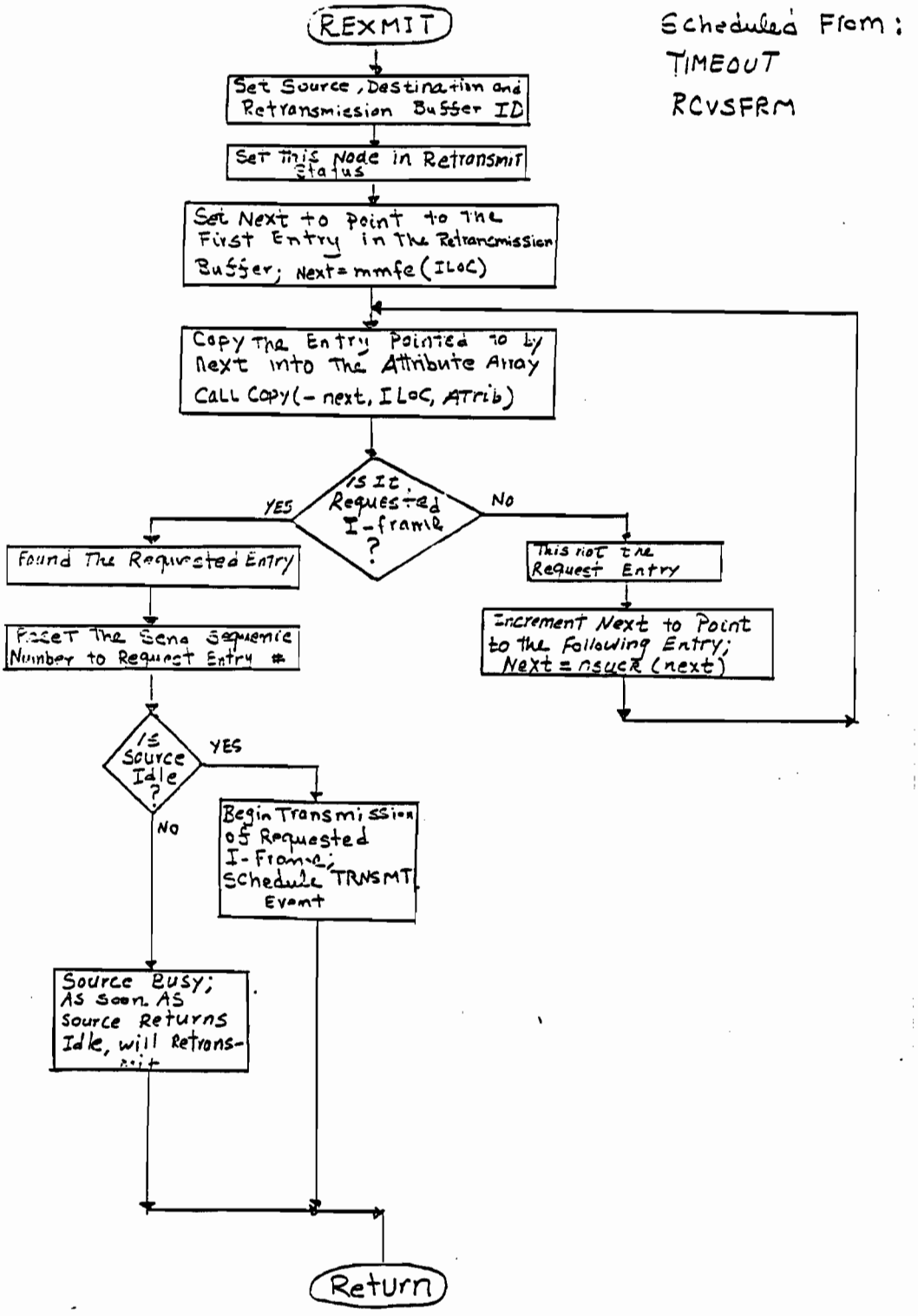


Figure A1.12 REXMIT Event Flow Diagram

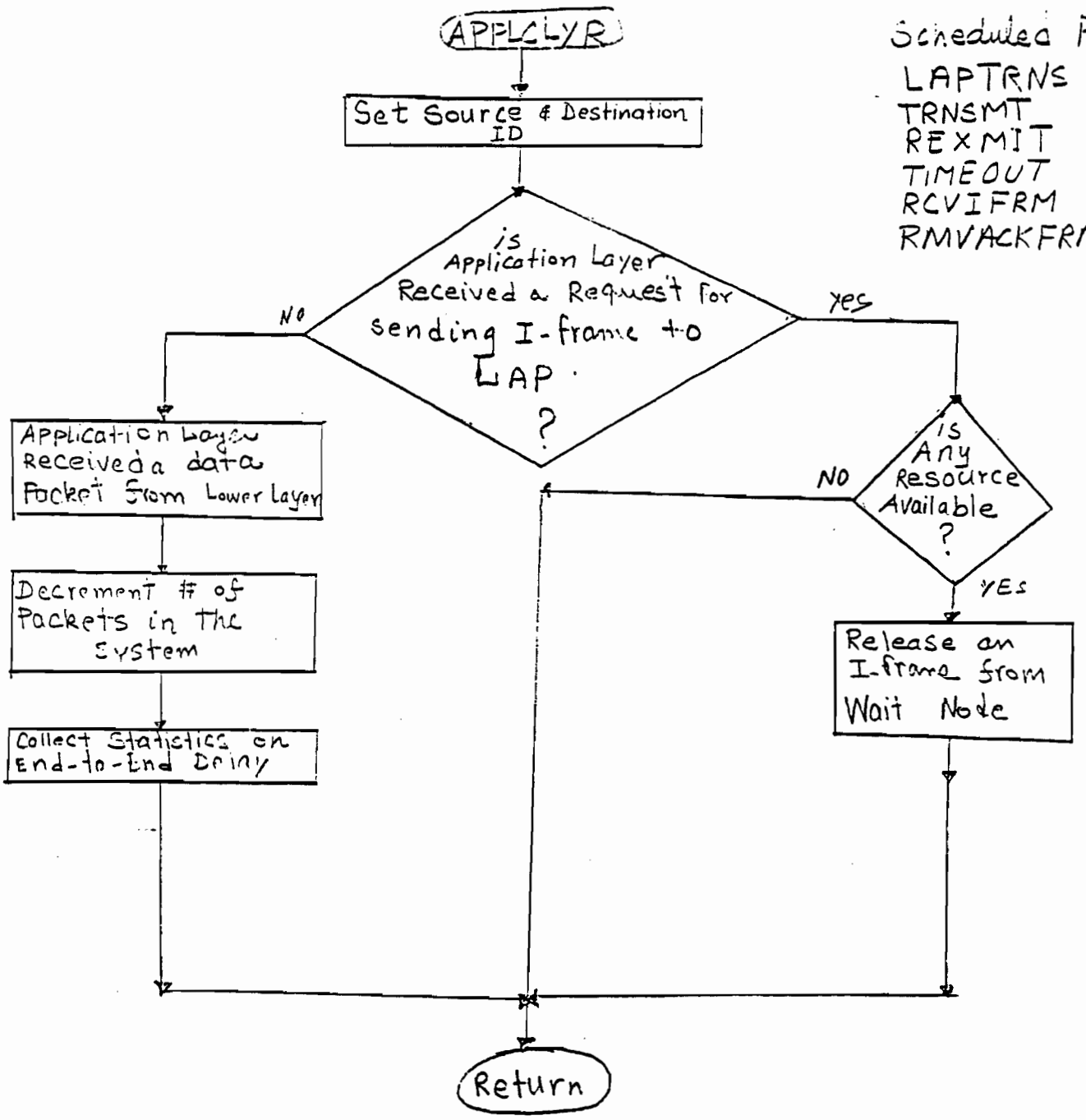


Figure A1.13 APPLCLYR Event Flow Diagram

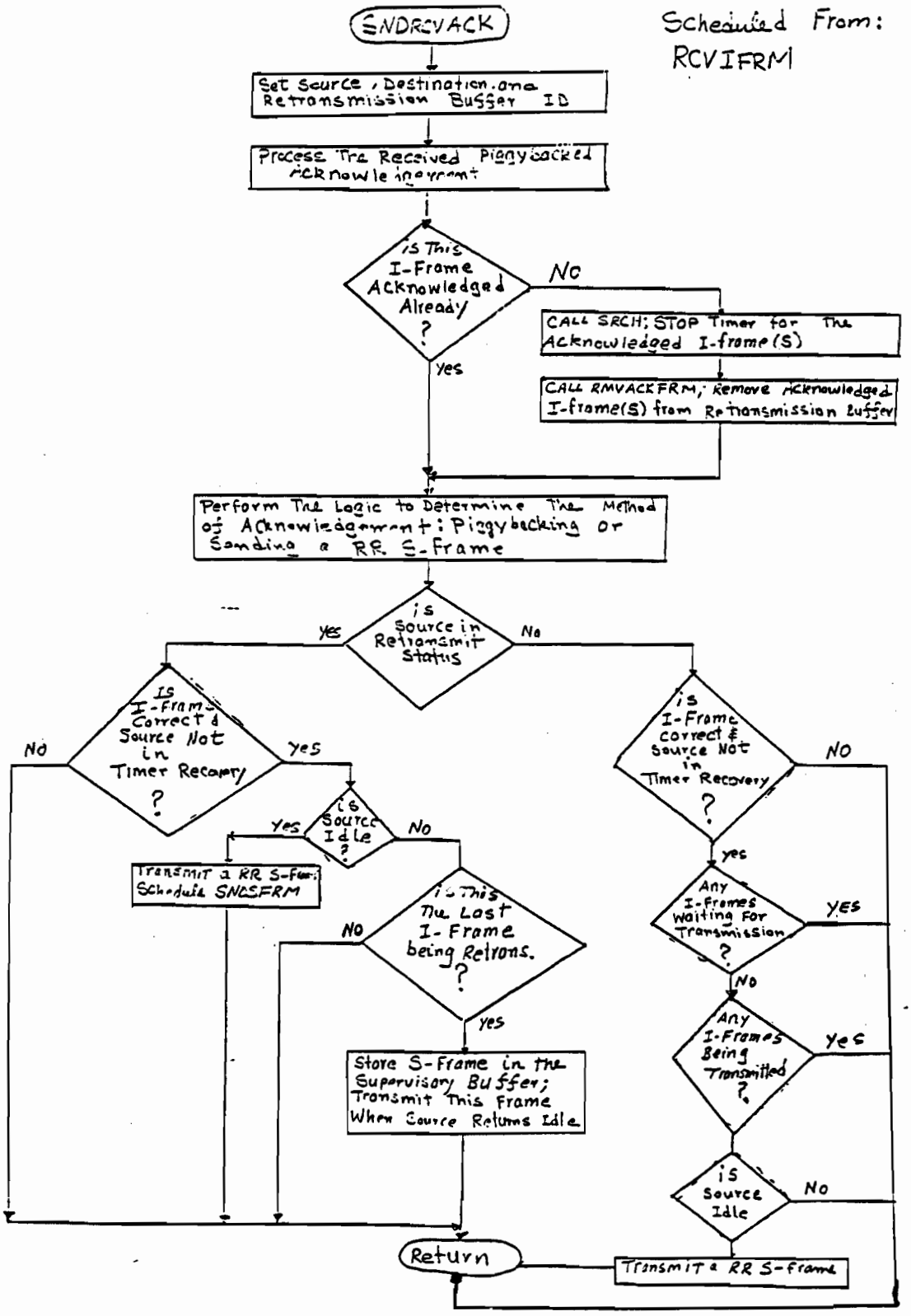


Figure A1.14 SNDRCVACK Subroutine Flow Diagram

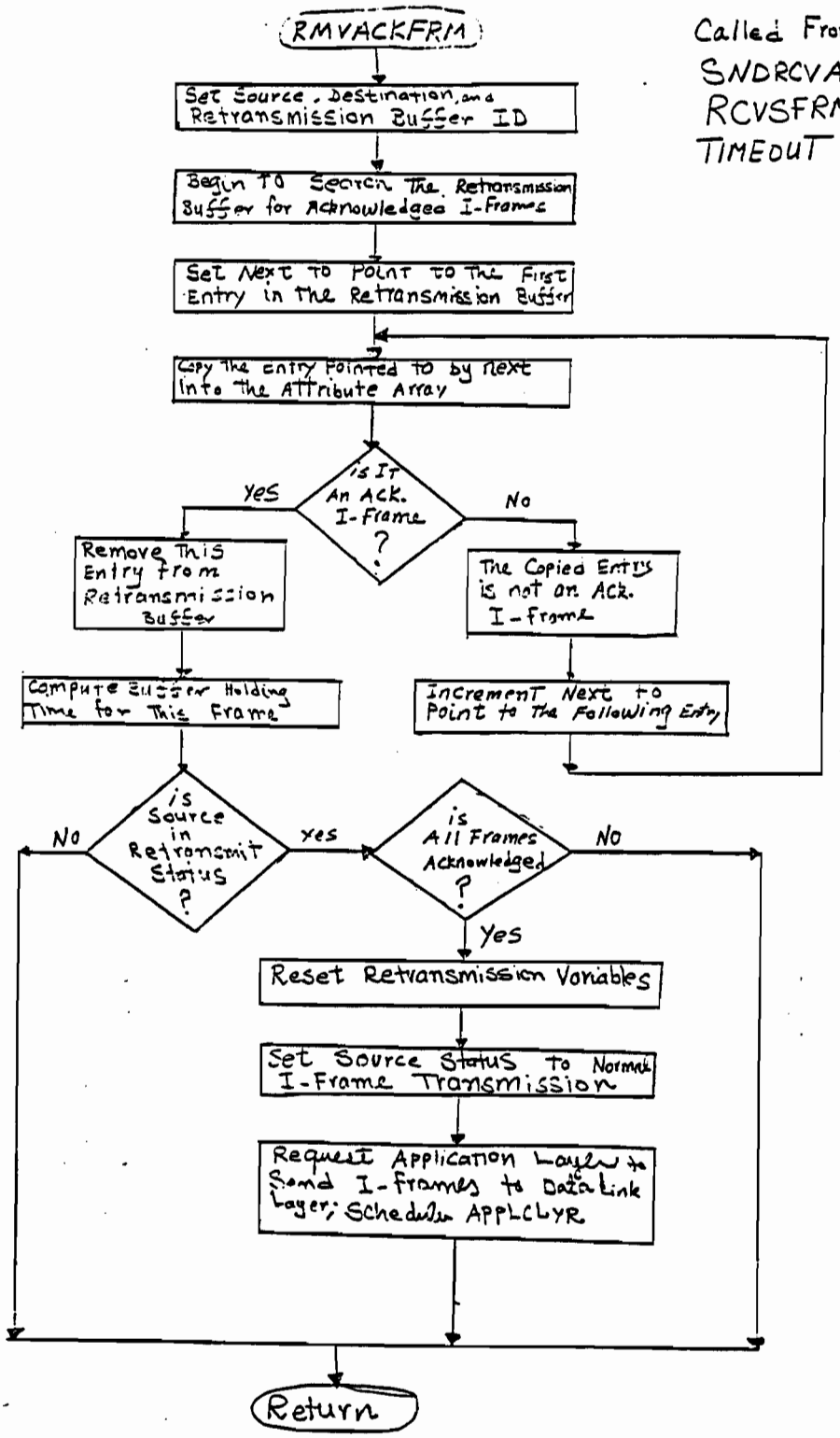


Figure A1.15 RMVACKFRM Subroutine Flow Diagram

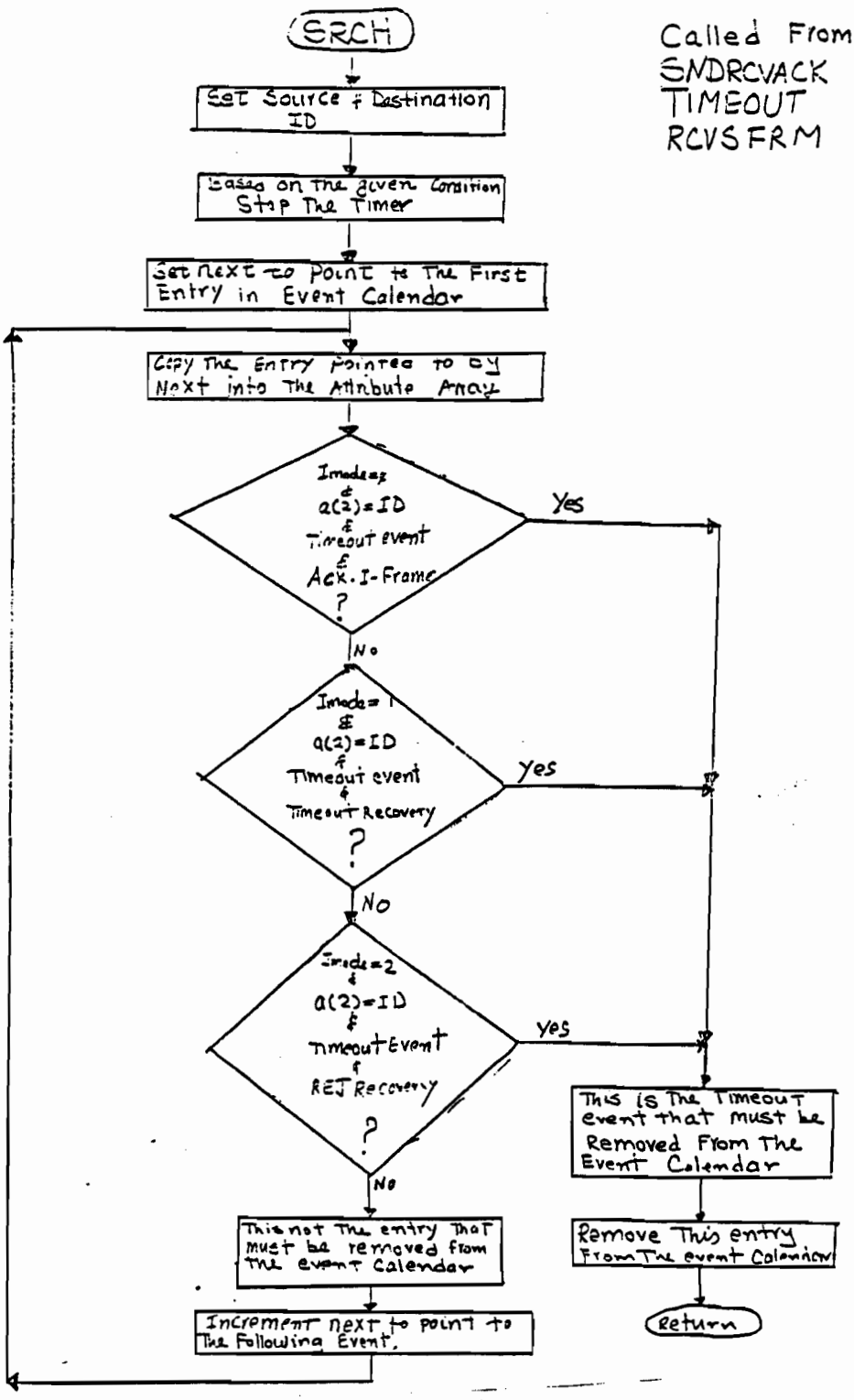


Figure A1.16 SRCH Subroutine Flow Diagram

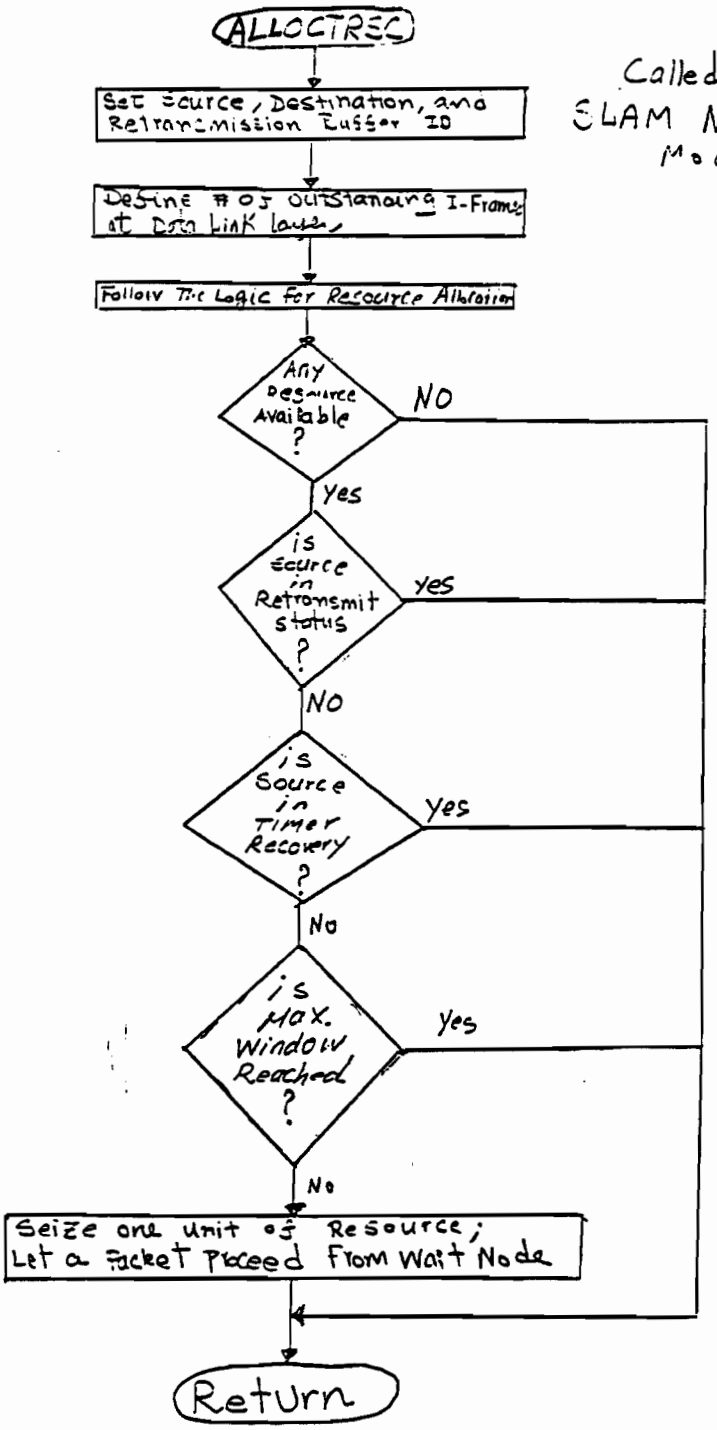


Figure A1.17 ALLOCTRSC Function Flow Diagram

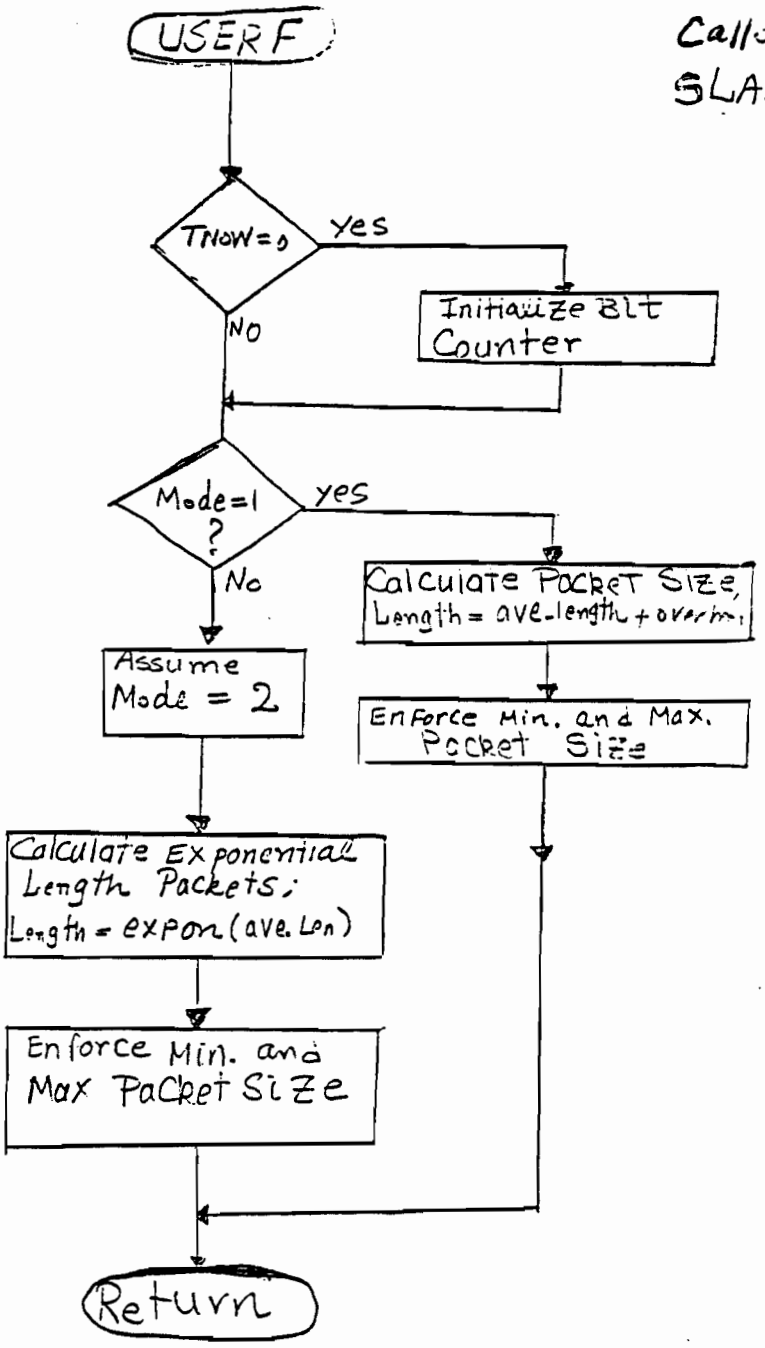


Figure A1.18 USERF Function Flow Diagram

References :

1. B.Wood,' A Performance Evaluation of Local Area Networks', Technical Report,TISL-6731-3,January 1986

2. A.B.Pritsker,' Introduction to Simulation and SLAM II ',Halsted Press Inc.,1984

This part of Appendix 1 contains the routines that form the discrete event portion of the TISL LAPD simulation system. The purpose of this section is to show the exact code that is executed to effect the simulation results. A detailed description of these routines, including flow diagrams, is found first portion of this Appendix.

```
C ------------------------------------------------------------------
C        TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C FILENAME: Params.dat
C ------------------------------------------------------------------
C PURPOSE :
C
C         This is a section of code that is inserted at the beginning
C         each of the FORTRAN subroutines that are part of the SLAM
C         simulation through the use of the statement 'include params.dat'.
C         The purpose of this section if code is as follows:
C         +  Define all user and SLAM variables
C         +  Establish common blocks that contain the user/SLAM varaibles
C
C
C ------------------------------------------------------------------------
c
       implicit none
c
c define slam variables:
       integer ii,mfa,mstop,nclnr,ncrdr,nprnt,nnrun,nnset,ntape
       real    atrib,dd,ddl,dtnow,ss,ssl,tnext
     1         ,tnow,xx
c
      common/scom1/ atrib(100),dd(100),ddl(100),dtnow,ii,mfa,mstop,nclnr
     1         ,ncrdr,nprnt,nnrun,nnset,ntape,ss(100),ssl(100)
     2         ,tnext,tnow,xx(200)
c
c define user variables:
c
c
c lapd user variables:
      integer ndsts,irejsts,itmoutcnt,irxmtcnt,icount,iotsqcnt
     2,idmgpktcnt,iwndsz,sfrm,pktcnt,rcvsvr,isnd,irxsts
     3,totpktcnt,ipkterr,ichksts,jwndrt,itmot,iakvar,ktmp,jsnd
     4,krcv,ifrm,jfrm,infobts,ircvcnt,jchksts,rrfrmcnt,sfrmcnt
     5,bfrcnt,iabrtpkt,ipktrej,ilstpkt,rrsts
c
      real trctm,biterr,blkerr,xohbit,maxsta,maxsz,minsz,rate
c
      common/iucom/ xohbit,rate,maxsz,minsz,biterr,blkerr,trctm
     1      ,maxsta,ndsts(25),irejsts(25),itmoutcnt(25),idmgpktcnt(25)
     2      ,irxmtcnt(25),icount(25),iotsqcnt(25),iwndsz,sfrm(25)
     3      ,pktcnt(25),rcvsvr(25),isnd(25),irxsts(25)
     +      ,totpktcnt(25),ipkterr(25),ichksts(25),jwndrt(25),jsnd(25)
     +      ,itmot(25),iakvar(25),ktmp(25),krcv(25)
     +      ,ifrm(25),jfrm(25),ircvcnt(25),jchksts(25),infobts(25)
     +      ,bfrcnt(25),rrfrmcnt(25),sfrmcnt(25),rrsts(25)
     +      ,ilstpkt(25),iabrtpkt(25),ipktrej(25)

c ------------------------------------------------------------------------
c
c                       End of PARAMS.DAT
c
c ------------------------------------------------------------------------
```

```
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: intlc       AUTHOR: M.Kashefi          DATE: 7-15-86
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------
C PURPOSE :     Initialize SLAM variables and user defined variables in
C               TISL LAP-D simulation.
C ------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !    DESCRIPTION
C ------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------
C                 ! initlap - initialize LAPD variables
C ------------------------------------------------------------------
      subroutine intlc
c
c The following routine will initialize all the variables in the lapd model
      call initlap
      return
      end
```

```
C ------------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------------
C PROGRAM NAME: initlap.for       AUTHOR: M.Kashefi         DATE: 12-15-85
C ------------------------------------------------------------------------
C VERSION NUMBER : 1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------------
C PURPOSE :     Initialize SLAM variables and user defined variables in
C               TISL LAP-D simulation.
C ------------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C ------------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C params.dat      ! file  !      !        ! Declares variables and
C                 !       !      !        ! establishes common block.
C ------------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------------
C                 ! (none)
C ------------------------------------------------------------------------
      SUBROUTINE INITLAP
C
      INCLUDE 'PARAMS.DAT'
C
      INTEGER I
C
      DO 5 I = 2,20
         ATRIB(I) = 0
    5 CONTINUE
C
C     XX(26) = MAXSTA = NUMBER OF NODES IN THE NETWORK
      MAXSTA = XX(26)
C
C     XX(75) = RATE = LINE RATE IN Mbps
      RATE = XX(75)
C
C     XX(81) = MAXSZ = MAX PACKET SIZE WITH OVERHEAD
      MAXSZ = XX(81)
C
C     XX(82) = MINSZ = MIN PACKET SIZE WITH OVERHEAD
      MINSZ = XX(82)
C
C     XX(83) = XOHBIT = NUMBER OF OVERHEAD BITS
      XOHBIT = XX(83)
C
C     XX(110) = IWNDSZ = MAXIMUM NUMBER OF OUTSTANDING FRAMES(WINDOW SIZE)
      IWNDSZ = IFIX(XX(110))
C
C
C     XX(113) = BITERR = PROBABILITY THAT A BIT BEING IN ERROR
      BITERR = XX(113)
C
C     XX(114) = TRCTM = VARIABLE USED FOR TRACING AND DEBUGGING PURPOSES
C     MEASURED IN MICRO SEC.
      TRCTM = XX(114)
```

```
C
C       XX(116) = PRPGDLY = ONE WAY PROPAGATION DELAY
        PRPGDLY = XX(116)
C
C       XX(117) = ACKPKT = ACKPKT PROCESSING TIMES IN MICRO SECONDS
        ACKPKT = XX(117)
C
C       XX(118) = BMOVE = BLOCK MOVE PROCESSING TIMES IN MICRO SECONDS
        BMOVE = XX(118)
C
C       XX(119) = ENQUEU = ENQUEUE PROCESSING TIMES IN MICRO SECONDS
        ENQUEU = XX(119)
C
C       XX(120) = DEQUEU = DEQUEUE PROCESSING TIMES IN MICRO SECONDS
        DEQUEU = XX(119)
C
C       XX(121) = RCVCOM = RCVCOM FROM APPLICATION LAYER PROCESSING TIMES
C                          IN MICRO SECONDS
        RCVCOM = XX(121)
C
C       XX(122) = SNDCOM = SEND COMMAND TO APPLICATION LAYER PROCESSING TIMES
        SNDCOM  = XX(122)
C
C       XX(123) = INFPKT = INFPKT PROCESSING TIMES IN MICRO SECONDS
        INFPKT = XX(123)
C
C       XX(124) = RCVMSG = RCVMSG PROCESSING TIMES IN MICRO SECONDS
        RCVMSG = XX(124)
C
C       XX(125) = RECEIVE = RECEIVE PROCESSING TIMES IN MICRO SECONDS
        RECEIVE = XX(125)
C
C       XX(126) = SEND = SEND PROCESSING TIMES IN MICRO SECONDS
        SEND = XX(126)
C
C       XX(127) = SNDSPR = SEND SUPERVISORY FRAME PROCESSING TIMES IN MICRO
C                          SECONDS
        SNDSPR = XX(127)
C
C       XX(128) = SNDMSG = SNDMSG PROCESSING TIMES IN MICRO SECONDS
        SNDMSG = XX(128)
C
        DO 10 I = 1,MAXSTA
C
           NDSTS(I)= 0
           IREJSTS(I) =0
           ITMOUTCNT(I)=0
           IRXMTCNT(I)=0
           ICOUNT(I)=0
           IOTSQCNT(I)=0
           IDMGPKTCNT(I)=0
           PKTCNT(I) = 0
           RCVSVR(I) = 1
C
   10 CONTINUE
C
        DO 20 I = 1,MAXSTA
C
           TOTPKTCNT(I) = 0
           ITMOT(I) = 0
```

```
            ISND(I) = 0
            IRXSTS(I)=0
            IPKTERR(I) = 0
            ICHKSTS(I) = 0
            JSND(I) = 0
            IAKVAR(I) = 0
            SFRM(I) = 0
            IABRTPKT(I)=0
C
   20 CONTINUE
C
      DO 30 I = 1,MAXSTA
C
            KRCV(I) = 0
            IFRM(I) = 0
            JCHKSTS(I) = 0
            IRCVCNT(I) = 1
            INFOBTS(I) = 0
            JFRM(I) = 0
            RRFRMCNT(I) = 0
            SFRMCNT(I) = 0
            BFRCNT(I) = 0
            RRSTS(I) = 0
            IPKTREJ(I) = 0
C
C
   30 CONTINUE
C
      RETURN
      END
```

```
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: event.for      AUTHOR: M.S.Kashefipour        DATE: 7-15-86
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0         RELEASE DATE : 7-15-86
C ------------------------------------------------------------------
C PURPOSE :     This series of subroutines form the discrete event
C               portion of the TISL LAP-D simulation.
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C    NAME                    DESCRIPTION
C ------------------------------------------------------------------
C event            ! Interfaces the SLAM scheduling routine with the events
C                  ! that are to be scheduled
C alloctrsc        ! performs logic routines on resource allocation
C sndifrm          ! receives a packet from higher layers , makes a copy
C                  ! of it and would pass the packet to the transmitter
C trnsmt           ! start timer , start transmitting
C laptrns          ! terminates the frame after a successful transmission
C                  ! and sets the transmitter to idle
C laprcv           ! check for correct CRC ,and handover the received frame
C                  ! to the data link layer.
C sndrcvack        ! takes necessary action for sending and receiving
C                  ! acknowledgments
C rcvifrm          ! destination receives an error free packet,make sure
C                  ! packet is in sequence
C srch             !  remove timeout event from the event calender
C                  ! (stop timer)
C rmvackfrm        ! packet is acknowledged remove this packet from
C                  ! retransmission buffer
C sndsfrm          ! destination receives out of sequence packet ,send a
C                  ! request to the source for retrans. of this packet
C                  ! ( send a REJ frame )
C rcvsfrm          ! source that sent a out of seq. packet receives a request
C                  ! for retransmission of that packet
C timeout          ! timer expires for the packet which no ack. received
C rexmit           ! begin retransmission of requested packet and any other
C                  ! packets outstanding
C applyclyr        ! releases packets from the wait node and receives
C                  ! data packets form the data link layer
C ------------------------------------------------------------------
C AMENDMENTS:
C ------------------------------------------------------------------
C     DATE          DESCRIPTION
C ------------------------------------------------------------------
C                  ! (none)
C ------------------------------------------------------------------
C
C NOTE:   These routines must be linked to the SLAM FORTRAN library!!!
C
C ------------------------------------------------------------------
C
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME:  event
C
C PURPOSE :       This routine interfaces the event scheduling routine
C                 with the events that must be scheduled.  This routine
C                 allows events to be called by their number as opposed
C                 to calling a routine by its name.  This subroutine is
C                 required by all SLAM discrete event models.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME                TYPE     CLASS  RANGE
C ------------------------------------------------------------------
C ievent           !integer! read  ! 1-10     !Number of the event that is
C                  !       !       !          !is being called.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C                  !       !       !          !(none)
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                    ROUTINES SCHEDULING THIS SUBROUTINE:
C ------------------------------------------------------------------
C (none)                               SLAM driver file
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C sndifrm          ! receives a packet from higher layer,makes a copy of
C                  ! it and store it in retransm. buffer and begin trnsm.
C                  ! the packet
C trnsmt           ! when the transmitter is idle will transmit the frame,
C                  ! start the timer for each outgoing frame,and transmit
C                  ! the next packet in sequence as soon as finish transmitting
C                  ! the current frame
C laptrns          ! performs the logic for packet termination and
C                  ! transmission of the next frame
C laprcv           ! performs the logic for a error free communication
C                  ! between a source and destination , also based
C                  ! the frame type would invoke the appropriate part
C                  ! of the data link layer.
C rcvifrm          ! destination receives a frame,check if this frame is
C                  ! in sequence and is error free
C sndsfrm          ! receiver rejects the out of seq. frame ,send a request
C                  ! for retransm. for this packet
C rcvsfrm          ! source which sent the out of seq. frame ,has received a
C                  ! request for retrans. from the other node
C timeout          ! timer expired for the frame which no ack. received
C rexmit           ! begin retransmission of requested frame and any other
C                  ! frames outstanding
C applyclyr        ! implements the user and is to receives data packets
C                  ! from the data link layer or to send data packets
C                  ! to the data link layer
C------------------------------------------------------------------
C
      SUBROUTINE EVENT(I)
C
         GO TO (1,2,3,4,5,6,7,8,9,10),I
C
```

```
      1 CALL SNDIFRM
        RETURN
C
      2 CALL TRNSMT
        RETURN
C
      3 CALL LAPTRNS
        RETURN
C
      4 CALL LAPRCV
        RETURN
C
      5 CALL RCVIFRM
        RETURN

      6 CALL SNDSFRM
        RETURN
C
      7 CALL RCVSFRM
        RETURN
C
      8 CALL TIMEOUT
        RETURN
C
      9 CALL REXMIT
        RETURN
C
     10 CALL APPLCLYR
        RETURN
        END
C
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: sndifrm                        EVENT NUMBER: 1
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                Collect statistics for time that this packet
C                was queued in the application layer. Next assign
C                a sequence number to this packet and store a duplicate
C                of it in the repeate queue in case of retransmission.
C                Finally hand over the packet to the transmit part
C                of the data link layer for piggybaking the acknowledgment
C                and timer considerations.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C SLAM Network Model                  trnsmt,  event 2
C                                     sndsfrm, event 6
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C filem           ! A SLAM routine that places items in files.
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
C
      SUBROUTINE SNDIFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, ILOC, MMLE, NEXT, NNQ, ITMP
C
      REAL QUDLY, PRSSDLY
C
      INODE=IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE=IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      ILOC=IFIX(ATRIB(4))
C     DEFINE THE REPEATE QUEUE OF THE SOURCE NODE
C
      CALL COLCT(ATRIB(11), 31)
C     COLLECT STAT. ON AVE. MESSAGE LENGTH
C
      PKTCNT(INODE) = PKTCNT(INODE) + 1
C     INCREMENT # OF FRAMES GENERATED FROM THIS NODE
C
      ATRIB(5) = PKTCNT(INODE)
C     SET THE PACKET COUNT NUMBER
C
      ATRIB(12) = TNOW
```

```
C       VAR.  TO RECORD THE LAP TIME DELAY
C
        QUDLY = TNOW - ATRIB(1)
C       COMPUTE THE QUEUEING DELAY FROM USER TO LINK LEVEL
C
        CALL COLCT(QUDLY,5)
C       COLLECT STATS.  ON QUEUEING DELAY
C
        JSND(INODE)=JSND(INODE) + 1
C       INCREMENT THE SEND SEQ # FOR THIS NODE ( V(s) - PROTOCOL VERSION )
C
        IF(JSND(INODE).GT.IWNDSZ)THEN
C          CHECK FOR WINDOW ROTATION
C
           JSND(INODE) = 1
C          RESET THE SEND SEQ #
C
        ENDIF
C
        ATRIB(6) = JSND(INODE)
C       ASSIGN THE SEQUENCE NUMBER OF THE NEXT PACKET TO BE
C       TRANSMITTED ( N(s) - PROTOCOL VERSION )
C
        CALL FILEM(ILOC,ATRIB)
C       SAVE THE PACKET IN THE REPEAT QUEUE IN CASE OF RETRANSMISSION
C
        IF(NDSTS(INODE).EQ.2)THEN
C          IF NODE IS IN RETRANSMIT STATUS INCREMENT THE NUMBER
C          OF OUTSTANDING FRAMES TO BE TRANSMITTED.
C          THIS CONDITION CAN ONLY OCCUR WHEN APPLICATION LAYER
C          DOESN'T KNOW THE CURRENT STATUS OF DATA LINK LAYER,
C          SO IT WOULD SEND A FRAME TO DATA LINK LAYER WHICH
C          NORMALLY IT SHOULDN'T.
C
           BFRCNT(INODE) = BFRCNT(INODE) + 1
C          INCREMENT NUMBER OF OUTSTANDING FRAMES TO BE
C          TRANSMITTED
C
           RETURN
C
        ENDIF
C
        IF(ICHKSTS(INODE).EQ.1)THEN
C          CHECK IF TRANSMITTER BUSY
C          TRANSMITTER IS BUSY, STORE THE PACKET, WHEN TRANSMITTER
C          RETURNS TO IDLE WILL TRANSMIT THIS FRAME.
C
           JFRM(INODE) = JFRM(INODE) + 1
C          INCREMENT NUMBER OF PACKETS WAITING FOR TRANSMISSION
C
           RETURN
C
        ENDIF
C
        IF(SFRM(INODE).GE.1)THEN
C          CHECK IF A SUPERVISORY FRAME WATING FOR TRANSMISSION
C          IF SO TAKE THE PROPER STEPS TO TRANSMIT THE S-FRAME AND
C          STOP TRANSMISSION OF I -FRAMES
C
           ATRIB(20) = 0.0
```

```
C          SET A CONDITION IF THIS NODE TIMEOUT, THE SCHDL.
C          TRANS.  OF I-FRAME WILL BE REMOVED FROM THE EVENT
C          CALENDAR
C
           ATRIB(18) = 1.0
C          S-FRAME INDICATOR FOR THIS PACKET
C
           ICHKSTS(INODE) = 1
C          SET THE TRANSMITTER BUSY
C
           PRSSDLY = SNDSPR
C          PROCESSING TIME TE SEND A SUPERVISORY FRAME
C
           CALL SCHDL(6,PRSSDLY,ATRIB)
C          START THE TRANSMISSION OF S-FRAME
C
           RETURN
C
         ENDIF
C
         ICHKSTS(INODE) = 1
C        SET THE TRANSMITTER BUSY FOR THIS NODE
C
         CALL SCHDL(2,0.0,ATRIB)
C       INVOKE THE TRANSMIT PART OF THE DATA LINK LAYER FOR
C        PROPER FRAMING ,ACKNOWLEDGMENT PIGGYBACKING ,  AND
C        TIMER CONSIDERATIONS
C
         RETURN
         END
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: trnsmt                          EVENT NUMBER: 2
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                perform the transmissin and retransmission of
C                I - frames ,piggyback the acknowledgement of the
C                outgoing frame,  start the retransmission timer
C                for the outgoing I - frames , and finally handover
C                the frame to the transmitter for transmission.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laptrns , event 3                     laptrns, event 3
C rexmit  , event 9                     timeout, event 8
C sndifrm , event 1                     applclyr, event 10
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C copy            ! A SLAM routine that copies the attributes
C                 ! of an entity from a file into the attribute
C                 ! array
C
C ------------------------------------------------------------------
      SUBROUTINE TRNSMT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, ILOC, MMLE, NEXT, NPRED, NNQ, I
     1, ITMP, J, NNRSC

      REAL PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(4))
C     DEFINE THE REPEATE QUEUE FOR THIS NODE
C
      ITMP = ISND(INODE)
C     SAVE SEQ. NUM OF THE LAST PACKET TRANSMITTED
C
      IF(IRXSTS(INODE).EQ.1)THEN
C        CHECK IF SOURCE NODE IS IN THE RETRANSMIT STATUS
C
         NDSTS(INODE) = 2
```

```
C          SOURCE RETRANSMITTING
C
           IRXMTCNT(INODE) = IRXMTCNT(INODE) + 1
C          INCREMENT NUMBER OF RETRANSMITTED PACKETS
C          FROM THIS NODE
C
         ENDIF
C
         IF(KTMP(INODE).EQ.0)THEN
C          SET A CONDITION THAT WHEN SOURCE IS IN RETRANS. STATUS
C          THIS PART OF THE CODE WOULD NOT BE CONCIDERED AT THE
C          TIME
C
           ISND(INODE)=ISND(INODE)+1
C          INCREMENT THE SEND SEQUENCE NUMBER FOR
C          THIS NODE ( V(s) - PROTOCOL VERSION )
C
           IF(ISND(INODE).GT.IWNDSZ)THEN
C            CHECK FOR WINDOW ROTATION
C
             ISND(INODE) = 1
C            RESET THE SEND SEQ #
C
           ENDIF
C
         ELSE
C
           KTMP(INODE) = 0
C          RESET THE VARIABLE THAT INDICATES BEGINING OF RETRANSMISSION
C
         ENDIF
C
         NEXT = MMLE(ILOC)
C        SET THE NEXT POINTER TO THE LAST PACKET ARRIVED
C        IN THE TRANSMISSION QUEUE
C
C
      20 IF(NEXT.EQ.0)THEN
C          NO ENTRIES IN THIS FILE
C
           IF(IRXSTS(INODE).EQ.1)THEN
C            JUST FINISHED RETRANSMITTING ALL THE
C            REQUESTED PACKETS
C
             ISND(INODE) = ITMP
C            SET THE SEND SEQ. NUM EQUAL TO THE LAST PACKET
C            WAS TRANSMITTED
C
             IRXMTCNT(INODE) = IRXMTCNT(INODE) - 1
C            DECREMENT # OF RETRANS. FRAMES, NO FRAMES TO SEND
C            THIS VAR. SHOULD NOT HAVE BEEN INCREMINTED
C
             BFRCNT(INODE) = 0
C            RESET THE NUMBER OF OUTSTANDING I - FRAMES
C            FOR TRANSMISSION - NO PACKET TO RETRANSMIT
C
             ICHKSTS(INODE) = 0
C            SET THE SOURCE TO IDLE STATUS
C
             ATRIB(8) = 1.0
```

```
C           SET A VARIABLE THAT INFORMS THE APPLICATION
C           LAYER THAT LINK LAYER IS READY TO TRANSMIT THE
C           NEXT FRAME
C
            ATRIB(20) = 0.0
C           NON TIMEOUT EVENT
C
            PRSSDLY = BMOVE + ENQUEU + RCVCOM
C           COMPUTE THE PROCESSING TIME NEEDED FOR PACKET
C           TRANSFER FROM APPLICATION LAYER TO THE LINK LAYER
C
            CALL SCHDL(10,PRSSDLY,ATRIB)
C           SCHDL APPLICATION LAYER
C
            RETURN
C
         ELSE
C
C           THE ENTRY WAS NOT FOUND , RESTORE ALL THE PARAMETERS
C           BEFORE THIS SUBROUTINE WAS CALLED
C
            ISND(INODE) = ITMP
C           RESTORE THE SEQ. NUM OF THE LAST PACKET TRANS.
C
            ICHKSTS(INODE) = 0
C           SET THE TRANSMITTER TO IDLE STATUS
C
            JFRM(INODE) = 0
C           NO FRAMES WAITING FOR TRANSMISSION
C
            ATRIB(8) = 1.0
C           SET A VARIABLE THAT INFORMS THE APPLICATION LAYER
C           THAT LINK LAYER IS READY TO TRANSMIT THE NEXT FRAME
C
            PRSSDLY = BMOVE + ENQUEU + RCVCOM
C           COMPUTE THE PROCESSING TIME NEEDED TO TRANSFER
C           ONE PACKET FROM APPLICTION LAYER TO LINK LAYER
C
            CALL SCHDL(10,PRSSDLY,ATRIB)
C           SCHDL APPLICATION LAYER
C
C           RETURN
C
         ENDIF
C
      ENDIF
C
      CALL COPY(-NEXT,ILOC,ATRIB)
C     COPY THE PACKET ATTRIBUTES FROM TRANSMISSION QUEUE
C     INTO THE ATTRIBUTE ARRAY
C
      IF(IFIX(ATRIB(6)).EQ.ISND(INODE))THEN
C        CHECK IF THE COPIED PACKET IS THE NEXT PACKET TO
C        BE TRANSMITTED
C
         ATRIB(19) =KRCV(INODE)
C        PIGGYBACK THE ACKNOWLEDGEMENT TO THE OUTGOING
C        I - FRAME
C
         JCHKSTS(INODE) = 1
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: laptrns                        EVENT NUMBER: 3
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                First terminates the frame after a successful transmission
C                and sets the transmitter to idle. Second schedule the
C                propagation event for the frame just transmitted . Also
C                collect statistics on the average transmission time for
C                frame just transmitted ( includes S & I frames). Finally
C                start to transmit the next frame . To transmit the next
C                frame , source would transmit any S - frames waiting for
C                transmission before transmitting any I - frames . Also
C                for transmitting I - frames priority is given to I - frames
C                that are waiting for retransmission.
C                If source node has no S - frames and I - frames waiting
C                for transmission schedule the application layer to send
C                more I - frames to the data link layer.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |        | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |        | Declare variables and
C                   |       |      |        | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                EVENTS SCHEDULEING THIS EVENT:
C ------------------------------------------------------------------
C event 4,  laprcv                                  event 2, trnsmt
C event 2,  trnsmt
C event 6,  sndsfrm
C event 10, applclyr
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C colct             | SLAM routine to collect statistics on variables.
C------------------------------------------------------------------
      subroutine laptrns
c
      include 'params.dat'
      integer inode, jnode, nnrsc, i, iloc, nnq
      real tsys, taccss, tchnl, totsys, temp, prssdly
c
      inode = ifix(atrib(2))
c     define the source node
c
      jnode = ifix(atrib(3))
c     define the destination
c
      iloc = ifix(atrib(4))
c     define the repeat queue number
c
      ichksts(inode) =  0
```

```
C          SET THE TRANSMITTER BUSY ....  I - FRAME TRANSMISSION
C
           ICHKSTS(INODE) = 1
C          TRANSMITTER  BUSY ....  FRAME TRANSMISSION .  NOTE THAT
C          THIS VARIABLE IS USED FOR ALL FRAMES BUT " JCHKSTS "
C          IS USED FOR TRANSMISSION OF I-FRAMES ONLY
C
           ATRIB(20) = 1.0
C          TIMEOUT EVENT
C
           IF(IRXSTS(INODE).EQ.1)THEN
C             IF SOURCE IS IN RETRANSMIT STATUS , DECREMENT
C             THE NUMBER OF FRAMES OUTSTANDING FOR RETRANSMISSION
C
              BFRCNT(INODE) = BFRCNT(INODE) - 1
C             DECREMENT # OF OUTSTANDING I-FRAMES WAITING FOR TRANSMISSION
C
           ENDIF
C
           CALL SCHDL(8,ATRIB(7),ATRIB)
C          START THE TIMER FOR THE OUTGOING I - FRAME
C          IF THE TRANSMITTING NODE DOESN'T RECEIVE AN
C          ACKNOWLEDGEMENT FOR THIS FRAME BEFORE TIMER
C          EXPIRES , THE PROCEDURES FOR TIMEOUT RECOVERY
C          CONDITION WILL BE FOLLOWED
C          SCHEDULE THE TIME OUT EVENT TO OCCUR AT TIME
C          EQUAL TO ATRIB(7)
C
           ATRIB(20) = 0.0
C          RESET THE TIMER VARIABLE
C
        ELSE
C
           NEXT = NPRED(NEXT)
C          CONTINUE THE SAERCH FOR I-FRAMES IN THE
C          TRANSMISSION QUEUE
C
           GO TO 20
C
        ENDIF
C
        TOTPKTCNT(INODE) = TOTPKTCNT(INODE) + 1
C       INCREMENT NUMBER OF I-FRAMES TRANSMITTED FROM THIS NODE
C
        PRSSDLY = SEND + SNDMSG
C       COMPUTE THE PROCESSING DELAY NEEDED FOR FRAMING,
C       PIGGYBACKING ACKNOW.  ,AND HANDING OVER THE FRAME
C       TO THE TRANSMITTER
C       TRANSMISSION PROCESSING DELAY
C
        CALL SCHDL(3,PRSSDLY + ATRIB(9),ATRIB)
C
        RETURN
        END
C
```

```
c       set the transmitter idle
c
        jchksts(inode) = 0
c       set the transmitter idle(i-frames only)
c
        if(ifix(atrib(18)).le.0)then
c          check if the transmitted frame was an I - frame , if so
c          make sure that this frame is valid and is not to be
c          discarded
c
           if(ifrm(inode).eq.0)then
c             valid I - frame ,schdl the propagation event
c
              prssdly = prpgdly + rcvmsg + receive + crcchk
c             processing and propagation time
c
              call schdl(4,prssdly,atrib)
              propagation event
c
              call colct(atrib(9),1)
c             collect statistics on average transmission time for I - frames
c
              call colct(atrib(9),2)
c             collect statistics for all frames
           else
c             invalid frame . This frame is to suspended because
c             of timer recovery condition or REJ recovery condition
              ifrm(inode) = 0
c
           endif
c
        else
c          frame transmitted is a S - frame , schdl the propagation
c          event
c
           prssdly = prpgdly + rcvmsg + receive + crcchk
c          processing and propagation time
c
           call schdl(4,prssdly,atrib)
           propagation event
c
           call colct(atrib(9),2)
c          collect statistics for all frames
        endif
c
c       source just finished transmitting a frame , before transmitting
c       the next frame ,source is to check some conditions before
c       transmitting the next in sequence I - frame. Those conditions
c       are as follows :
c       Any S - frame to send ?
c       Is source node is is retransmit status ?
c
        if(ndsts(inode).gt.0.or.sfrm(inode).ge.1)then
c          check if the source node is in retransmission status,or
c          has a S-FRM waiting to be transmitted
c
           if(sfrm(inode).ge.1)then
c             S-FRM waiting for transmission
c
              atrib(18) = 1.0
```

```
c            S-FRM indicator
c
             sfrm(inode) = sfrm(inode) - 1
c            decrement number of S-frames waiting for transmission
c
             prssdly = sndspr
c            compute the processing time to send a supervisory
c            frame
c
             ichksts(inode) = 1
c            set the transmitter busy
c
             call schdl(6,prssdly,atrib)
c            schdl the sndsfrm event for transm.  of S-FRM
c
             return
c
          elseif(ndsts(inode).gt.0)then
c            source node is in retransmit status ,start
c            of requested frames
c
             if(itmot(inode).ne.1)then
c               before retransmit make sure that node is not in timer
c               recovery condition
c
                if(ndsts(inode).eq.1)then
c                  check the status of retransmission recovery
c
                   bfrcnt(inode) = nnq(iloc)
c                  node is about to start retransmission process, define
c                  the number of outstanding frames to be retransmitted
c
                endif
c
                if(bfrcnt(inode).gt.0)then
c                  check if any more frames outstanding for
c                  retransmission
c
                   ichksts(inode) = 1
c                  set the transmitter busy
c
                   call schdl(2,0.0,atrib)
c                  schdl trnsmt event for transmission of I-FRMS
c
                endif
c
             endif
c
             return
c
          endif
c
c         else
c            source node has no S-FRM waiting and isnot in
c            retrans. status. Also the transmitted packet
c            was a I-FRM . Start the process of packet trans.
c
          endif
c
          if(itmot(inode).ne.1)then
```

```
c          if node is not in timer recovery condition, then start to send
c          I-frames
c
           if(jfrm(inode).ge.1)then
c             packets waiting for transmission at the link level
c             don't release any packet from wait node(application
c             level) until the waiting packets are transmitted
c
              jfrm(inode) = jfrm(inode) - 1
c             Decrement number of packets waiting for transmission
c             for this node
c
              ichksts(inode) = 1
c             set the transmitter busy
c
              call schdl(2,0.0,atrib)
c             schdl trnsmt event for transmission of I-frames
c
           else
c
              atrib(8) = 1.0
c             set the application layer indicator that tells application
c             layer to send I-frame to the data link layer
c
              atrib(20) = 0.0
c             non timeout event
c
              prssdly = bmove + enqueu + rcvcom
c             compute the processing time needed for application
c             layer to send a frame to LLC level
c
              call schdl(10,prssdly,atrib)
c             schdl application layer
c
           endif
c
        endif
c
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: laprcv                        EVENT NUMBER: 4
C ------------------------------------------------------------------
C
C PURPOSE :      This ruoitne performs two functions. First, it checks if
C                frame (I - FRAMES only) has a correct CRC(no transmission
C                errors) .If so handover the frame to the upper part of
C                the data link layer. If frame is in error ,no action
C                will be taken and discard the frame. Second , based
C                on the frame type ( I - FRAME or S _ FRAME ) determine
C                which routine should be invoked to process the received
C                frame.If received frame is an I - frame schdl RCVIFRM
C                event ,but if received frame is a S - frame schdl
C                RCVSFRM event.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file   ! read !         !Declare variables and
C                 !        !      !         !establish common block
C ------------------------------------------------------------------
C EVENTS SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laptrns, event 3                          rcvifrm, event 5
C                                           rcvsfrm, event 7
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
c
      subroutine laprcv
c
      include 'params.dat'
      integer inode,jnode
c
      real prssdly,rannum,unfrm
c
      inode = ifix(atrib(2))
c     define the source node
c
      jnode = ifix(atrib(3))
c     define the destination node
c
      if(ifix(atrib(18)).eq.0)then
c        check if the received frame is an informatiom frame (I - FRAME)
c
         blkerr = 1 - ((1 - biterr)**atrib(11))
c        compute the probability of this packet
c        being in error
c
         rannum = unfrm(0.0,1.0,7)
c
c        using a uniform random number generator to
c        determine randomly if this packet is in error
c
```

```
            if(rannum.gt.blkerr)then
c             check if this packet is in error
c
              atrib(14) = 0.0
c             correct packet
c
              prssdly = infpkt
c             processing time needed for information frames
c
              call schdl(5,prssdly,atrib)
c             error free packet , hand over this packet to the
c             data link layer
            else
c
              atrib(14) = 1.0
c             packet in error
c
c             invalid frame, discard it and don't invoke the rcvifrm
c             event
c
              icount(inode) = icount(inode) + 1
c             frame in error, increment number of frames in error
c             for this node
c
              idmgpktcnt(inode) = idmgpktcnt(inode) + 1
c             increment number of damaged frames received by this node
c
            endif
c
          elseif(ifix(atrib(18)).ge.1)then
c            check if the received frame is a supervisory frame( S - FRAME)
c
            atrib(2) = float(jnode)
c           the node that was a receiver becomes a transmitter
c
            atrib(3) = float(inode)
c           the node that was a transmitter becomes a receiver
c
            atrib(4) = atrib(2) + maxsta
c           based on the source node determine the repeate queue
c
            prssdly = ackpkt
c           processing time to process the received supervisory
c           frame .
c
            call schdl(7,prssdly,atrib)
c           schdl RCVSFRM event
c
          endif
c
          return
          end
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rcvifrm                          EVENT NUMBER: 5
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                First, compute the probability of received frame
C                being in error then use a uniform random number
C                generator to determine if this frame is in error.
C                Second, check if this frame is in error if so discard
C                the frame . If the frame is error free, then check if
C                this frame is in sequence if so increment receive seq.
C                # . Third call subroutine SNDRCVACK to remove the
C                timeout event for the acknowledged frame from the
C                event calendar, and remove the acknowledged frame from the
C                retransmission buffer.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laprcv, event 4                          sndsfrm, event 3
C                                          subroutine sndrcvack
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE RCVIFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER  INODE, JNODE, JTMP, NNQ, I
C
      REAL RANNUM, UNFRM, SAVE, SAVE1, TMDLY, LPDLY, NMBTS, PRSSDLY
      DIMENSION SAVE1(42)
C
      INODE = IFIX(ATRIB(2))
C     SET THE SOURCE TO INODE
C
      JNODE = IFIX(ATRIB(3))
C     SET THE DESTINATION TO JNODE
C
      IF(IRCVCNT(JNODE).GT.IFIX(ATRIB(5)))RETURN
C        CHECK FOR DUPLICATE FRAMES
C
      IF(IFIX(ATRIB(6)).EQ.RCVSVR(JNODE)) THEN
C        PACKET CONTAINED NO ERROR , NOW MAKE SURE IF THIS PACKET HAS
C        THE CORRECT SEQUENCE NUMBER
C
         IF(IREJSTS(JNODE).GE.1)THEN
C           CHECK IF THE RECEIVING NODE HAS A REJECTED FRAME
C
            IF(IFIX(ATRIB(6)).EQ.IPKTREJ(JNODE))THEN
```

```
C
C          CHECK FOR THE SEQUENCE NUMBER OF THE FRAME WHICH
C          CAUSED THE REJ CONDITION
C
           IREJSTS(JNODE) = 0
           RESET THE REJECT VARIABLE
C
           IPKTREJ(JNODE) = 0
           RESET THE VARIABLE WHICH STORES THE SEQUENCE # OF
           REJECTED FRAME
C
        ENDIF
C
      ENDIF
C
      NMBTS = ATRIB(11) - XX(83)
      CALCULATE # OF INFORMATION BITS
C
      INFOBTS(INODE) = INFOBTS(INODE) + IFIX(NMBTS)
C     ACCUMULATE TOATL # OF SUCCESSFUL BITS TRANSMITTED
      BY THE TRANSMITTING NODE
C
      LPDLY = TNOW - ATRIB(12)
      CALCULATE LAP DELAY
C
      CALL COLCT(LPDLY,6)
C     COLLECT STAT. FOR AVE LAP DELAY
C
      RCVSVR(JNODE) = RCVSVR(JNODE) + 1
C     CORRECT PACKET, INCREMENT THE RECEIVE SEQ # FOR THIS NODE
C
      KRCV(JNODE) = KRCV(JNODE) + 1
C     INCREMENT THE ACK. VARIABLE FOR THIS NODE
C
      IRCVCNT(JNODE) = IRCVCNT(JNODE) + 1
C     INCREMENT THE # OF CORRECT PACKETS RECEIVED
C     BY THIS NODE PLUS ONE ( SEQ # OF THE NEXT PACKET
C     TO BE RECEIVED BY THIS NODE )
C
      IF(RCVSVR(JNODE).LE.IWNDSZ)THEN
C        MAKE SURE THE RECEIVER SEQUENCE IS WITHIN THE SPECIFIED
C        LIMIT . THEN UPDATE THE RECEIVE SEQUENCE NUMBER WHICH
C        ACKNOWLEDGES ALL THE PACKETS NUMBERED UP TO AND INCLUDING
C        RCVSVR(JNODE) - 1
C
         ATRIB(17) = RCVSVR(JNODE)
C        UPDATE THE RECEIVE SEQ. #
C
      ELSE
C
         RCVSVR(JNODE) = 1
C        RESET THE RECEIVE SEQ #
C
         ATRIB(17) = RCVSVR(JNODE)
C        DEFINE THE RECEIVE SEQ #.
C
      ENDIF
C
      ATRIB(8) = 2.0
C     APPLICATION LAYER INDICATOR, SEND I-FRAME TO APPLICATION LAYER
C
```

```
            ATRIB(20) = 0.0
C           NON TIMEOUT EVENT
C
            PRSSDLY = BMOVE + SNDCOM
C           PROCESSING TIME TO SEND FRAME TO APPLICATION LAYER
C
            CALL SCHDL(10,PRSSDLY,ATRIB)
C           SCHDL APPLICATION LAYER
C
            CALL SNDRCVACK
C           CORRECT PACKET AT THE RECEIVER ,CALL SND_RCV_ACK
C           SUBROUTINE TO TAKE NECESSARY ACTION FOR SENDING
C           AND RECEIVING ACKNOWLEDGMENTS
C
         ELSE
C
            ATRIB(14) = 2.0
C           OUT OF SEQUENCE PACKET INDICATOR
C
            CALL SNDRCVACK
C           CALL SND_RCV_ACK SUBROUTINE TO TAKE CARE OF
C           SENDING AND RECEIVIG ACKNOWLEDGMENTS

            IREJSTS(JNODE) = IREJSTS(JNODE) + 1
C           OUT OF SEQUENCE FRAME , INCREAMENT THE FRAME REJ VAR.
C
            ICOUNT(INODE) = ICOUNT(INODE) + 1
C           INCREMENT NUMBER OF FRAMES IN ERROR FOR THIS
C           NODE
C
            IOTSQCNT(INODE) = IOTSQCNT(INODE) + 1
C           INCREAMENT NUMBER OF OUT SEQ. FRAMES RECEIVED
C           BY THIS NODE
C
            IF(IREJSTS(JNODE).LE.1)THEN
C              CHECK IF THERE IS ALREADY ONE REJ FRAME OUTSTANDING
C              IF NOT , THEN PROCEED TO SEND A REJ FRAME TO THE NODE
C              WHICH TRANSMITTED THE OUT OF SEQ. FRAME
C
               IPKTREJ(JNODE) = IFIX(ATRIB(6))
C              STORE THE SEQUENCE NUMBER OF THE FRAME CAUSED
C              THE REJECT CONDITION
C
               ATRIB(16) = IRCVCNT(JNODE)
C              DEFINE THE SEQ # OF THE NEXT PACKET TO BE RECEIVED
C
               ATRIB(17) = RCVSVR(JNODE)
C              REQUEST OF RETRANSMISSION OF LAST PACKET
C              RECEIVED CORRECTLY BY THIS NODE + 1
C
               ATRIB(18) = 1.0
C              S-FRM INDICATION
C
               ATRIB(19) = IRCVCNT(JNODE) - 1.0
C              DEFINE THE ACKNOWLEDGE VARIABLE
C
               ATRIB(20) = 0.0
C              NON TIME OUT EVENT
C
               ATRIB(2) = FLOAT(JNODE)
```

```
C
C           NODE THAT WAS RECEIVING BECOMES A TRANSMITTER
C
            ATRIB(3) = FLOAT(INODE)
C           NODE THAT WAS A TRANSMITTING BECOMES A RECEIVER
C
            ATRIB(4) = ATRIB(2) + MAXSTA
C           DEFINE THE REPEAT QUEUE FOR TRANSMITTING NODE
C
            PRSSDLY = SNDSPR
C           PROCESSING TIME TO SEND A SUPERVISORY FRAME
C
            IF(ICHKSTS(JNODE).EQ.0)THEN
C              CHECK IF TRANSMITTER IS IDLE
C
               ICHKSTS(JNODE) = 1
C              SET THE TRANSMITTER BUSY
C
               CALL FILEM((NCLNR-2),ATRIB)
C              STORE THE SUPEVISORY FRAME IN A TEMPORARY BUFFER
C
               CALL SCHDL(6,PRSSDLY,ATRIB)
C              START THE TRANSMISSION OF S-FRAME AFTER THE PROCESSING DELAY
C
            ELSE
C              TRANSMITTER IS BUSY
C
               SFRM(JNODE) = SFRM(JNODE) + 1
C              S-FRM WAITING FOR TRANSMISSION
C
               CALL FILEM((NCLNR-2),ATRIB)
C              STORE THE REJ FRAME IN A TEMPORARY BUFFER, AS SOON AS
C              TRANSMITTER BECOMES IDLE THIS S-FRAME WILL BE TRANSMITTED
C
            ENDIF
C
         ENDIF
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: sndsfrm                          EVENT NUMBER: 6
C ------------------------------------------------------------------
C
C PURPOSE :      This routine implements the source of a node which
C                transmits only S-FRAMES . A supervisory frame can be
C                a RR frame which acknowledeges a frame that just received
C                by the receiver of this node , a REJ FRAME which
C                indicates that the receiver of this node has received
C                an out of sequence packet , and finally a RR frame used
C                for timeout inquiry status purpose. The distinction
C                among different frames are provided as follows:
C                if ATRIB(18) = 1 then the frame is a REJ FRAME
C                if ATRIB(18) = 2 then the frmae is a RR  FRAME used
C                for acknowledgement purposes
C                if ATRIB(18) = 3 then the frame is a RR  FRAME used
C                for timeout inquiry status purpose
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C sndifrm,  event 1
C rcvifrm , event 5                          laptrns, event 3
C laptrns , event 3
C timeout , event 8
C subroutine sndrcvack
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C copy            ! copies the attributes of an entity from a file, and
C                 ! place them in attribute array
C rmove           ! A SLAM routine which removes an entity from a file
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE SNDSFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, MMFE, NSUCR, ILOC, NNQ, JLOC, I
C
      REAL TEMP, PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(4))
C     DEFINE THE REPEAT QUEUE OF THE NODE WHICH
C     TRANSMITTED THE FRAME
C
```

```
      PERFORM THE NECESSARY LOGIC TO DETERMINE WHAT KIND OF
      SUPERVISORY IS TO BE SENT

      IF(IFIX(ATRIB(18)).EQ.1.OR.IFIX(ATRIB(18)).EQ.3)THEN
         CHECK IF WE ARE TO TRANSMIT A REJ  FRAME OR A RR
         FRAME FOR TIMEOUT INQUIRY STATUS

         START THE SEARCH IN THE QUEUE THAT SUPERVISORY FRAMES ARE
         STORED FIND THE APPROPRIATE S - FRAME AND START THE TRANS.
         OF THIS FRAME IMMEDIATELY

         JLOC = NNQ((NCLNR-2))
         DEFINE THE NUMBER OF ENTRIES IN THE FILE WHICH
         S-FRAMES ARE STORED

         NEXT = MMFE((NCLNR-2))
         START TO TRANSMIT THE S-FRAME THAT HAS BEEN WEAITING
         FOR TRANSMISSION IN FILE NCLNR - 2

50       IF(NEXT.EQ.0)THEN
            NO ENTRIES IN SUPERVISORY QUEUE

            RETURN

         ENDIF

         CALL COPY(-NEXT,(NCLNR-2),ATRIB)
         COPY THE ATTRIBUTES OF THIS ENTRY IN THE ATTRIBUTE ARRAY

         IF(INODE.EQ.IFIX(ATRIB(2)))THEN
            CHECK IF THIS IS THE ENTRY THAT WE ARE LOOKING FOR

            IF(IFIX(ATRIB(18)).EQ.1)THEN
               CHECK IF THE SUPERVISORY FRAME UN QUESTION IS A REJ
               FRAME, IF SO INCREMENT NUMBER OF REJ FRAMES TRANSMITTED
               BY THIS NODE

               SFRMCNT(INODE) = SFRMCNT(INODE) + 1
               INCREMENT NUMBER OF REJ FRAMES TRANSMITTED FROM THIS NODE

            ENDIF

            ICHKSTS(INODE) = 1
            SET THE TRANSMITTER BUSY

            ATRIB(11) = MINSZ
            DEFINE THE FRAME SIZE FOR THE S-FRM

            ATRIB(9) = ATRIB(11)/RATE
            TRANSMISSION TIME IN MICROSEC.

             ATRIB(20) = 0.0
             NON TIMEOUT EVENT

             PRSSDLY = SEND
             PROCESSING TIME NEEDED TO HAND OVER THIS FRAME
             TO THE LOWER PART OF THE DATA LINK LAYER

             CALL SCHDL(3,PRSSDLY + ATRIB(9),ATRIB)
             SCHDL THE TRANSMISSION OF THIS FRAME
```

```
C
            CALL RMOVE(-NEXT,(NCLNR-2),ATRIB)
C           REMOVE THE WAITING S-FRAME FROM THE SUPERVISORY
C           QUEUE
C
            RETURN
C
         ELSE
C
            NEXT = NSUCR(NEXT)
C           PROCEED TO THE NEXT ENTRY IN SUPERVIS. QUEUE
C
            GO TO 50
C
         ENDIF
C
      ELSEIF(IFIX(ATRIB(18)).EQ.2)THEN
C        CHECK IF THIS NODE IS TO SEND A RR FRAME FOR
C        ACKNOWLEDEGEMENT PURPOSE
C
         ICHKSTS(INODE) = 1
C        SET THE SOURCE BUSY
C
         ATRIB(11) = XOHBIT
C        SET THE PACKET SIZE FOR THIS FRAME
C
         ATRIB(9) = ATRIB(11) / RATE
C        COMPUTE LENGTH OF THIS PACKET IN MICROSEC
C
         ATRIB(18) = 2.0
C        S - FRAME INDICATOR
C
         ATRIB(20) = 0.0
C        NON TIMEOUT EVENT
C
         RRFRMCNT(INODE) = RRFRMCNT(INODE) + 1
C        INCREMENT # OF RR FRAMES TRANSMITTED FROM THIS NODE
C
         PRSSDLY = SEND
C        PROCEESING TIME NEEDED TO HAND OVER THIS FRAME TO THE
C        LOWER PART OF THE DATA LINK LAYER
C
         CALL SCHDL(3,PRSSDLY + ATRIB(9),ATRIB)
C        BEGIN TRANSMISSION OF THIS FRAME
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rcvsfrm                          EVENT NUMBER: 7
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs following functions: it
C                receives an REJ-FRM and will immediately start the
C                necessary procedures to retransmit the requested
C                frame and any other frames outstanding. To do that,
C                this subroutine will remove all the time out events
C                and scheduled transmission events from the event calender,
C                then immediately will start the retransmission process.
C                Also receives RR frames which acknowledges frames transmitted
C                by this node. Finally is to respond and take necessary
C                actions for RR timeout inquiry status frames.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laprcv, event 4                        rexmit,  event 9
C                                        sndsfrm, event 6
C                                        timeout, event 8
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE RCVSFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, NSUCR, MMFE, I, NNQ, JLOC, JTMP
     1, ITMP, IMODE, IACK
C
      REAL SAVE, SAVE1, PRSSDLY
      DIMENSION SAVE(20), SAVE1(20)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      IMODE = IFIX(ATRIB(18))
C     DEFINE WHAT KIND OF SUPERVISORY FRAME IS RECEIVED
C
      GO TO (1,2,3), IMODE
C     BASED ON THE SUPERVISORY TYPE BRANCH TO THE APPROPRIATE PART
C     OF THE CODE
C
    2 CONTINUE
C     RR SUPERVISORY FRAME RECEIVED ACKNOWLEDGES ALL THE TRANSMITTED
C     PACKETS BY THIS NODE MINUS ONE ( N(r) - 1 PROTOCOL VERSION )
```

```
C
C
      CALL SRCH
C     REMOVE THE TIMEOUT EVENT FROM THE EVENT CALENDER
C     FOR THE ACKNOWLEDGED PACKET ( STOP THE TIMER )
C
      CALL RMVACKFRM
C     REMOVE THE ACKNOWLEDGED PACKET FROM THE REPEAT QUEUE
C
      RETURN
C
    1 CONTINUE
C     REJ FRAME RECEIVED, FIRST WE ARE TO CHECK IF SOURCE IS IN
C     TIMER RECOVERY CONDITION , IF SO THE REJ FRAME WILL BE IGNORED
C     IF SOURCE ISNOT IN TIMER RECOVERY CONDITION , REJ FRAME WILL BE
C     PROCESSED IMMEDIATELY. THE NODE THAT RECEIVES A REJ FRAME SHOULD
C     RETRANSMIT THE REQUESTED I - FRAME  AND ANY OTHER I - FRAMES
C     OUTSTANDING AS SOON AS POSSIBLE. TO DO THAT FIRST, THE RECEIVE
C     SEQUENCE NUMBER IS USED AS AN ACKNOWLEDGEMENT FOR ALL TRANSMITTED
C     I - FRAMES MINUS ONE . SECOND RESET THE TIMER AND IMMEDIATELY
C     START THE PROCESS OF RETRANSMISSION.
C
      IF(ITMOT(INODE).EQ.1)RETURN
C        CHECK IF SOURCE IS  IN TIMEOUT RECOVERY CONDITION,
C        IF SO, THEN RCVSFRM WOULD BE IGNORED
C
      IF(JCHKSTS(INODE).EQ.1)THEN
C        CHECK IF TRANMISSION OF I - FRAME IS IN PROGRESS, IF SO
C         SUSPEND THE TRANSMISSION OF I - FRAME .
C
         IFRM(INODE) = 1
C        SET A CONDITION THAT THE FRAME IN PROGRESS WOULD BE IGNORED
C        BY THE MAC LEVEL ( WILL BE DISCARDED DURING NORMAL TRANSMISSION)
C
         IABRTPKT(INODE) = IABRTPKT(INODE) + 1
C        INCREMENT NUMBER OF I-FRAMES ABORTED DURING TRANSMISSION
C        BECAUSE OF " REJ " ERROR RECOVERY
C
      ENDIF
C
      DO 25 I = 1,20
C
   25    SAVE1(I) = ATRIB(I)
C
C     SAVE THE ATTRIBUTES OF THIS FRAME, WHILE SEARCHING THE
C     EVENT CALENDER
C
      ATRIB(10) = 2.0
C     SET A CONDITION THAT INDICATES FROM WHAT ROUTINE THE SRCH
C     SUBROUTINE IS CALLED.  NOTE THAT FOR EACH RECOVERY CONDITION
C     THE TIMER RESETTING IS DIFFERENT.
C
      CALL SRCH
C     STOP TIMER FOR TRANSMITTED FRAMES
C
      ATRIB(10) = 0.0
C     RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
      ATRIB(19) = ATRIB(16) - 1
C     SET THE ACKNOWLEDGE VARIABLE EQUAL TO RECEIVER'S VARIABLE
```

```
C      MINUS ONE
C
       CALL RMVACKFRM
C      REMOVE THE ACKNOWLEDGED PACKETS FROMS THE REPEAT QUEUE
C
       DO 46 I = 1,20
   46     ATRIB(I) = SAVE1(I)
C         RESTORE THE ATTRIBUTES VALUES
C
       ATRIB(20) = 0.0
C      NON TIMEOUT EVENT
C
C      THE NODE THAT IS RECEIVED THE REJ FRAME IS TO START
C      THE RETRANSMISSION OF REQUESTED I - FRAME(S)
C
       CALL SCHDL(9,0.0,ATRIB)
C      CALL THE REXMIT EVENT TO TAKE CARE OF I - FRAME
C      RETRANSMISSION
C      BEGIN RETRNASMISSION OF ALL OUTSTANDING FRAMES
C
       RETURN
C
    3  CONTINUE
C      RR TIMEOUT INQUIRY STATUS FRAME RECEIVED ,FIRST  CHECK IF THIS
C      FRAME IS A COMMAND OR A RESPONSE FRAME.  IF IS A COMMAND FRAME THE
C      DATA LINK LAYER MUST SEND A RESPONSE TO THE TRANSMITTING NODE AS
C      SOON AS POSSIBLE.  BUT IF RECEIVED FRAME IS A RESPONSE FRAME ,  THE
C      TIMEOUT EVENT WILL BE CALLED TO DEAL WITH THIS FRAME APPROPRIATELY
C
       IF(IFIX(ATRIB(15)).EQ.1)THEN
C         CHECK IF THIS FRAME IS A COMMAND FRAME, IF SO PROCESS THE
C         ACKNOWLEDEGEMENT CONTAINED IN THE FRAME AND IMMEDIATELY
C         BEGIN TO SEND A RESPONSE TO THE NODE WHICH TRANSMITTED
C         THE RR FRAME .
C
          DO 75 I = 1,20
   75        SAVE(I) = ATRIB(I)
C            SAVE THE ATTRIBUTES OF THIS FRAME WHILE IN SEARCHING
C            IN THE EVENT CALENDER AND REPEATE QUEUE
C
          ATRIB(19) = ATRIB(19) - 1
C         DEFINE THE ACKNOWLEDGEMENT SEQUENCE NUMBER ( N(r) - 1
C         PROTOCOL VERSION )
C
          CALL SRCH
C         STOP THE TIMER FOR THE ACKNOWLEDGED FRAME(S)
C
          CALL RMVACKFRM
C         REMOVE THE ACKNOWLEDGED FRAME(S) FROM THE REPEATE QUEUE
C
          DO 80 I = 1,20
   80        ATRIB(I) = SAVE(I)
C            RESTORE THE ATTRIBUTE VALUES
C
          IF(ICHKSTS(INODE).EQ.0)THEN
C            THIS NODE IS TO SEND A RR RESPONSE FRAME TO THE TRANSMITTING
C            NODE, FIRST CHECK IF THIS NODE IS IDLE, IF SO BEGIN THE
C            THE PROCESS TO TRANSMIT THE RR RESPONSE SUPERVISORY FRAME
C            BUT IF SOURCE IS BUSY ,THEN STORE TH RR RESPONSE FRAME
C            IN THE S - FRAME BUFFER .  THIS FRAME WILL BE TRANSMITTED
```

```
C            AS SOON AS NODE BECOME IDLE.
C
             ICHKSTS(INODE) = 1
C            SET THE TRANSMITTER BUSY
C
             ATRIB(15) = 2.0
C            SET THE FRAME TYPE A RESPONSE FRAME
C
             ATRIB(17) = RCVSVR(INODE)
C            DEFINE THE FRAME SEQUENCE NUMBER EXPECTED BY THIS NODE
C
             ATRIB(19) = IRCVCNT(INODE)
C            DEFINE THE ACKNOWLEGE VARIABLE - THIS NODE ACKNOWLEDGES
C            ATRIB(19) - 1 I- FRAMES ( N(r) - 1 PROTOCOL VERSION )
C
             ATRIB(20) = 0.0
C            NON TIMEOUT EVENT
C
             CALL FILEM(NCLNR-2,ATRIB)
C            STORE THIS FRAME IN A TEMPORARY BUFFER , AND BEGIN
C            TRANSMITTING THIS FRAME IMMEDIATELY
C
             PRSSDLY = SNDSPR
C            PROCESSING TIME NEEDED TO SEND A SUPERVISORY FRAME
C
             CALL SCHDL(6,PRSSDLY,ATRIB)
C            ACTIVATE THE TRNSMT PART OF THE DATA LINK LAYER TO
C            PREPARE THIS FRAME FOR TRANSMISSION
C
             RETURN
C
          ELSE
C
             SFRM(INODE) = SFRM(INODE) + 1
C            TRANSMITTER IS BUSY - S - FRAME WAITING FOR TRANSMISSION
C
             ATRIB(15) = 2.0
C            SET THE TYPE OF THIS FRAME TO RESPONSE
C
             ATRIB(17) = RCVSVR(INODE)
C            DEFINE THE SEQUENCE NUMBER OF NEXT FRAME EXPECTED TO BE
C            RECEIVED BY THIS NODE
C
             ATRIB(19) = IRCVCNT(INODE)
C            DEFINE THE ACKNOWLEDGE VARIABLE
C
             ATRIB(20)= 0.0
C            NON TIMEOUT EVENT
C
             CALL FILEM(NCLNR-2,ATRIB)
C            STORE THE RR FRAME IN THE S - FRAME BUFFER, AS SOON AS NODE
C            BECOME IDLE WILL TRANSMIT THIS FRAME
C
             RETURN
C
          ENDIF
C
       ELSEIF(IFIX(ATRIB(15)).EQ.2)THEN
C         CHECK IF THE RECEIVED  FRAME IS A RESPONSE RR TIMEOUT INQUIRY
C         STATUS FRAME , IF SO THE TIMEOUT ROUTINE WILL BE CALLED TO
```

```
C          PROCESS THIS FRAME
C
           ATRIB(20) = 0.0
C          NON TIMEOUT EVENT
C
           CALL SCHDL(8,0.0,ATRIB)
C          CALL THE TIMEOUT EVENT TO PROCESS THE RECEIVED SUPERVISORY
C          FRAME
C
        ENDIF
C
        RETURN
        END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME:  timeout                          EVENT NUMBER:  8
C ------------------------------------------------------------------
C
C PURPOSE :       This routine is invoked when a transmitting node doesn't
C                 receive an acknowledgement for a transmitted I - frame
C                 before the timeout priod expires . Since it can happen
C                 that an I - frame has been correctly received, but the
C                 acknowledgement has either been sent or lost, a RR timeout
C                 inquiry status command frame is sent to the other node
C                 prior to retransmission to avoid a duplication of the
C                 I - frame already sent.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME                TYPE     CLASS   RANGE
C ------------------------------------------------------------------
C params.dat       ! file   ! read  !         !Declare variables and
C                  !        !       !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:       EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C trnsmt, event 2                        rexmit , event 9
C                                        sndsfrm, event 6
C                                        applylyr, event 10
C                                        subroutine srch
C                                        subroutine rmvackfrm
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl            ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE TIMEOUT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, NSUCR, MMFE, I, NNQ, JLOC, JTMP
     1, ITMP, ILOC, IACK, NNRSC
C
      REAL SAVE, PRSSDLY
      DIMENSION SAVE(20)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION
C
      ILOC = IFIX(ATRIB(4))
C     DEFINE THE REPEAT QUEUE OF THE TRANSMITTING NODE
C
      IF(ITMOT(INODE).EQ.1.AND.IFIX(ATRIB(18)).EQ.0)RETURN
C        IF TRANSMITTING NODE IS IN TIMEOUT RECOVERY CONDITION
C        THE SUBSEQUENT TIMEOUTS WILL BE IGNORED UNTIL THE TIMEOUT
C         RECOVERY CONDITION IS OVER .
C
      IF(IFIX(ATRIB(18)).EQ.0)THEN
C        CHECK IF THIS ROUTINE IS INVOKED BEACAUSE OF TIMEOUT EXPIRY
```

```
C       OF AN I - FRAME .  THIS ROUTINE IS ALSO SCHEDULED WHEN IT RECEIVES
C       A RESPONSE FOR THE COMMAND RR FRAME SENT SOME EARLIER TIME.
C
        ITMOT(INODE) = 1
C       TRANSMITTING NODE IS IN TIMER RECOVERY CONDITION, FURTHER
C       TRANSMISSION OF I - FRAMES IS SUSPENDED UNTIL END OF THE
C       TIMER RECOVERY CONDITION
C
        RRSTS(INODE) = RRSTS(INODE) + 1
C       INCREMENT NUMBER OF RR TIMEOUT INQURY STATUS FRAMES SENT BY
C       THIS NODE
C
        IF(JCHKSTS(INODE).EQ.1)THEN
C          CHECK IF TRANSMISSION OF I - FRAME IS IN PROGRESS
C          IF SO ,THE TRANSMITTER WILL SUSPEND THE TRANSMISSION
C          OF I - FRAME AND WILL RETRANSMIT IT AFTER THE TIMER
C          RECORY CONDITION IS OVER
C
           IFRM(INODE) = 1
C          SET A CONDITION THAT WHEN SOURCE NODE IS IN TIMEOUT
C          RECOVERY CONDITION ,THE PACKET ALREADY IN TRANSMISSION
C          PROCESS WILL BE IGNORED BY THE RECEIVER
C
           IABRTPKT(INODE) = IABRTPKT(INODE) + 1
C          INCREMENT NUMBER OF I-FRAMES ABORTED DURING TRANSMISSION
C          BECAUSE OF TIMEOUT RECOVERY CONDITION
C
           ATRIB(19) = PKTCNT(INODE)
C          DEFINE THE ACKNOWLEGE VARIABLE - THIS NODE ACKNOWLEDGES
C          ATRIB(19) - 1 I - FRAMES RECEIVED BY THIS NODE ( N(r) -
C          PROTOCOL VERSION ) .
C
           ATRIB(10) = 1.0
C          SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C          THE SRCH SUBROUTINE IS CALLED.  NOTE THAT FOR EACH
C          RECOVERY CONDITION THE TIMER RESETTING IS DIFFERENT.
C          IN THIS CASE THE TIMER IS STOPPED FOR THE I - FRAME
C          IN TRANSMISSION AND NOT FOR THE FRAMES ALREADY TRANSMITTED
C
           CALL SRCH
C          STOP TIMER FOR THE FRAME IN TRANSMISSION
C
           ATRIB(10) = 0.0
C          RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
        ENDIF
C
        IF(ICHKSTS(INODE).EQ.0)THEN
C          THIS NODE IS TO SEND A RR COMMAND FRAME TO THE OTHER
C          NODE ,FIRST CHECK IF THIS NODE IS IDLE ,IF SO BEGIN
C          THE PROCESS TO TTRANSMIT THE RR COMMAND SUPERVISORY FRAME,
C          BUT IF SOURCE IS BUSY ,THEN STORE THE RR COMMAND FRAME
C          IN THE S - FRAME BUFFER , THIS FRAME WILL BE TRANSMITTED
C          AS SOON AS NODE BECOME IDLE
C
           ATRIB(15) = 1.0
C          SET THE FRAME TYPE TO COMMAND FRAME
C
           ATRIB(18) = 3.0
C          RR TIMEOUT INQUIRY STATUS FRAME INDICATOR
```

```
C
            ATRIB(19) = IRCVCNT(INODE)
C           DEFINE THE ACKNOWLEDGE VARIABLE FOR THE OUTGOING SUPERVISORY
C           FRAME . IT ACKNOWLEDEGES ATRIB(19) -1 I - FRAMES RECEIVED
C           BY THIS NODE
C
            ATRIB(20) = 0.0
C           NON TIMEOUT EVENT
C
            ICHKSTS(INODE) = 1
C           SET THE TRANSMITTER BUSY
C
            CALL FILEM((NCLNR-2),ATRIB)
C           STORE THIS FRAME IN A TEMPORARY BUFFER , AND BEGIN
C           TRANSMITTING THIS FRAME IMMEDIATELY
C
            PRSSDLY = SNDSPR
C           PROCESSING TIME NEEDED TO SEND A SUPERVISORY FRAME
C
            CALL SCHDL(6,PRSSDLY,ATRIB)
C           ACTIVATE THE TRANSMIT PART OF THE DATA LINK LAYER TO
C           PEPARE THIS FRAME FOR TRANSMISSION
C
            RETURN
C
         ELSE
C           SOURCE IS BUSY , STORE THE SUPERVISORY FRAME IN
C           THE S - FRAME BUFFER , THIS FRAME WILL BE
C           TRANSMITTED AS SOON AS NODE BECOME IDLE.
C
            ATRIB(15) = 1.0
C           SET THE FRAME TYPE TO COMMAND FRAME
C
            ATRIB(18) = 3.0
C           RR TIMEOUT INQUIRY STATUS FRAME INDICATOR
C
            ATRIB(19) = IRCVCNT(INODE)
C           DEFINE THE ACKNOWLEDGE VARIABLE FOR THE OUTGOING SUPERVISORY
C           FRAME . IT ACKNOWLEDGES ATRIB(19) - 1 I - FRAMES RECEIVED
C           BY THIS NODE
C
            ATRIB(20) = 0.0
C           NON TIMEOUT EVENT
C
            SFRM(INODE) = SFRM(INODE) + 1
C           SOURCE IS BUSY - S -FRAME WAITING FOR TRANSMISSION
C
            CALL FILEM((NCLNR-2),ATRIB)
C           STORE THE RR COMMAND FRAME IN THE S - FRAME BUFFER , AS
C           SOON AS SOURCE BECOME IDLE ,IT WILL TRANSMIT THIS FRAME
C
            RETURN
C
         ENDIF
C
      ELSEIF(IFIX(ATRIB(18)).EQ.3)THEN
C        A RR TIMEOUT INQUIRY STATUS RESPONSE FRAME RECEIVED,
C        THE DATA LINK LAYER OF THIS NODE WOULD RESET THE
C        TIMER RECOVERY CONDITION VARIABLE AND THEN PROCESS
C        THE ACKNOWLEGEMENT CONTAINED IN THE RECEIVED SUPERVISORY
```

```
C          FRAME.  IF RECEIVED ACKNOWLEDGMENT SEQUENCE NUMBER
C          IS SMALLER THAN TRANSMITTER'S SEND SEQUENCE NUMBER
C          NODE WILL START THE RETRANSMISSION PROCESS OF ALL
C          OUTSTANDING (UNACKNOWLEDGED) I - FRAMES IMMEDIATELY
C
           ITMOT(INODE) = 0
C          RESET THE TIMER RECOVERY VARIABLE
C
           IACK = IFIX(ATRIB(19))
C          SAVE THE ACKNOWLEDGE SEQUENCE NUMBER
C
           DO 90 I = 1,20
   90         SAVE(I) = ATRIB(I)
C
           ATRIB(19) = FLOAT(IACK - 1)
C          ACKNOWLEDGE ALL THE TRANSMITTED I - FRAMES UP TO
C          RECEIVER SEQUENCE NUMBER MINUS ONE
C
           ATRIB(10) = 0.0
C          SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C          THE SRCH SUBROUTINE IS CALLED
C
           CALL SRCH
C          STOP TIMER FOR ALL ACKNOWLEDGED I - FRAMES
C
           CALL RMVACKFRM
C          REMOVE THE ACKNOWLEDEGED I - FRAMES FROM THE REPEAT
C          QUEUE
C
           DO 95 I = 1,20
   95         ATRIB(I) = SAVE(I)
C
           IF(IACK.LE.PKTCNT(INODE))THEN
C             CHECK IF THE RECEIVER SEQUENCE NUMBER IS LESS THAN OR
C             EQUAL TO SENDER'S SEND SEQUENCE NUMBER, IF SO THEN
C             SENDER WOULD RETRANSMIT ALL OUTSTANDING I - FRAMES
C
              ITMOUTCNT(INODE) = ITMOUTCNT(INODE) + 1
C             NODE IS IN TIMEOUT ERROR RECOVERY CONDITION , THEREFORE
C             INCREMENT NUMBER OF TIMEOUTS BY THIS NODE
C
              ATRIB(10) = 2.0
C             SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C             THE SRCH SUBROUTINE IS CALLED
C
              CALL SRCH
C             SOURCE IS ABOUT TO RETRANSMIT ALL OUTSTANDING I - FRAMES
C             RESET THE TIMER FOR ALL I _ FRAMES
C
              ATRIB(10) = 0.0
C             RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
              CALL SCHDL(9,0.0,ATRIB)
C             BEGIN THE RETRANSMISSION OF ALL OUTSTANDING FRAMES
C
           ELSEIF(NNRSC(INODE).LT.1)THEN
C             THE OTHER NODE ACKNOWLEGES ALL FRAMES SENT BY THIS NODE
C             INFORM THE APPLICATION LAYER THAT DATA LINK LAYER IS
C             READY TO ACCEPT MORE I - FRAMES
C
```

```
            ATRIB(8) = 1.0
C           SET A CONDITION THAT TELLS APPLICATION LAYER THAT DATA
C           LINK LAYER IS EXPECTING I - FRAMES
C
            ATRIB(20) = 0.0
C           NON TIMEOUT EVENT
C
            PRSSDLY = BMOVE + ENQUEU + RCVCOM
C           PROCESSING TIME TO SEND A I - FRAME FROM APPLICATION
C           LAYER TO THE DATA LINK LAYER
C
            CALL SCHDL(10,PRSSDLY,ATRIB)
C           SCHDL APPLICATION LAYER
C
         ENDIF
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rexmit                              EVENT NUMBER: 9
C ------------------------------------------------------------------
C
C PURPOSE :      This routine is to search the repeat queue,and find
C                the requested frame or a frame which caused the timeout.
C                After finding that frame ,it will set the transmitter
C                send seq # equal to the seq # of the frame that just
C                found in thwe repeat queue. Then will invoke the source
C                transmitter to transmit this frame and any other frames
C                outstanding.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C timeout, event 8                          trnsmt, event 2
C rcvsfrm, event 7
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C copy            ! A SLAM routine that copies the attributes of an entity
C                 ! into the attribute array
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE REXMIT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE,JNODE,NEXT,NSUCR,ILOC,MMFE,IRCVSQ,NNRSC,NNQ
C
      REAL PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(4))
C     DEFINE THE REPEAT QUEUE OF THE NODE WHICH
C     TRANSMITTED THE FRAME
C
      IF(IRXSTS(INODE).EQ.0)THEN
C        CHECK IF THIS NODE IS ALREADY IN RETRANSMISSION CONDITION
C        IF NOT SET THIS NODE IN RETRANSMIT STATUS
C
         NDSTS(INODE) = 1
C        SET THE SOURCE TRANSMITTER IN RETRANSMIT CONDITION (BUSY)
C
      ELSE
C        SOURCE ISNOT IN RETRANSMIT CONDITION
```

```
C
          BFRCNT(INODE) = NNQ(ILOC)
C         DEFINE NUMBER OF OUTSTANDING FRAMES THAT ARE TO BE
C         RETRANSMITTED
C
       ENDIF
C
       IRCVSQ = IFIX(ATRIB(17))
C      SAVE THE RECEIVER SEQ #
C
C      BEGIN SEARCH THE REPAET QUEUE ,AND FIND THE ENTRY NUMBER FOR
C      THE REQUESTED PACKET
C
       NEXT = MMFE(ILOC)
C      START THE SEARCH FOR THE REQUESTED PACKET FROM THE BEGINING
C      OF THE REPEAT QUEUE
C
C      SEARCH THE REPEAT QUEUE,AND FIND THE REQUESTED ENTRY
C
    20 IF(NEXT.EQ.0)THEN
C         END OF REPEAT QUEUE IS REACHED , NO ENTRY FOUND
C         IN THIS BUFFER WITH GIVEN INFORMATION , THEREFORE
C         IGNORE THIS SUBROUTINE AND RETURN
C
C         SINCE FOUND NO ENTRY WITH THE GIVEN INFORMATION
C         RESET ALL THE RETRANSMIT VAR.  FOR THIS NODE
C
          IFRM(INODE) = 0
C         RESET THE VARIABLE THAT INDICATES AN I-FRAME SHOULD BE
C         SUSPENDED OR NOT
C
          BFRCNT(INODE) = 0
C         RESET THE NUMBER OUTSTANDING I-FRAMES WAITING FOR TRANSMISSION
C
          NDSTS(INODE)=0
C         RESET NODE STATUS
C
          IRXSTS(INODE) = 0
C         RESET RETRANSMISSION VARIABLE
C
          ATRIB(8) = 1.0
C         APPLICATION LAYER INDICATOR, LAP ASKS APPLICATION LAYER
C         TO SEND I-FRAMES TO LLC LEVEL
C
          ATRIB(20) = 0.0
C         NON TIMEOUT EVENT
C
          PRSSDLY = BMOVE + ENQUEU + RCVCOM
C         PROCESSING TIME NEEDED TO SEND I-FRAME TO THE LLC LAYER
C
          CALL SCHDL(10,PRSSDLY,ATRIB)
C         SCHDL APPLICATION LAYER
C
          RETURN
C
       ENDIF
C
       CALL COPY(-NEXT,ILOC,ATRIB)
C      COPY THE ATTRIBUTES OF THIS ENTRY INTO THE ATTRIBUTE ARRAY
C
```

```
      IF(IFIX(ATRIB(6)).EQ.IRCVSQ)THEN
C        FOUND THE ENTRY THAT WAS SEARCHING FOR NOW START
C        THE PROCESS OF RETRNASMISSION
C
         IRXSTS(INODE) = 1
C        SET THE SOURCE IN RETRNASM.  STATUS
C
         KTMP(INODE) = 1
C        SET A CONDITION THAT TELLS THE SOURCE THE START OF
C        RETRANSMISSION OF I-FRAMES.  SO SOURCE WOULD NOT INCR.
C        THE SEND SEQ # AT THE FIRST TIME.
C
         ISND(INODE) = IFIX(ATRIB(6))
C        RESET THE SEND SEQ # EQUAL TO THE SEQ # OF
C        REQUESTED FRAME
C
         IF(ICHKSTS(INODE).EQ.0)THEN
C           CHECK IF THE TRANSMITTER IS IDLE
C
            BFRCNT(INODE) = NNQ(ILOC)
C           # OF OUTSTANDING I-FRAMES TO BE RETRANSMITTED
C
            ICHKSTS(INODE) = 1
C           SET THE TYRANSMITTER BUSY
C
            NDSTS(INODE) = 2
C           SET THE NODE IN RETRANSMISSION STATUS
C
            CALL SCHDL(2,0.0,ATRIB)
C           START THE RETRANSMISSION PROCESS IMMEDIATELY
C
         ENDIF
C
      ELSE
C
         NEXT = NSUCR(NEXT)
C        PROCEED TO THE NEXT ENTRY OF THIS FILE
C
         GO TO 20
C        CONTINUE THE SEARCH FOR THE ENTRY IN QUESTION
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: applclyr
C ------------------------------------------------------------------
C
C PURPOSE :       This routine implements the user level in a Local
C                 Area Network. It performs following functions:
C                 First , releases packets from the wait node ( user buffer)
C                 and send this frame to the Link Layer. Second receives
C                 error free  information frames from the lower layers. Finally
C                 collects statistics on the ave network delay and ave node
C                 to node delay
C                 channel array to determine the node's perception of the
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file   ! read !         !Declare variables and
C                 !        !      !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:      EVENTS SCHEDULED BY THIS ROUTINE
C ------------------------------------------------------------------
C laptrns, event 3
C trnsmt, event 2
C rexmit, event 9
C timeout, event 8
C rcvifrm, event 5
C subroutine rmvackfrm
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C colct     !  A SLAM routine that collects statistics on variables.
C ------------------------------------------------------------------
      subroutine applclyr
c
      include ' params.dat '
c
      integer inode,nnrsc
      real tmdly
c
      inode = ifix(atrib(2))
c     set the source node
c
      if(ifix(atrib(8)).eq.1)then
c        check if applclyr is asked to release frames to lower layer
c
         if(nnrsc(inode).lt.1)then
c           check if this node has any resource available
c           before it can free a packet from the user buffer
c
            call free(inode,1)
c           free one packet from the user buffer
c
         endif
c
      elseif(ifix(atrib(8)).eq.2)then
c         check if application layer is receiving a frame
```

```
c           from the lower layer
c
            xx(115) = xx(115) - 1.0
c           decrement a packet from total packets in the system
c
            tmdly = tnow - atrib(1)
c           calculate user to user delay
c
            call colct(tmdly,(inode+6))
c           collect delay statistics for this node
c
            call colct(tmdly,29)
c           collect delay statistics for total system delay
c
        endif
c
        return
        end
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: sndrcvack
C ------------------------------------------------------------------
C
C PURPOSE :     This subroutine takes the necessary action for receiving
C               and sending acknowldegements. When a node receives an
C               I - frame ,it containes a piggybacked acknowledgement
C               which is used to ack. all I - frames transmitted by this
C               node . If a node has no I - frame to send ( can't piggyback
C               the acknowledgement) then would send a RR supervisory
C               frame to acknowledge the I - frame just received by this
C               node.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS ROUTINE:     EVENTS SCHEDULED BY THIS ROUTINE:
C ------------------------------------------------------------------
C rcvifrm, event 5                        sndsfrm, event 6
C                                         subroutine srch
C                                         subroutine rmvackfrm
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
C
      SUBROUTINE SNDRCVACK
C
      INCLUDE ' PARAMS.DAT '
C
      INTEGER I,INODE,JNODE,ITMP,NNQ
C
      REAL SAVE1,SAVE2,PRSSDLY
      DIMENSION SAVE1(20),SAVE2(20)
C
      INODE = ATRIB(2)
C     DEFINE THE SOURCE NODE
C
      JNODE = ATRIB(3)
C     DEFINE THE DESTINATION
C
      DO 10 I = 1,20
   10    SAVE2(I) = ATRIB(I)
C        SAVE THE ATTIRBUTES FOR THIS FRAME
C
      ATRIB(2) = FLOAT(JNODE)
C     CHANGE THE STATUS OF THIS NODE FROM RECEIVER TO TRANSMITTER
C     THE NODE THAT WAS RECEIVING PACKET BECOMES A TRANSMITTER
C
      ATRIB(3) = FLOAT(INODE)
C     THE NODE THAT WAS TRANSMITTING BECOMES A RECEIVER
C
```

```
      ATRIB(4) = ATRIB(2) + MAXSTA
C     DEFINE THE REPEAT QUEUE BASED ON THE TRANSMITTER
C
      IF(IFIX(ATRIB(19)).EQ.0)GO TO 20
C        CHECK IF THIS IS THE FIRST I - FRAME RECEIVED BY THIS NODE
C        IF SO I - FRAME CONTAINES NO ACKNOWLEDGEMENT THEREFORE
C        BEGIN THE PROCESS TO SEND A RR FRAME
C
      DO 30 I = 1,20
   30    SAVE1(I) = ATRIB(I)
C
      IF(IFIX(ATRIB(19)).GT.IAKVAR(JNODE))THEN
C        CHECK IF THE RECEIVED ACKNOWLEDGMENT IS GREATER THAN
C        ACKNOWLEDGEMENT STATE VARIABLE ( V(a) PROTOCOL VERSION ), IF
C        SO PROCESS THE RECEIVED ACKNOWLEDGEMENT.
C
         CALL SRCH
C        REMOVE THE TIMEOUT EVENT FOR THE ACK. PACKET FROM THE
C        EVENT CALENDER
C
         CALL RMVACKFRM
C        REMOVE THE ACK. PACKET FROM THE REPEAT QUEUE
C
      ENDIF
C
      DO 35 I = 1,20
   35    ATRIB(I) = SAVE1(I)
C
C
   20 CONTINUE
C     CHECK IF THIS NODE HAS NO I - FRAME TO SEND , IF SO BEGIN TO
C     SEND A RR FRAME WHICH ACKNOWLEDGES THE FRAME JUST RECEIVED
C     BY THIS NODE
C
      IF(NDSTS(JNODE).LE.1)THEN
C        CHECK IF THE RECEIVING NODE IS IN NORMAL I - FRAME TRANSMISSION
C        CONDITION, IF SO TAKE THE PROPER STEPS TO SEND OR NOT TO SEND
C        A RR FRAME
C
         IF(JFRM(JNODE).LE.0.AND.IFIX(ATRIB(14)).EQ.0.AND.NNQ(JNODE)
     1      .LE.0)THEN
C           THIS NODE HAS NO FRAMES WAITING FOR TRANSMISSION
C           TAKE THE NECESSARY STEPS TO TRANSMIT A "RR" FRAME
C           TO THE OTHER NODE . THE RR FRAME WOULD ACKNOWLEDGE
C           ALL THE PACKETS UP TO RECEIVER'S SEQ. NUMBER MINUS
C           ONE. THE RR FRAME WOULD BE SENT TO THE OTHER NODE
C           IF AND ONLY IF A PACKET HAS BEEN RECEIVED CORRECTLY
C
            IF(ICHKSTS(JNODE).EQ.0.AND.ITMOT(JNODE).EQ.0)THEN
C              MAKE SURE THE SOURCE ISNOT TRANSMITTING AND ALSO
C              ISNOT IN TIMER RECOVERY CONDITION
C
               ATRIB(18) = 2.0
C              RR SUPERVISORY FRAME INDICATOR
C
               ATRIB(19) = ATRIB(5)
C              SET THE SEQUENCE NUMBER OF LAST PACKET RECEIVED CORRECTLY
C              EQUAL TO THE ACKNOWLEDGE VARIABLE
C
               ATRIB(20) = 0.0
```

```
C                NON TIMEOUT EVENT
C
                 ICHKSTS(JNODE) = 1
C                SET THE TRANSMITTER BUSY
C
                 PRSSDLY = ACKPKT
C                PROCESSING DELAY TO SEND A SUPERVISORY FRAME
C
                 CALL SCHDL(6,PRSSDLY,ATRIB)
C                START SENDING THE RR-FRAME
C
              ENDIF
C
           ENDIF
C
        ELSEIF(NDSTS(JNODE).EQ.2)THEN
C          THE RECEIVING NODE IS IN RETRANSMIT STATUS, IF THIS NODE HAS NO
C          OUTSTANDING FRAMES THEN PREPARE TO SEND A RR FRAME TO ACKNOW.
C          I - FRAME JUST RECEIVED BY THIS NODE
C
           IF(ITMOT(JNODE).EQ.0.AND.IFIX(ATRIB(14)).LE.0)THEN
C             MAKE SURE THAT THIS NODE IS NOT IN TI ER RECOVERY CONDITION
C             AND THE RECEIVED I - FRAME IS VALID ( NO TRANSMISSION ERROR
C             OR OUT OF SEQUENCE)
C
              IF(ICHKSTS(JNODE).EQ.0)THEN
C                MAKE SURE THE TRANSMITTER ISNOT BUSY
C
                 ATRIB(18) = 2.0
C                RR SUPERVISORY FRAME INDICATOR
C
                 ATRIB(20) = 0.0
C                NON TIMEOUT EVENT
C
                 ATRIB(19) = ATRIB(5)
C                SET THE SEQUENCE NUMBER OF LAST PACKET RECEIVED
C                CORRECTLY EQUAL TO THE ACKNOWLEDGE VARIABLE
C
                 ICHKSTS(JNODE) = 1
C                SET THE TRANSMITTER BUSY
C
                 PRSSDLY = ACKPKT
C                PROCESSING DELAY TO SEND A SUPERVISORY FRAME
C
                 CALL SCHDL(6,PRSSDLY,ATRIB)
C                START SENDING THE RR-FRAME
C
              ELSEIF(BFRCNT(JNODE).LE.1)THEN
C                SINCE TRANSMITTER IS BUSY AND NODE IS IN RETRANSMIT
C                STATUS, CHECK IF THIS IS THE LAST I-FRAME BEING
C                RETRANSMITTED BY THIS NODE
C
C                TRANSMITTER IS BUSY, STORE THE SUPREVISORY FRAME IN
C                A TEMPORARY BUFFER, AS SOON AS SOURCE RETURNS IDLE
C                WILL TRANSMIT THIS FRAME.
C
                 SFRM(JNODE) = SFRM(JNODE) + 1
C                S-FRAME WAITING FOR TRANSMISSION
C
                 ATRIB(18) = 2.0
```

```
C                S-FRAME INDICATOR
C
                 ATRIB(19) = ATRIB(5)
C                SET THE ACKNOWLEDGE VARIABLE
C
                 ATRIB(20) = 0.0
C                NON TIMEOUT EVENT
C
                 CALL FILEM(NCLNR-2,ATRIB)
C                STORE THE SUPREVISORY FRAME IN THE S-FRAME BUFFER

              ENDIF
C
           ENDIF
C
        ENDIF
C
        DO 40 I = 1,20
   40      ATRIB(I) = SAVE2(I)
C          RESTORE THE ATTRIBUTE VALUES
C
        RETURN
        END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: rmvackfrm
C
C PURPOSE :       This routine is called when a frame have arrived correctly
C                 at the receiver. This subroutine removes the acknowledged
C                 frame from the repeat queue. Aslo collects statistics on the
C                 average buffer holding time. Finally if the source node is in
C                 retransmit status and has finished retransmitting all
C                 outstanding I - frames would reset the retransmit variables
C                 and schedules the application layer event to send more I -frames
C                 to the data link layer.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME                TYPE     CLASS  RANGE
C ------------------------------------------------------------------
C                  |        |       |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat       | file   | read  |         | Declare variables and
C                  |        |       |         | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C applclyr, event 10                 subroutine, sndrcvack
C                                    rcvsfrm, event 7
C                                    timeout, event 8
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C colct            | SLAM routine to collect statistics on variables.
C rmove            | SLAM routine to remove  an entity from a file
C copy             | SLAM routine to copy attributes of an entity from
C                  | a file
C------------------------------------------------------------------
      SUBROUTINE RMVACKFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, MMFE, NSUCR, ISQNM, NNG, ILOC
     1, NNRSC, JLOC, ITMP, JTMP
C
      REAL BFHLDTM, PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(ATRIB(3)))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(4))
C     DEFINE THE REPEAT QUEUE OF THE NODE
C     WHICH TRANSMITTED THE FRAME
C
      ISQNM = IFIX(ATRIB(19))
C     DEFINE THE ACK. FRAME
C
```

```
      ITMP = NNQ(ILOC)
C     DEFINE # OF OUTSTANDING FRAMES IN THE REPEAT QUEUE
C
      JTMP = 0
C
C     BEGIN TO REMOVE THE ACK. PACKETS FROM THE REPEAT QUEUE, SEARCH
C     THE ENTIRE REPEAT QUEUE FOR ACK. PACKETS AND TAKE THE DELAY
C     STAT. OF THE ACK. PKTS
C
      DO WHILE(JTMP.LE.ITMP)
C
         NEXT = MMFE(ILOC)
C        SET THE POINTER " NEXT " TO THE FIRST LOCATION IN
C        THE RETRANSMISSION BUFFER WHERE THE ACK. FRAME IS STORED
C
    5    IF (NEXT.EQ.0)RETURN
C             NO ENTRIES IN THIS FILE
C
         CALL COPY(-NEXT, ILOC, ATRIB)
C        COPY THE ATTRIBUTES OF THE ENTRY POINTED TO BY
C        NEXT INTO THE ATTRIBUTE ARRAY.
C
         IF(IFIX(ATRIB(5)).LE.ISQNM)THEN
C           CHECK IF COPIED ENTRY IS LESS THAN OR EQUAL TO THE ACK.
C           PACKET
C
            CALL RMOVE(-NEXT, ILOC, ATRIB)
C           REMOVE THIS ENTRY FROM THE REPEAT QUEUE SINCE
C           THIS PACKET HAS BEEN ACKNOWLEDGED
C
            BFHLDTM = TNOW - ATRIB(1)
            BFHLDTM = BFHLDTM + DEQUEU
C           CALCULATE THE ER HOLDING TIME FOR THIS PACKET
C
            CALL COLCT(BFHLDTM, 30)
C           COLLECT STATS. ON THE TOTAL AVE. BUFFER HOLDING TIME
C
            IAKVAR(INODE) = IFIX(ATRIB(5))
C           UPDATE THE ACKNOWDLEGE VARIABLE
C
            IF(IRXSTS(INODE).EQ.1.AND.NNQ(ILOC).LE.0)THEN
C              CHECK IF THIS NODE IS IN RETRANSMIT STATUS, IF
C              TRUE ALSO CHECK IF THE LAST FRAME HAS BEEN ACK.
C              BEFORE RESETTING THE RETRANSMIT VARIABLES.
C
               BFRCNT(INODE) = 0
C              RESET # OF OUTSTANDING FRAMES FOR TRANSMISSION VARIABLE
C
               IRXSTS(INODE) = 0
C              RESET THE RETRANSMIT CONDITION FOR THIS NODE
C
               NDSTS(INODE) = 0
C              SET THE SOURCE NODE IDLE
C
               ATRIB(8) = 1.0
C              APPLICATION LAYER INDICATOR, LAP EXPECTS MORE I-FRAMES
C
               ATRIB(20) = 0.0
C              NON TIMEOUT EVENT
C
```

```
                PRSSDLY = BMOVE + ENQUEU + RCVCOM
C               PROCESSING TIME FOR APPLICATION LAYER TO SEND THE NEXT
C               I-FRAME TO LLC LAYER
C
                CALL SCHDL(10,PRSSDLY,ATRIB)
C               SCHDL APPLICATION LAYER
C
             ENDIF
C
          ELSE
C
C            THE ENTRY COPIED ISNOT THE ACKNOWLEDEGED PACKET,
C            CONTINUE THE SEARCH UNTIL THAT PACKET IS FOUND
C
             NEXT = NSUCR(NEXT)
C            POINT TO THE NEXT ENTRY IN REPEAT QUEUE
C
             GO TO 5
C
          ENDIF
C
          JTMP = JTMP + 1
C
       ENDDO
C
       RETURN
       END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: srch
C
C PURPOSE :      This routine removes scheduled TIMEOUT event(s) from
C                the event calendar in the event of an acknowledged
C                I-frame, or timeout recovery, or REJ recovery.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |        | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |        | Declares variables and
C                   |       |      |        | establishes the common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C (none)                             subroutine sndrcvack
C                                    rcvsfrm,  event 7
C                                    timeout,  event 10
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C copy              | SLAM routine that copies attributes from elements in
C                   | in a file.
C rmove             | SLAM routine that removes entries from the files.
C ------------------------------------------------------------------
      SUBROUTINE SRCH
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, MMFE, NSUCR, IRCV, I, NNQ, JTMP, JLOC
     1, IMODE, ITMP
C
      REAL SAVE
      DIMENSION SAVE(20)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(3))
C     DEFINE THE DESTINATION NODE
C
      IRCV = IFIX(ATRIB(19))
C     DEFINE THE ACK. FRAME
C
      IMODE = IFIX(ATRIB(10))
C     INDICATE FOR WHAT RECOVERY CONDITION THIS SUBROUTINE
C     IS CALLED:  IMODE = 0 - ACKNOWLEDGE I-FRAME(S)
C                 IMODE = 1 - TIMEOUT RECOVERY CONDITION
C                 IMODE = 2 - REJ RECOVERY CONDITION
C
      DO 5 I = 1,20
    5    SAVE(I) = ATRIB(I)
C
```

```
      JLOC = NNQ (NCLNR)
C     DEFINE THE BUFFER CONTENTS OF THE EVENT CALENDER
C
      JTMP = 0
C
C     BEGIN TO REMOVE THE TIMEOUT EVENT FOR THE ACK. PACKET(S) FROM
C     THE EVENT CALENDER
C
      DO WHILE(JTMP.LE.JLOC)
C
         NEXT = MMFE(NCLNR)
C        SET THE POINTER " NEXT " TO THE FIRST LOCATION IN THE EVENT
C        CALENDER , WHICH IS THE NEXT EVENT DUE TO OCCUR
C
   10    IF (NEXT.EQ.0)THEN
C           THERE ARE NOT ANY EVENTS DUE TO OCCUR AND THE ROIUTINE
C           SHOULD NOT HAVE BEEN CALLED
C
            DO 7 I = 1,20
    7          ATRIB(I) = SAVE(I)
C
            RETURN
C
         ENDIF
C
         CALL COPY(-NEXT,NCLNR,ATRIB)
C        COPY THE ATTRIBUTES OF THIS ENTRY INTO THE ATTRIBUTE ARRAY
C
         ITMP = IFIX(ATRIB(2))
C        SAVE THE TRANSMITTING NODE
C
         IF(IMODE.EQ.0.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(20)).EQ.1
     1       .AND.IFIX(ATRIB(5)).LE.IRCV)THEN
C           CHECK IF THIS ENTRY IS AN ACKNOWLEDGED  I-FRAME
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           REMOVE THE TIME OUT EVENT FOR THE ACKNOWLEDGED
C           I-FRAME FROM THE EVENT CALENDAR
C
         ELSEIF(IMODE.EQ.1.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(20)).EQ.1
     1       .AND.IFIX(ATRIB(5)).EQ.IRCV)THEN
C           CHECK IF THIS ENTRY IS FOR TIMEOUT RECOVERY
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           TIMEOUT RECOVERY CONDITION, STOP THE TIMER FOR I-FRAME
C           IN TRANSMISSION AND NOT FOR THE FRAMES ALREADY TRANSMITTED
C
         ELSEIF(IMODE.EQ.2.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(20)).EQ.1
     1       )THEN
C           CHECK IF THIS ENTRY IS FOR REJ RECOVERY CONDITION
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           REJ RECOVERY CONDITION, STOP THE TIMER FOR ALL TRANSMITTED
C           I-FRAMES
C
         ELSE
C           THE ENTRY COPIED OUT ISNOT A TIMEOUT EVENT
C           SCHEDULED BY SOURCE NODE SO WE CONTINUE THE SEARCH
C
            NEXT = NSUCR(NEXT)
```

```
C            PROCEED TO THE NEXT ENTRY IN THE EVENT CALENEDER,
C            CONTINUE THE SEARCH UNTIL THE DESIRED ENTRY IS FOUND
C
             GO TO 10
C
          ENDIF
C
          JTMP = JTMP + 1
C
        ENDDO
C
        DO 8 I = 1,20
      8    ATRIB(I) = SAVE(I)
C
        RETURN
        END
```

```
C -------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C -------------------------------------------------------------------
C PROGRAM NAME: otput.for      AUTHOR: M.Kashefi          DATE: 12-15-85
C -------------------------------------------------------------------
C VERSION NUMBER : 1.0           AMENDMENT DATE(S) :
C -------------------------------------------------------------------
C PURPOSE :     Report specific performance indicators as part of the
C               TISL LAP-D simulation system.
C -------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME            ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C -------------------------------------------------------------------
C                  !       !      !         ! (none)
C -------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C -------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C -------------------------------------------------------------------
C                  ! otputlap print user statistics for LAPD model
C -------------------------------------------------------------------
c       This subroutine is part of SLAM summary reporting and involves a
c       call to another routine called output which has got the actual
c       summary reporting program.
c
        subroutine otput
c
        call otputlap
        return
        end
c       This routine will enable the user to add other output routines
c       while integrating the layers.
```

```
      SUBROUTINE OTPUTLAP
C
      INCLUDE ' PARAMS.DAT '
      INTEGER I
      REAL THRUPUT,LOAD
      DIMENSION LOAD(25)
C
C     IF(NNRUN .LT.  IFIX(XX(27))) RETURN
C
      WRITE(NPRNT,1) XX(111)
    1 FORMAT(/,' TIMEOUT PERIOD = ',F14.2,4X,' MICROSEC ')
C
      WRITE(NPRNT,2)XX(113)
    2 FORMAT(/,' BIT ERROR RATE = ',F11.9)
C
      DO 35 I = 1, MAXSTA
C
      WRITE(NPRNT,3)I,TOTPKTCNT(I)
    3 FORMAT(////,' TOTAL # OF PACKETS TRANSMITTED FROM NODE ',I3
     1,' = ',I7)
C
      WRITE(NPRNT,5)I,PKTCNT(I)
    5 FORMAT(/,' TOTAL PACKETS GENERATED FROM NODE ',I3,' = ',I6)
C
      WRITE(NPRNT,6)I,IABRTPKT(I)
    6 FORMAT(/,' TOTAL PACKETS ABORTED DURING TRANSMISSION FROM NODE
     1',I3,' =',I5)
C
      WRITE(NPRNT,10)I,ICOUNT(I)
   10 FORMAT(/,' # OF PACKETS IN ERROR FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,15)I,IOTSQCNT(I)
   15 FORMAT(/,' # OF OUT SEQUENCE PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,20)I,IDMGPKTCNT(I)
   20 FORMAT(/,' # OF DAMAGED PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,25)I,IRXMTCNT(I)
   25 FORMAT(/,' # OF RETRANSMITTED PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,30)I,ITMOUTCNT(I)
   30 FORMAT(/,' # OF TIME OUTS BY NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,40)I,INFOBTS(I)
   40 FORMAT(/,' TOTAL NUMBER OF INFORMATION BITS TRANSMITTED FROM
     1 NODE ',I3,' = ',I13)
C
      WRITE(NPRNT,50)I,RRFRMCNT(I)
   50 FORMAT(/,' TOTAL NUMBER OF "RR" PACKETS TRANSMITTED FROM
     1 NODE ',I3,' = ',I5)
C
      WRITE(NPRNT,60)I,SFRMCNT(I)
   60 FORMAT(/,' TOTAL NUMBER OF  "S" PACKETS TRANSMITTED FROM
     1 NODE ',I3,' = ',I5)
C
      WRITE(NPRNT,65)I,RRSTS(I)
   65 FORMAT(/,' TOTAL NUMBER OF " RR TIMEOUT INQUIRY  STATUS " PACKETS
     1 TRANSMITTED FROM NODE',I3,' =',I5)
C
      LOAD(I) = (INFOBTS(I) * ( 1/XX(75))) / XX(25)
```

```
C       CALCULATE THROUGHPUT FOR THIS NODE
C
        THRUPUT = THRUPUT + LOAD(I)
C       CALCULATE TOTAL THROUGHPUT
C
         WRITE(NPRNT,45)I,LOAD(I)
   45    FORMAT(//,' THROUGHPUT FOR NODE ',I3,' = ',F8.5)
C
   35   CONTINUE
C
        WRITE(NPRNT,55)THRUPUT
   55   FORMAT(////,' TOTAL THROUGHPUT = ',F9.6)
C
        RETURN
        END
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: alloctrsc
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:  when
C                source is not in retransmit status/max window/timeout
C                recovery, then it will seize a resource and lets an
C                entity proceed from network model to the discrete event
C                model
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !         !Declare variables and
C                 !       !      !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:      EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C SLAM                                                  none
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C seize           ! A SLAM routine that allocates resoource to a particular
C                 ! node
C ------------------------------------------------------------------
      SUBROUTINE ALLOC(I,IFLAG)
C
      INCLUDE 'PARAMS.DAT'
      INTEGER NNQ,NNRSC,ILOC,IFLAG,I,JLOC,KLOC
C
      IFLAG = 0
C
      ILOC = IWNDSZ
C     DEFINE WINDOW SIZE
C
      JLOC = I + MAXSTA
C     DEFINE THE REPEAT QUEUE FILE NUMBER
C
      KLOC = NNQ(JLOC)
C     DEFINE NUMBER OF I-FRAMES IN THE REPEAT QUEUE
C
C     ALLOCATE RULE  - SEIZE RESOURCE FOR SOURCE NODE
      IF(NNRSC(I).LE.0.OR.NDSTS(I).EQ.2.OR.KLOC.EQ.ILOC.OR.
     1ITMOT(I).EQ.1) RETURN
C
        CALL SEIZE(I,1)
        IFLAG = -1
C
      RETURN
      END
```

```
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: userf.for      AUTHOR: v.s.frost        DATE: 8-15-85
C ------------------------------------------------------------------
C VERSION NUMBER : 1.1         AMENDMENT DATE(S) : 9-1-85 by b.j.wood
C ------------------------------------------------------------------
C PURPOSE :     This function calculates packet lengths for
C               packets created in the TISL CSMA/CD simulation.
C ------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME           ! TYPE   ! CLASS! RANGE   !     DESCRIPTION
C ------------------------------------------------------------------
C imode           !integer! read  ! 1,2,3   ! Determines which rule wil be
C                 !        !      !         ! used to calculate packet size
C ------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat      ! file   !      !         ! Declare variables and
C                 !        !      !         ! establish common.
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------
C                 ! (none)
C ------------------------------------------------------------------
c
c       This function calculates the length of data packets created in
c       the SLAM driver file.  Userf has three modes as described be-
c       low.  In each mode, the packet size is calculated and the
c       network overhead (xx(83)) is added.  This packet size is then
c       made to conform to the maximum and minimum data packet size
c       permitted by the network.  In each case, the packet size that
c       is returned is in units of BITS!  The average data packet size
c       is given in the SLAM driver file as xx(inode) where inode is
c       the node number of the node that is creating this packet.
c
        function userf(imode)
c
        include 'params.dat'
        integer inode,nnow,imode
        real avelen,userf,xleng,xrand,unfrm,expon
c
c               Get average length from the xx array for the current node.
c
        inode = ifix(atrib(2))
        avelen = xx(inode)
c
c                Initialize the bit counter
c
c                Go to appropriate mode code.
c
        go to(1,2)imode
c
c
c       Mode 1: Fixed length packets.
c
    1   userf = avelen + xx(83)
        if(userf.gt.maxsz) userf = maxsz
        if(userf.lt.minsz) userf = minsz
```

```
      return
c
c
c     Mode 2: Exponential length data packets
c
    2 userf = expon(avelen,1) + xx(83)
      if(userf.gt.maxsz) userf = maxsz

      if(userf.lt.minsz) userf = minsz
c
      return
c
c
      end
```

APPENDIX 2

The purpose of this appendix is to describe the Network Model and the routines that make up the Discrete Event portion of the TISL Integrated LAP-D/CSMA-CD LAN Model. Here, we will describe SLAM Network Model and routines using flow diagram and word descriptions. The actual routines are shown at the end of this appendix.

As we mentioned is previous chapters, the LAN Model developed here is written in simulation language called SLAM[2](Simulation Language for Alternative Modeling). For this effort the sole interface between the user and simulation is the SLAM driver file as shown in figure 6.1. Details of pertinent SLAM statements will be discussed here. For further details see [2]. The model as represented in the SLAM driver file contains five segments:

1. Parameter definition
2. Statistics file definition
3. Traffic generation
4. Entry into the network
5. Execution parameters

In figure 6.1 lines 4-155 contain the parameter definitions. The traffic timing, simulation run, and network parameters are set using the SLAM INTLC statement and SLAM user xx(.) array. The INLTC statement is used to assignn initial values to SLAM variables, in this case the xx(.) array.

The details of the xx(.) array are not significant only the meaning of each element. The meaning of these variables will be discussed in the following sections.

Lines 159-192 inform SLAM processor that statistics will be collected for each node. The statistics of interest are the average number of I-frames in the system, access delay, channel delay, user to MAP queuing delay, user to LAP queuing delay, average MAP delay, average LAP delay, and average user-to-user delay for all the stations in the network.

The generation of traffic is accomplished in lines 194-643. The attribute array, atrib(.), is described first. Each packet which is generated has an attribute array associated with it. This array describes the state of the packet as it progresses through the system. The user of this model need not be concerned with most of values of the attribute array. However, atrib(2) holds source identity, atrib(9) holds packet length in microseconds, atrib(11) holds packet length in bits, atrib(25) holds the destination identity, atrib(26) holds the queue number of the retransmission buffer, atrib(29) holds the packet timeout period, and atrib(36) determines if a node is a background stations or transport station.

After the packet was generated (SLAM CREATE statement), it is queued in the wait node where it is stored until the seizure of resource became available through SLAM ALLOC function. Then packets would enter the network where the specifics of LAPD-D and CSMA/CD protocols are implemented. This done in lines 619-632.

The last aspect of the model deals with SLAM execution parameters i.e., how the simulation is to run. this is done in lines 1,31-41, and 639-644.

For further description of SLAM statements given in figure 6.1 see chapter five of [1]. Now describe the Discrete Event of Model of LAP/MAP (LAP-D/CSMA-CD) Integrated LAN Model. This model is a combined form of CSMA/CD implemented in [1] and LAP-D in chapter 5 and Appendix 1. The developed model is highly modular in which each protocol function of MAP and LAP is implemented independently which gives us the flexibility to modify any part of the protocol. The Discrete Event part of the integrated LAN model consists of nineteen events, nine subroutines, and two functions. In addition, there are five other routines defined as follows:

1. PARAMS.DAT file which declares the LAP, MAP and SLAM variables.

2. INITLAP and INITMAP which initialize the LAP and MAP variables respectively.

These routines are called by SLAM INTLC routine at the beginning of each simulation run.

3. OTPUTLAP and OTPUTMAP which prints the user collected statistics for LAP and MAP respectively. These routines are called by SLAM at the end of each simulation run time.

Tables A2.1 and A2.2 show a relationship between the events and user subroutines in the integrated LAP/MAP simulation model.

When an I-frame is generated is sent to the data link layer for sequencing, piggybacking of acknowledgement, and transmission. In the LAP alone model, the data link layer schedules the transmission event(LAPTRNS) with regard that channel is idle and not used by any other nodes. In case of LAP/MAP model, the data link layer is divided in two distinct sublayers:Logical Link Control(LAP-D) and Media Access Control(CSMA/CD) in which the actual transmission of data in done by MAC. In this the media is shared with other nodes in the network, therefore a transmitting node must sense the media idle before it can access the channel and transmit. This process is performed by the Media Access Protocol.

			schdl *
17	trnsmt	sense	schdl *
		timeout	copy *
		applclyr	
18	laprcv	rcvifrm	schdl *
		rcvsfrm	
19	applclyr	-	free *

* => SLAM Subroutine

Table A2.2 User Subroutines in Discrete Event LAP/MAP Model

Media Access Protocol Subroutines

Subroutine Name	Schedules To	Call To
colisn	etrans	search
		calc
		schdl *
search	(none)	copy *
		rmove *
calc	sense	schdl *
exdefr	sense	rmove *
		schdl *

Link Access Protocol Subroutines

laptrns	trnsmt	colct *
	sndsfrm	schdl *
	applclyr	
sndrcvack	sndsfrm	schdl *
		srch
		rmvackfrm
rmvackfrm	applclyr	rmove *
		colct *
		copy *
srch	-	rmove *
		copy *
alloctrsc	-	seize *

* => SLAM Subroutine

To develop an Integrated LAN model, certain changes had to be made in both protocols. The vital changes are concerned with the communication between LAP and MAP, distinction between background node and transport node, and the enforcement of no S-frames are discarded by the MAC layer which follows the error free transmission assumption for S-frames.

We now proceed to describe the changes that have been in CSMA/CD and LAPD simulation models in order to develop an integrated LAN model. Since[1] fully described the CSMA/CD simulation software, this report will not furnish any description of the events/subroutines that remained unchanged. The reader is encouraged to refer to [1] to further details. The two routines that are modified to provide the integration between LAP and MAP are CALC subroutine and NODRST even. In addition, COLISN subroutine considers processing time for the collision and deferring. Also in SENSE event some statistics(e.g.,MAP queuing delay)are modified to satisfy the integrated model. The last two are self explanatory in the code and there is not need for further details.

CALC subroutine performs two major functions after a collision has occurred, first it checks to see if too many collisions have occurred for this I-frame. If this is true then the packet is terminated and we return from this routine. In the integrated LAN model the above statement is true only for I-frames. When I-frames reach the maximum collision limit will be discarded, but if S-frame reaches the maximum collision limit will not be discarded by the MAC

layer because of the assumption made for S-frames and therefore the S-frame transmission is enforced by resetting its attempt limit to one. This phenomenon would impose the least backoff delay on the S-frame. The second function performed by CALC subroutine is the implementation of the backoff algorithm described in [4.5].

The next routine of CSMA/CD modified is NODRST event. This routine simulates the logic that interprets the meaning of trailing edge of packets that reach a given node. In this evenat, either a packet transmission is terminted, a packet is released from the defer file, or the node simply takes note of the end of the packet. When a packet is successfully transmitted, this event calls LAPTRNS subroutine which in a way MAC requests more frames for transmission from LAP.

The end of transmission of a frame is stored in JTERMN variable. When a trialing edge of packet reaches a node. The node checks if this packet is at destination , if so then check further if this frame is a terminating packet(check JTERMN) and not just a jam signal. If these conditions are met then the destination node accepts the frame. Next this node checks if this I-frame(if it is I-frame and not a S-frame)should be discarded because of transmitting node's LAP REJ or Timeout recovery condition. If this condition is false then handover this I-frame to LAP with appropriate processing times for further processing. In addition to these changes in MAP, also FRERSC subroutine is replaced by LAPTRNS subroutine which is basically a gateway routine between LAP and MAP

of transmitting node. The user variables and SLAM[2] Global and Attribute variables remained unchanged as it given in [1] for the MAP. See chapter 6 for statistics and SLAM definition variables.

We now describe the changes in LAP for the implementation of LAP/MAP simulation model. First change the event numbers that are given in table A1.1 to the event numbers in table A2.1. Table A2.3 shows the event relationship of LAP model and LAP/MAP model.

Table A2.3 The Event relationship between LAP and LAP/MAP Models

LAP		LAP/MAP	
Event	Event Name	Event	Event Name
1	SNDIFRM	11	SNDIFRM
2	TRNSMT	17	TRNSMT
3	LAPTRNS	Subroutine	LAPTRNS
4	LAPRCV	18	LAPRCV
5	RCVIFRM	12	RCVIFRM
6	SNDSFRM	13	SNDSFRM
7	RCVSFRM	14	RCVSFRM
8	TIMEOUT	15	TIMEOUT
9	REXMIT	16	REXMIT
10	APPLCLYR	19	APPLCLYR

The second change is the SLAM Attribute Values. AS it described in chapter 6 MAP uses attributes 2 to 24 and LAP uses attributes 25 to 42 for the integrated LAP/MAP model. The LAP alone model uses attributes 2 to 20, therefore the necessary change must be made to implement the integrated LAP/MAP model. These changes are given in table A2.4.

Table A2.4 SLAM Attribute Array Comparisons of LAP and LAP/MAP Model

LAP	LAP/MAP
atrib(2)	atrib(2)
atrib(3)	atrib(25)
atrib(4)	atrib(26)
atrib(5)	atrib(27)
atrib(6)	atrib(28)
atrib(7)	atrib(29)
atrib(8)	atrib(30)
atrib(9)	Defined in MAC
atrib(10)	atrib(32)
atrib(11)	Defined in MAC
atrib(12)	atrib(34)
atrib(13)	--
atrib(14)	atrib(31)
atrib(15)	atrib(33)
atrib(16)	atrib(38)
atrib(17)	atrib(39)
atrib(18)	atrib(40)
atrib(19)	atrib(41)
atrib(20)	atrib(42)

The description of LAP and LAP/MAP attributes and other variables are defined in chapters 5 and 6 respectively.

The logical steps described in LAP of the LAP/MAP model is the same as the LAP alone model described in Appendix 1. The major difference is in terms of event numbers scheduled and attribute array numbers which are given in tables A2.3 and A2.4. In the LAP alone model two events are slightly modified in order to implement the integrated LAP/MAP model. First LAPTRNS event is changed to a subroutine which is called from NODRST event of MAP. As we mentioned the LAPTRNS is the replacement of original FRERSC subroutine implemented in [1], therefore the first part of this subroutine contains the FRERSC logical steps and the second part is identical to the LAPTRNS event described in Appendix 1.

The next event that was modified is LAPRCV event. In addition to description of this event given in Appendix 1, we need to check if the received I-frame is generated by a background station or transport station. This logic is implemented using the ATRIB(36) which is set when the packet was generated.(see section 6.1.2.1). The I-frame is accepted if and only if the received frame is generated by a transport station.

Also in the integrated model in TRNSMT and SNDSFRM events the transmission event(LAPTRNS) is replaced by the first event of MAC which is the SENSE event. It should be noted that in LAP alone model since the media is not shared, therefore one event took care of the transmission event. On the other hand in the integrated

model the media is shared, then the implementation of Media Access Protocol becomes necessary in order to access the channel.

The last change was made in the LAP model in the way that the status of the source node is checked. In the integrated model, in addition to LAP status , we check the status of MAC which done using the variable NODEST (node state).

This concludes our discussion on the routines that make up the event file. The remaining routines that we did not discuss in this Appendix are unchanged, therefore the description of them are omitted. If a routine is concerned with LAP then it is fully described in Appendix 1, and also if a routine is concerned with MAP then refer to [1] for description of it.

The routines that form the Discrete Event portion of the TISL LAP/MAP Integrated LAN model is given at the end of this appendix.

References:

1. B.Wood,' A Performance Evaluation of Local Area Networks',Technical Report,TISL - 6731-3,January 1986

2. A.B.Pritsker,' Introduction to Simulation and SLAM II ',Halsted Press Inc., 1984

```
C ------------------------------------------------------------------
C        TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C FILENAME: Params.dat
C ------------------------------------------------------------------
C PURPOSE :
C
C         This is a section of code that is inserted at the beginning
C         each of the FORTRAN subroutines that are part of the SLAM
C         simulation through the use of the statement 'include params.dat'.
C         The purpose of this section if code is as follows:
C         +  Define all user parameters.
C         +  Define all user and SLAM variables
C         +  Establish common blocks that contain the user/SLAM varaibles
C
C
C ------------------------------------------------------------------------
c   this is a 1 mbit csma/cd conf.
c
c define user parameters:
c
        implicit none
        integer maxsta,mxcoll,idefer,itrans,icolpr,ijamng,iterm
     1          ,iidle,intrmv,iseed,mxboff
c
        real    waitim,rjmtim,slttim,maxsz,minsz,mxlkdy,lkdlyf
c
        parameter ( idefer = 0 )
        parameter ( itrans = 1 )
        parameter ( icolpr = 2 )
        parameter ( ijamng = 3 )
        parameter ( iterm  = 4 )
        parameter ( iidle  = 5 )
        parameter ( intrmv = 6 )
c
c define slam random number stream for backoff selection
        parameter ( iseed = 7 )
c
c
c define slam variables:
        integer ii,mfa,mstop,nclnr,ncrdr,nprnt,nnrun,nnset,ntape
        real    atrib,dd,ddl,dtnow,ss,ssl,tnext
     1          ,tnow,xx
c
      common/scom1/ atrib(100),dd(100),ddl(100),dtnow,ii,mfa,mstop,nclnr
     1          ,ncrdr,nprnt,nnrun,nnset,ntape,ss(100),ssl(100)
     2          ,tnext,tnow,xx(200)
c
c define user variables:
c
c map user variables:
      integer ntstus,nodest,icnt,icoll,frstat,excoll,atemps
     1        ,bitsgd,bitsbd,ncstat,lkstat,nbkoff
c
      real stadly,timegd,tgood,lkdly,mxdly,sumbit,smboff,xspak,xprob
     1     ,rate,eedly,xohbit,jtermn
c
c lapd user variables:
      integer ndsts,irejsts,itmoutcnt,irxmtcnt,icount,iotsqcnt
     2,idmgpktcnt,iwndsz,sfrm,pktcnt,rcvsvr,isnd,irsc,ists,irxsts
```

```
     3, totpktcnt, ipkterr, ichksts, jwndrt, itmot, iakvar, ktmp, jsnd
     4, krcv, ifrm, jfrm, infobts, ircvcnt, jchksts, rrfrmcnt, sfrmcnt
     5, bfrcnt, ibadpak, iabrtpkt, ipktrej, ilstpkt, rrsts
c
      real trctm, biterr, blkerr
        common/iucom/ ntstus(25), nodest(25), icnt(25)
     1      , icoll(25), stadly(25), excoll(25)
     2      , frstat(25), timegd(25), tgood, atemps(100)
     3      , bitsgd, bitsbd, ncstat(25), lkstat(25)
     4      , lkdly(25), sumbit, mxdly(25), nbkoff, smboff
     5      , lkdlyf, xspak, xprob, rate, mxcoll, mxboff, waitim
     6      , rjmtim, slttim, eedly, maxsz, minsz, xohbit
     7      , maxsta, ndsts(25), irejsts(25), itmoutcnt(25), idmgpktcnt(25)
     8      , irxmtcnt(25), icount(25), iotsqcnt(25), iwndsz, sfrm(25)
     9      , pktcnt(25), rcvsvr(25), isnd(25), irsc(25), ists(25), irxsts(25)
     +      , totpktcnt(25), ipkterr(25), ichksts(25), jwndrt(25), jsnd(25)
     +      , itmot(25), iakvar(25), ktmp(25), jtermn(25), krcv(25)
     +      , ifrm(25), jfrm(25), ircvcnt(25), jchksts(25), infobts(25)
     +      , bfrcnt(25), rrfrmcnt(25), sfrmcnt(25), rrsts(25)
     +      , trctm, biterr, blkerr
     +      , ilstpkt(25), ibadpak(25), iabrtpkt(25), ipktrej(25)
c ------------------------------------------------------------------------
c
c                    End of PARAMS.DAT
c
c ------------------------------------------------------------------------
```

```
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: intlc       AUTHOR: M. Kashefi          DATE: 7-15-86
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------
C PURPOSE :     Initialize SLAM variables and user defined variables in
C               TISL LAP-D/CSMA-CD simulation.
C ------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C ------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------
C                 ! initlap - initialize LAPD variables
C                 ! initmap - initialize CSMA/CD variables
C ------------------------------------------------------------------
      subroutine intlc
C
c The following routine will initialize all the variables in the map model
      call initmap
c The following routine will initialize all the variables in the lapd model
      call initlap
      return
      end
```

```
C<FF>
C ------------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------------
C PROGRAM NAME:  initmap.for       AUTHOR:  b.j.wood          DATE:  8-15-85
C ------------------------------------------------------------------------
C VERSION NUMBER :  1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------------
C PURPOSE :     Initialize SLAM variables and user defined variables in
C               TISL CSMA/CD simulation.
C ------------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C ------------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C params.dat      ! file  !      !        ! Declares variables and
C                 !       !      !        ! establishes common block.
C ------------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------------
C                 ! (none)
C ------------------------------------------------------------------------
      subroutine initmap
c
      include 'params.dat'
      real busdly,totdly(25,25),tmpmax
      integer i,j,k,itemp
c
c     Convert paremeters that were defined in the SLAM driver file as
c     xx(.) variables to FORTRAN parameters.
c
c     xx(26) = maxsta = number of active nodes
      itemp = ifix(xx(26))
      maxsta = itemp
c
c     xx(75) = rate = line rate in Mbps
      rate = xx(75)
c
c     xx(76) = mxcoll = max number of collisions
      itemp = ifix(xx(76))
      mxcoll = itemp
c
c     xx(77) = mxboff = backoff limit
      itemp = ifix(xx(77))
      mxboff = itemp
c
c     xx(78) = waitim = interframe spacing
      waitim = xx(78)
c
c     xx(79) = rjmtim = jam time
      rjmtim = xx(79)
c
c     xx(80) = slttim = slot time
      slttim = xx(80)
c
c     xx(81) = maxsz = max packet size with overhead
```

```
      maxsz = xx(81)
c
c     xx(82) = minsz = min packet size with overhead
      minsz = xx(82)
c
c     xx(83) = xohbit = number of overhead bits
      xohbit = xx(83)
c
c     xx(84) = eedly = end to end bus delay
      eedly = xx(84)
c
c     xx(85) = lkdlyf = one way node to bus delay ( all nodes same dist. )
      lkdlyf = xx(85)
c
c     xx(86) = xspak = small packet size used to model host echos
      xspak = xx(86)
c
c     xx(87) = xprob = probability that a xspak will be trans from host
      xprob = xx(87)
c
c     xx(131) = sensng = sensing processing time in microseconds
      sensng = xx(131)
c
c     xx(132) = collsn = collision processsing time in microseconds
      collsn = XX(132)
c
c     xx(133) = deffrng = deffering processing time in microseconds
      deffrng = xx(133)
c
c     xx(134) = crcchk = CRC CHECK processing time in microseconds
      crcchk = xx(134)
c
c     Initialize some attributes to zero.
c
      do 10 i = 3,17
            atrib(i) = 0.0
   10 continue
c
c     Initialize the node status arrays.
c
      do 20  i=1,maxsta
             nodest(i) = 5
             ncstat(i) = 0
             jtermn(i) = 0
c
   20 continue
c
c     Initialize the station delay array.  If the user does not wish that
c     the bus ports be equidistiant, then the following code must be
c     replaced with explicit definitions for each of the array elements.
c     Note, eedly is the end-to-end bus delay in microseconds.
c
      do 30 i=1,maxsta-1
            stadly(i)=eedly/float(maxsta-1)
   30 continue
c
c     Initialize the link delay array.  If the user does not wish each of
c     the links to be equidistant from the bus, then the following code
c     must be replaced with explicit dfinitions for each of the array
c     elements.  Note, lkdlyf is the average distance from each node to
```

```
c       the bus ports.
c
        do 40 i=1,maxsta
                lkdly(i)= lkdlyf
   40   continue
c
c       The following lines of code are used to determine the maximum
c       delay encountered by each node in the system.  In other words,
c       how long does it take before each node in the system sees a
c       packet from a particular node.  These values are stored in the
c       mxdly array.
c
c       Calculate delays between each pair of nodes in the network.
c       These delays are stored in the totdly(i,j) array.
c
        do 100 i = 1,maxsta
                do 200 j = 1,maxsta
                        busdly = 0.0
c
                        if (i.lt.j) then
                                do 300 k=i,(j-1)
 300                                    busdly=busdly+stadly(k)
                                totdly(i,j)=busdly + lkdly(i) + lkdly(j)
                        end if
c
                        if (i.gt.j) then
                                do 400 k=j,(i-1)
 400                                    busdly=busdly+stadly(k)
                                totdly(i,j)=busdly + lkdly(i) + lkdly(j)
                        end if
c
                        if(i.eq.j) then
                                totdly(i,j)=0.0
                        endif
c
 200                continue
 100    continue
c
c       Find the maximum delay for each node from the totdly(i,j) array.
c       Note, mxdly is the maximum round-trip delay.
c
        do 500 i=1,maxsta
                do 600 j=1,maxsta
                        tmpmax=max(tmpmax,totdly(i,j))
 600            continue
                mxdly(i)=2.0*tmpmax
                tmpmax=0.0
 500    continue
c
c       The intlc.for routine is executed each time the simulation is.
c       run.  If we wish to make multiple runs so that we can collect
c       batch means, some statistics should not be reintiialized in
c       subsiquent simulation runs.  Therefore, the following lines
c       of code will be executed only on the first in any series of
c       simulation runs.  Note, nnrun indicates the number of times
c       this simulation has been run.
c
c       if( nnrun .eq. 1 ) then
c               If TRUE, this is the first time that this routine
c               has been executed.   So, we will proceed to
```

```
c                initialize some statistic collection variables.
c
                 do 111  i=1,maxsta
                         icnt(i) = 0
                         excoll(i)=0
                         frstat(i)=0
                         icoll(i) = 0
                         timegd(i) = 0.0
111              continue
c
                 do 50  i=1,mxcoll
                         atemps(i)=0
50               continue
c
                 bitsgd=0
                 bitsbd=0
c        endif
c
         return
         end
```

```
C<FF>
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: initlap.for       AUTHOR: M.Kashefi        DATE: 12-15-85
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------
C PURPOSE :     Initialize SLAM variables and user defined variables in
C               TISL LAP-D simulation.
C ------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !    DESCRIPTION
C ------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat      ! file  !      !        ! Declares variables and
C                 !       !      !        ! establishes common block.
C ------------------------------------------------------------------
C                    SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------
C                 ! (none)
C ------------------------------------------------------------------
      SUBROUTINE INITLAP
C
      INCLUDE 'PARAMS.DAT'
C
      INTEGER I
C
      DO 5 I = 25,42
         ATRIB(I) = 0
    5 CONTINUE
C
C     XX(110) = IWNDSZ = MAXIMUM NUMBER OF OUTSTANDING FRAMES(WINDOW SIZE)
      IWNDSZ = IFIX(XX(110))
C
C     XX(113) = BITERR = PROBABILITY THAT A BIT BEING IN ERROR
      BITERR = XX(113)
C
C     XX(114) = TRCTM = VARIABLE USED FOR TRACING AND DEBUGGING PURPOSES
C     MEASURED IN MICRO SEC.
      TRCTM = XX(114)
C
C     XX(117) = ACKPKT = ACKPKT PROCESSING TIMES IN MICROSECONDS
      ACKPKT = XX(117)
C
C     XX(118) = BMOVE = BLOCK MOVE PROCESSING TIMES IN MICROSECONDS
      BMOVE = XX(118)
C
C     XX(119) = ENQUEU = ENQUEU PROCESSING TIMES IN MICROSECONDS
      ENQUEU = XX(119)
C
C     XX(120) = DEQUEU = DEQUEU PROCESSING TIMES IN MICROSECONDS
      DEQUEU = XX(120)
C
C     XX(121) = RCVCOM FROM APPLICATION LAYER PROCESSING TIMES
C               IN MICROSECONDS
```

```
      RCVCOM = XX(121)
C
C     XX(122) = SNDCOM TO APPLICATION LAYER PROCESSING TIMES
C               IN MICROSECONDS
      SNDCOM = XX(122)
C
C     XX(123) = INFPKT = INFPKT PROCESSING TIMES IN MICROSECONDS
      INFPKT = XX(123)
C
C     XX(124) = RCVMSG = RCVMSG PROCESSING TIMES IN MICROSECONDS
      RCVMSG = XX(124)
C
C     XX(125) = RECEIVE = RECEIVE PROCESSING TIMES IN MICROSECONDS
      RECEIVE = XX(125)
C
C     XX(126) = SEND = SEND PROCESSING TIMES IN MICROSECONDS
      SEND = XX(126)
C
      XX(127) = SNDSPR = SEND SUPREVISORY FRAME PROCESSING TIMES
C               IN MICROSECONDS
      SNDSPR = XX(127)
C
C     XX(128) = SNDMSG = SNDMSG PROCESSING TIMES IN MICROSECONDS
      SNDMSG = XX(128)
C
      DO 10 I = 1,MAXSTA
C
         NDSTS(I)= 0
         IREJSTS(I) =0
         ITMOUTCNT(I)=0
         IRXMTCNT(I)=0
         ICOUNT(I)=0
         IOTSQCNT(I)=0
         IDMGPKTCNT(I)=0
         PKTCNT(I) = 0
         RCVSVR(I) = 1
C
   10 CONTINUE
C
      DO 20 I = 1,MAXSTA
C
         TOTPKTCNT(I) = 0
         ITMOT(I) = 0
         ISND(I) = 0
         IRXSTS(I)=0
         IPKTERR(I) = 0
         ICHKSTS(I) = 0
         IBADPAK(I) = 0
         JSND(I) = 0
         IAKVAR(I) = 0
         SFRM(I) = 0
         IABRTPKT(I)=0
C
   20 CONTINUE
C
      DO 30 I = 1,MAXSTA
C
         KRCV(I) = 0
         IFRM(I) = 0
         JCHKSTS(I) = 0
```

```
          IRCVCNT(I) = 1
          INFOBTS(I) = 0
          JFRM(I) = 0
          RRFRMCNT(I) = 0
          SFRMCNT(I) = 0
          BFRCNT(I) = 0
          RRSTS(I) = 0
          IPKTREJ(I) = 0
C
   30   CONTINUE
C
        RETURN
        END
```

```
C ------------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------------
C PROGRAM NAME:  event.for      AUTHOR:
C                               CSMA/CD--->B. J. Wood          Date : 8-15-85
C                                LAP-D --->M. S. Kashefipour   Date : 7-15-86
C ------------------------------------------------------------------------
C VERSION NUMBER : 2.0          RELEASE DATE : 7-15-86
C ------------------------------------------------------------------------
C PURPOSE :     This series of subroutines form the discrete event
C               portion of the TISL INTEGRATED LAPD/CSMA-CD simulation.
C ------------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------------
C    NAME                    DESCRIPTION
C ------------------------------------------------------------------------
C event             ! Interfaces the SLAM scheduling routine with the events
C                   ! that are to be scheduled
C
C             CSMA/CD EVENTS AND SUBROUTINES
C
C sense             ! Senses condition of channel for packets that attempt
C                   ! to access the network
C txmit             ! Initiates propagation of the leading edge of packet
C lprop             ! Propagates the leading edge of a packet to the left.
C rprop             ! Propagates the leading edge of a packet to the right.
C succes            ! Logic associated with successful packet transmission.
C etrans            ! Initiates propagation of the trailing edge of packet.
C lfprop            ! Propagates the trailing edge of a packet to the left.
C rfprop            ! Propagates the trailing edge of a packet to the right.
C nodset            ! Perfroms logic routines on leading packet edges.
C nodrst            ! Performs logic routines on trailing packet edges.
C colisn            ! Performs collision resolution logic.
C search            ! Removes succes routines from event calendar in colisn.
C calc              ! Implements backoff calculation algorithm.
C exdefr            ! Removes packet from defer file for subsequent
C                   ! retransmission.
C
C             LAP-D EVENTS AND SUBROUTINES
C sndifrm           ! receives a packet from higher layers, makes a copy
C                   ! of it and then pass the packet to the transmitter
C trnsmt            ! start timer start transmitting
C laprcv            ! check for correct CRC, and handover the received frame
C                   ! to the data link layer
C rcvifrm           ! destination receives an error free I-frame, make sure
C                   ! packet is in sequence
C sndsfrm           ! sends supervisory S-frames ( RR, REJ ,and RR for
C                   !  timeout recovery condtion)
C rcvsfrm           ! destination receives a supervisory S-frame, process
C                   ! this frame based on its type
C timeout           ! timer expires for the packet which acknowledgement
C                   ! was not received, follow the procedure for the timer
C                   ! receovery condition
C rexmit            ! begin retransmission of requested packet and any
C                   ! other packets outstanding
C applclyr          ! sends and receives I-frames to and from data link layer
C alloctrsc         ! perform logic routines on resource allocation
C laptrns           ! performs the logic for data link layer after
C                   ! MAC layer has sucessfully transmitted a frame
C sndrcvack         ! takes necessary action for sending and receiving
```

```
C                   ! acknowledgements
C srch              ! remove timeout event from the event calendar (stop timer)
C rmvackfrm         ! packet is acknowledged, remove this packet from
C                   ! the retransmission buffer
C ------------------------------------------------------------------------
C AMENDMENTS:
C ------------------------------------------------------------------------
C     DATE          DESCRIPTION
C ------------------------------------------------------------------------
C                   ! (none)
C ------------------------------------------------------------------------
C
C NOTE:  These routines must be linked to the SLAM FORTRAN library!!!
C
C ------------------------------------------------------------------------
C
```

```
C ------------------------------------------------------------------------
C          TISL CSMA-CD/LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------------
C SUBROUTINE NAME:  event
C
C PURPOSE :       This routine interfaces the event scheduling routine
C                 with the events that must be scheduled.  This routine
C                 allows events to be called by their number as opposed
C                 to calling a routine by its name.  This subroutine is
C                 required by all SLAM discrete event models.
C ------------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------------
C NAME                TYPE     CLASS  RANGE
C ------------------------------------------------------------------------
C ievent           |integer| read | 1-10    |Number of the event that is
C                  |       |      |         |is being called.
C ------------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C                  |       |      |         |(none)
C ------------------------------------------------------------------------
C EVENTS SCHEDULED:                    ROUTINES SCHEDULING THIS SUBROUTINE:
C ------------------------------------------------------------------------
C (none)                               SLAM driver file
C ------------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------------
C       CSMA/CD EVENTS
C
C sense            | Sense channel condition as packet attempts to access
C                  | the channel.
C txmit            | Begin propagation of leading edge of packet.
C lprop            | Propagate leading edge of packet to the left.
C rprop            | Propagate leading edge of packet to the right.
C succes           | Logic associated with successful packet transmission.
C etrans           | Begin propagation of trailing edge of packet.
C lfprop           | Propagate trailing edge of packet to the left.
C rfprop           | Propagate trailing edge of packet to the right.
C nodset           | Interprets meaning of leading packet edges.
C nodrst           | Interprets meaning of trailing packet edges.
C
C       LAP-D EVENTS
C
C sndifrm          | receives an I-frame from application layer
C                  | store a copy this frame into the retransmission
C                  | buffer and send it to the transmitter
C trnsmt           | when the transmitter is idle will transmit frames,
C                  | start the timer for each outgoing I-frame, consider
C                  | the logic for piggybacking of acknowledgemnts, and
C                  | retransmission of requested I-frames
C laprcv           | performs the logic for an error free communication
C                  | between a source and a destination, also based
C                  | on the frame type would invoke the appropriate
C                  | part of the data link layer
C rcvifrm          | destination receives an error free I-frame, make sure
C                  | this frame is in sequence, if not follow the procedures
C                  | for out of sequence packets
C sndsfrm          | receiver rejects an out of sequence frame, send a request
```

```
                    ! for retransmission for this packet
rcvsfrm             ! source which sent the out of sequnec packet, receives
                    ! a request for retransmission
timeout             ! timer exired, follow the procedure for timer recovery
                    ! sens a command S-frame to the receiving node
rexmit              ! find the requested I-frame in the retransmission buffer
                    ! and invoke the transmitter to  begin retransmission
                    ! of requested I-frame(s)
applclyr            ! implements the user and is to receive data packets
                    ! form the data link layer or to send data packets
                    ! to the data link layer
                    !
--------------------------------------------------------------------------


      subroutine event(ievent)
      integer ievent
      go to (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19),ievent

CSMA/CD events

  1 call sense
      return
  2 call txmit
      return
  3 call lprop
      return
  4 call rprop
      return
  5 call succes
      return
  6 call etrans
      return
  7 call lfprop
      return
  8 call rfprop
      return
  9 call nodset
      return
 10 call nodrst

c
c  LAP-D events
c
 11   call sndifrm
      return
 12   call rcvifrm
      return
 13   call sndsfrm
      return
 14   call rcvsfrm
      return
 15   call timeout
      return
 16   call rexmit
      return
 17   call trnsmt
      return
 18   call laprcv
      return
 19   call applclyr
```

```
c
      return
      end
C
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: sense                          EVENT NUMBER: 1
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs two functions.  First, it collects
C                statistics on packets are they are first presented
C                to the network.  Secondly, this routine tests the node/
C                channel array to determine the node's perception of the
C                channel.  If the channel appears idle, a transmit event
C                is scheduled.  If the channel appears busy, the packet
C                is placed in the defer file where it will remain until
C                exdefr is executed at this node.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C subroutine calc                          transmit, event 2
C subroutine exdefr
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C colct           ! A SLAM routine that collects statistics on variables.
C filem           ! A SLAM routine that places items in files.
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
C
      subroutine sense
      include 'params.dat'
      integer inode,nretry
      real tqueud,delay,csmadly
c
        inode = ifix(atrib(2))
c
c             Increment the first access atrib.
c
      atrib(17)=atrib(17)+1
c
      if  ( atrib(13) .eq. 0 ) then
c
c                   If TRUE, the packet has just left the node queue,
c                   so collect statistics and set statistic collection
c                   attributes.
c
            atrib(35) = tnow
c           measure the time that this packets spends in the MAC
c           level. This includes all the packets(Ifrm & Sfrm)
c
            if(ifix(atrib(40)).eq.0.and.ndsts(inode).ne.2)then
c
               tqueud=tnow-atrib(1)
```

```
                call colct(tqueud,3)
c
              endif
c
              atrib(13)=1
              atrib(14)=tnow
        end if
c
c             Set inode to the node that generated the packet.
c
        inode = ifix(atrib(2))
c
c             Set a(10) to the amount of time left after the collision
c             discrimination period.
c
        atrib(10) = atrib(9) - mxdly(inode)
c
c             Check the node's perception of the channel status.
c
        if ( ncstat(inode) .eq. 0 ) then
c
c                     If TRUE, the channel is sensed idle, so
c                     set a(5) to indicate a transmission event.
c
              atrib(5) = 1
c
c                     Calculate the proper delay for the packet in sense.
c                     Here, delay is used to refer to the amount of time
c                     before the leading edge of the packet reaches the bus.
c
              nretry=ifix(atrib(6))
c
              if(nretry.ne.0.and.nodest(inode).ne.idefer) then
c
c                     If TRUE, this subroutine was called from COLISN or
c                     EXDEFR, so we do not need to impose the waittime on
c                     the packet delay.
c
                 delay = lkdly(inode)
c
              else
c
c                     Otherwide, the packet is making its frist attempt to
c                     transmit, so the waittime (interframe spacing) must
c                     be added to the delay.
c
                 delay = waitim + lkdly(inode)
c
              end if
c
c                     Set the node state to TRANSMITTING.
c
              nodest(inode) = itrans
c
c                     Set a(12) to the current time plus the delay before
c                     we begin transmitting from this node.
c
              atrib(12)=tnow + (delay - lkdly(inode))
c
c                     Schedule TXMIT after waiting the delay.
```

```
c
                call schdl(2,delay,atrib)
c
        else
c               The channel is sensed busy; so, set a(5) to a txmit event.
c
                atrib(5) = 1
c
c                       Set the node state to defering.
c
                nodest(inode) = idefer
c
c                       File the packet in the defer file.  The packet will
c                       return to sense when the node state returns to idle
c                       via EXDEFR.
c
                call filem( (nclnr-1) ,atrib)
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------------
C EVENT NAME:  txmit                          EVENT NUMBER:  2
C ------------------------------------------------------------------------
C
C PURPOSE :      This routine simulated the propagation of the leading
C                edge of a packet starting from the origin node port.
C                Here, the packet is echoed back to the origin node,
C                and the packet is propagated to the right and left of
C                the origin node port.
C
C ------------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------------
C event 1, sense                      event 9, nodset
C                                     event 3, lprop
C                                     event 4, rprop
C ------------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------------
C schdl           ! SLAM routine to schedule events.
C ------------------------------------------------------------------------
C
      subroutine txmit
      include 'params.dat'
      integer inode
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c             Set destination node = generation node.  This is used by NODSET.
c
      atrib(15) = atrib(2)
c
c             Signal that we are now beginning a propagation event.
c
c
      atrib(5) = 0
c
c             Echo the leading edge of the packet back to the origin node by
c             scheduling the NODSET routine, event 9.
c
      call schdl(9,lkdly(inode),atrib)
c
c             Begin propagation to the adjacent ports on the bus.
c
      if ( inode .ne. 1 ) then
c                   If TRUE, we are NOT at the beginning of the line and
c                   should schedule propagation 1 port to the left, so
c                   move the left propagtion marker one port to the left.
```

```
c
                atrib(3) = inode - 1
c
c                       Schedule the packet to arrive at the next port
c                       to the left in the amount of time specified in
c                       the station delay array.
c
                call schdl(3,stadly(inode-1),atrib)
        else
c                       We are at the beginning of the bus, so set the left
c                       propagation marker to indicate that the packet has
c                       propagated all the way to the left.
c
                atrib(3) = 0
c
        end if
c
        if ( inode .ne. maxsta ) then
c                       If TRUE, we are not at the end of the line and should
c                       schedule a propagation 1 port to the right. So, set
c                       the right propagation marker one port to the right.
c
                atrib(4) = inode + 1
c
c                       Schedule the packet to arrive at the next port to
c                       the right in the amount of time specified in the
c                       station delay array.
c
                call schdl(4,stadly(inode),atrib)
        else
c               We are at the end of the line so, set the right propagtion
c               marker to indicate that the packet has finished propagating
c               right.
c
            atrib(4) = 0
c
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: lprop                          EVENT NUMBER: 3
C ------------------------------------------------------------------
C
C PURPOSE :      This routine simulates the propagation of the leading
C                edge of a packet along the bus to ports that are to the
C                left of the origin node.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !         !Declare variables and
C                 !       !      !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:      EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 2, transmit                    event 3, lprop
C event 3, lprop                       event 9, nodset
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events
C ------------------------------------------------------------------
C
      subroutine lprop
      include 'params.dat'
      integer poslft,inode
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c             Set poslft to the propagation left marker.
c
      poslft = ifix(atrib(3))
c
c             Set destination node = the port where we are now.
c
      atrib(15) = atrib(3)
c
c             Send leading edge of the packet from the bus to the node.
c
      call schdl(9,lkdly(poslft),atrib)
c
c             Set the propagate left marker to the next port to the left.
c
      atrib(3) = atrib(3) - 1
c
      if ( poslft .ne. 0 ) then
c             If TRUE, we have not finished propagating to the left and must
c             schedule the packet to arrive to the next port, so set a(5) to
c             indicate a propagation event.
c
              atrib(5) = 0
```

```
C
C                       Schedule the packet to arrive to the next port in the
C                       amount of time specified in the stadly array, and
C                       execute the lprop event again.
C
                call schdl(3,stadly(poslft),atrib)
        else
C               We have finished propagating to the left, so
C               set the left prop marker to indicated we are finished.
C
           atrib(3) = 0
C
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rprop                          EVENT NUMBER: 4
C ------------------------------------------------------------------
C
C PURPOSE :      This routine simulates propagation of the leading edge
C                of a packet along the bus to ports that are to the right
C                of the origin node.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 2, txmit                      event 4, rprop
C event 4, rprop                      event 9, nodset
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events.
C ------------------------------------------------------------------
C
      subroutine rprop
      include 'params.dat'
      integer posrgt,inode
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c             Set posrgt to the propagate left marker.
c
      posrgt = ifix(atrib(4))
c
c             Set destination node = the port where we are now.
c
      atrib(15) = atrib(4)
c
c             Send packet to destionation node.
c
      call schdl(9,lkdly(posrgt),atrib)
c
c             Set the propagation right marker to the next port to the right.
c
      atrib(4) = atrib(4) + 1
c
      if ( posrgt .ne. maxsta ) then
c                   If TRUE, we have not finished propagating to the right
c                   and must schedule the packet to arrive to the next port.
c                   So, set a(5) to indicate a propagation event.
c
            atrib(5) = 0
```

```
c
c                       Schedule the packet to arrive to the next port in the
c                       amount of time specified in the stadly array, and
c                       execute the RPROP event again.
c
                call schdl(4,stadly(posrgt),atrib)
        else
c               We have finished propagating to the right, so
c               set the right prop marker to indicated we are finished.
c
                atrib(4) = 0
c
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: succes                        EVENT NUMBER: 5
C ------------------------------------------------------------------
C
C PURPOSE :      This routine is scheduled after a packet has passed the
C                collision discrimination period without experiencing a
C                collision.  Here, the propagation of the end of a
C                successful packet is scheduled.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 9, nodset                     event 6, etrans
C                                     event 10, nodrst
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events.
C ------------------------------------------------------------------
C
      subroutine succes
      include 'params.dat'
      integer inode
      real paktim
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c             Set a(5) to indicate a transmission event.
c
      atrib(5) = 1
c
c             Set the node state to indicate that we are ready to
c             terminate the packet.
c
      nodest(inode) = iterm
c
c             Set paktim to be the time when the end-of-packet will
c             arrive at the source port.
c
      paktim = atrib(10) + lkdly(inode)
c
c             Schedule ETRANS to occur once the packet has finished
c             transmitting ( the amount left after the collision
c             discrimination period).
c
      call schdl(6,paktim,atrib)
c
```

```
c               Since the node is smart enough to know that it has finished
c               transmitting a packet before it sees the end of the packet,
c               we wish to schedule the end-of-packet logic event, NODRST,
c               so that it ocurrs immediately after the packet is finished
c               transmitting.  So, set a(16) to tell NODRST that it is being
c               scheduled before the end of packet is received.
c
      atrib(16)=1.0
c
c               Perform logic event after packet is finished transmitting.
c
      call schdl(10,atrib(10),atrib)
      return
      end
C
```

```
C ------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: etrans                        EVENT NUMBER: 6
C ------------------------------------------------------------------
C
C PURPOSE :      This routine simulates the propagation of the trailing
C                edge of a successful packet or jam signal, starting at
C                the origin node port.  Here, the trailing edge of the
C                packet is echoed back to the origin node, and then it is
C                propagated to ports to the right and left of the origin
C                node port.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 5, success                    event 10, nodrst
C subroutine colisn                   event 7, lfprop
C                                     event 8, rfprop
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events.
C ------------------------------------------------------------------
C
      subroutine etrans
      include 'params.dat'
      integer inode
c
c               Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c               Set destination node = the generating node.
c
      atrib(15) = atrib(2)
c
c               Echo end-of-packet to source node.
c
      call schdl(10,lkdly(inode),atrib)
c
c               Begin propagation of the end-of-packet.
c
      if ( inode .ne. 1 ) then
c                     We are not at the beginning of the line and should
c                     schedule the ending edge to propagate 1 port to the
c                     left.  Move the end of prop left marker one port to
c                     the left.
c
               atrib(7) = inode - 1
c
```

```
c                       Set a(5) to indicate a propagation event.
c
              atrib(5) = 0
c
c                       Schedule the ending edge of the packet to arrive at
c                       the next port to the left in the amount of time
c                       specified in the stadly array, and execute the
c                       lfprop event.
c
              call schdl(7,stadly(inode-1),atrib)
c
          else
c             We have finished propagating the ending edge to the left, so
c             set the end of prop left marker to indicate we have finished.
c
             atrib(7) = 0
c
          end if
c
          if ( inode .ne. maxsta ) then
c             If TRUE, we have not finished sending the ending edge of the
c             packet to the right of the origin port and must schedule
c             the ending edge to arrive to the next port to the right.
c
c                       Set the end prop right marker to the next port to the
c                       right.
c
              atrib(8) = inode + 1
c
c                       Set a(5) to indicate a propagation event.
c
              atrib(5) = 0
c
c                       Schedule the ending edge of the packet to arrive
c                       to the next port in the amount of time specified
c                       by the stadly array, and execute the rfprop event.
c
              call schdl(8,stadly(inode),atrib)
          else
c             We have finished the ending edge propagation to the right, so
c             set the end right prop marker to indicate we are finished.
c
             atrib(8) = 0
c
          end if
          return
          end
C
```

```
C ------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: lfprop                        EVENT NUMBER: 7
C ------------------------------------------------------------------
C
C PURPOSE :      This routine simulates the propagation of the trailing
C                edge of a packet to ports to the left of the origin node
C                port.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 6, etrans                     event 10, nodrst
C event 7, lfprop                     event 7, lfprop
C------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules routines.
C------------------------------------------------------------------
C
      subroutine lfprop
      include 'params.dat'
      integer poslft,inode
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c             Set poslft to the end prop left marker.
c
      poslft = ifix(atrib(7))
c
c             Set destination node.
c
      atrib(15) = atrib(7)
c
c             Echo end-of-packet to port node.
c
      call schdl(10,lkdly(poslft),atrib)
c
c             Decrement the end of prop left marker to point to the
c             next node to the left.
c
      atrib(7) = atrib(7) - 1
c
c             Set poslft to the next node to be propagated to.
c
      poslft = atrib(7)
c
      if ( poslft .ne. 0 ) then
```

```
c               we have not finished prop the ending edge of the packet
c               to the left and must schedule the ending edge to arrive
c               to the next node , so set a(5) to indicate propagation event.
c
                atrib(5) = 0
c
c                       Schedule the ending edge of the packet to arrive to
c                       the next node in the amount of time specified in the
c                       stadly array, and begin lfprop again.
c
                call schdl(7,stadly(poslft),atrib)
        else
c               we have finished propagating the ending edge of the packet to
c               the left set the end of prop to the left marker to indicate
c               we have finished.
c
           atrib(7) = 0
c
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------------
C EVENT NAME: rfprop                          EVENT NUMBER: 8
C ------------------------------------------------------------------------
C
C PURPOSE :      This routines simulates the propagation of the trailing
C                edge of a packet to ports to the right of the origin node
C                port.
C
C ------------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------------
C event 6, etrans                     event 10, nodrst
C event 8, rfprop                     event 8, rfprop
C------------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events.
C------------------------------------------------------------------------
C
      subroutine rfprop
      include 'params.dat'
      integer posrgt,inode
c
c             Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c             Set posrgt to the end prop right marker.
c
      posrgt = ifix(atrib(8))
c
c             Set destination node.
c
      atrib(15) = atrib(8)
c
c             Send end-of-packet to node.
c
      call schdl(10,lkdly(posrgt),atrib)
c
c             Increment the end prop right marker to point to the next node.
c
      atrib(8) = atrib(8) + 1
c
      if ( posrgt .ne. maxsta ) then
c             If TRUE, we have not finished propagating the ending edge of
c             the packet to the right and must schedule the ending edge to
c             arrive to the next node, so set a(5) to indicate a prop event.
c
              atrib(5) = 0
```

```
c
c                    Schedule the ending edge of the packet to arrive to
c                    the next node in the amount of time specified in the
c                    stadly array, and begin rfprop again.
c
            call schdl(8,stadly(posrgt),atrib)
        else
c           We have finished propagating the ending edge of the packet
c           to the right,so set the end of prop to the right marker to
c           indicate that we have finished.
c
           atrib(8) = 0
c
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: nodset                        EVENT NUMBER: 9
C ------------------------------------------------------------------
C
C PURPOSE :      This event simulates the logic that interprets the
C                leading edge of the leading edge of packets that arrive
C                at a given node.  Here, either the SUCCES event is
C                scheduled, a collision is detected, or the node simply
C                notes that the channel is busy.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 2, txmit                      event 5, succes
C event 4, rprop
C event 3, lprop
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C schdl           ! SLAM routine that schedules events
C colisn          ! User subroutine that handles collisions.
C ------------------------------------------------------------------
C
      subroutine nodset
      include 'params.dat'
      integer inode
      real colper,sav0
c
c             Set inode to be the number of this node.
c
      inode = ifix(atrib(15))
c
c
c             Increment the node-channel array to update the node's
c             perception of the channel condition.
c
      ncstat(inode) = ncstat(inode) + 1
c
      if(atrib(15).eq.atrib(2)) then
c             This condition is true if a node receives its own packet,
c             i.e., if this event was scheculed from transmit.  If TRUE,
c             we must make certain decisions about what has happened on
c             the channel.
c
              if ( ncstat(inode) .ge. 2 ) then
c                    A collision has occured during the interframe spacing,
c                    so set the node state to indicate that a collision has
c                    occured during the interframe spacing.
c
                 nodest(inode) = intrmv
```

```
c
c                    Call the collision subroutine so that backoff can
c                    be determined (amoung other things).
c
                  call colisn
c
               else if ( nodest(inode) .eq. itrans ) then
c                    The node is in the transmit state and we must schedule
c                    the end of the collision discrimination period, so
c                    set a(5) to indicate a transmission event.
c
                  atrib(5) = 1
c
c                    Set the node state to indicate that the node is
c                    in the collision discrimination period.
c
                  nodest(inode) = icolpr
c
c                    Calculate the new collision discrimination period.
c                    This takes into account that the event is scheduled
c                    after propagation to the node.
c
                  colper= mxdly(inode) - 2.0*lkdly(inode)
c
c                    Schedule success to be called after the slot
c                    time ( collision discrimination period ) has
c                    elapsed, since all the other nodes will allow
c                    inode to complete it's transmission now.
c
                  call schdl(5,colper,atrib)
c
               else
c                  If the node state was not set to transmit or ncstat was not
c                  greater than or equal to 2, then this event should not have
c                  been called.
c
                  write(nprnt,201)
  201             format('Error in node state from event(9)')
c
               end if
               return
c
      end if
c
c     The following routine simulates the logic used by a node during
c     a propagation event.  This is the case when the above condition
c     is not met (i.e., this event was not scheduled by txmit.
c
      if(ncstat(inode).eq.2.and.nodest(inode).eq.icolpr) then
c              a colisn has occured at the node port
c
c                    Save the origin node.
c
               sav0 = atrib(2)
c
c                    Set the node marker to the poslft node.
c
               atrib(2) = float(inode)
c
c                    Call the collision subroutine.
```

```
c
                call colisn
c
c                       Return to the origin node,
c
                atrib(2) = savO
c
        end if
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: nodrst                        EVENT NUMBER: 10
C ------------------------------------------------------------------
C
C PURPOSE :      This routine simulates the logic that interprets the
C                meaning of the trailing edge of packets that reach the
C                a given node.  In this event, either a packet is ter-
C                minated, a packet is released from the defer file, or
C                the node simply takes note of the end of the packet.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      | file  | read |         |Declare variables and
C                 |       |      |         |establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C event 5, succes                     (none)
C event 6, etrans
C event 7, lfprop
C event 8, rfprop
C ------------------------------------------------------------------
C SUBROUTINES CALLED BY THIS EVENT:
C ------------------------------------------------------------------
C laptrns          | Frees one unit of resource and collects statistics.
C exdefr           | Removes packets from the attribute file for retrans-
C                  | mission.
C ------------------------------------------------------------------
C
      subroutine nodrst
      include 'params.dat'
          integer inode,itmp
      real savO,prssdly
c
c               inode is the destination node.
c
      inode = ifix(atrib(15))
c
c
      if (atrib(16).eq.1.0) then
c                     If TRUE, this event was not propagated,
c                     so reset a(16).
c
              atrib(16)=0.0
c
c                     Since this event was not propagated, it would not
c                     be correct to decrement the node/channel status array.
c
      else
c                     The event was propagated, therefore we should decrement
c                     the node/channel status array.
c
              ncstat(inode) = ncstat(inode) - 1
c
```

```
c
        end if
c
        if(atrib(2).eq.atrib(15)) then
c
c       The above condition is true only if this routine was scheduled by
c       subroutine etrans.  If TRUE, then we perform the following logic
c       functions, else we proceed to a different set of logic functions.
c
                if ( nodest(inode) .eq. iterm ) then
c
                    itmp = ifix(atrib(2))
c                   save the transmitting node identity
c
                    frame termination indicator, this variable
c                   is set to 1 when a frame is sucessfully transmitted,
c                   this variable also distincts the jam signals from
c                   regular frame transmission
                    jtermn(itmp) = 1
c
                        The packet has finished transmission, so we
c                       call the laptrns subroutine so that statistics can
c                       be collected and the next packet can begin the process.
c
                        call laptrns
                        return
c
                else if ( nodest(inode) .eq. iidle ) then
c                       this routine was scheduled after the logic functions
c                       were previously performed in colisn and success.
c
                        continue
c
                else if ( nodest(inode) .eq. idefer ) then
c                       The node has a packet ready to begin the process.
                        continue
c
                else if ( nodest(inode) .eq. ijamng ) then
c                       Jam signal is over, so set the node state to idle.
                        nodest(inode) = iidle
                else
c                       IF node is not in the iidle, terminate, defer, or
c                       jamming state, then nodrst should not have been called.
c
                        write(nprnt,30)
   30                   format('Error in node state from event(10)')
                end if
c
      end if
c
      itmp = ifix(atrib(2))
c     save the transmitting node identity
c
      if(ifix(atrib(15)).eq.ifix(atrib(25)).and.jtermn
     1    (itmp).eq.1)then
c       check if the received frame is at destination,also
c       check fi the recieved frame is not a jam signal
c
        jtermn(itmp) = 0
c       reset the frame termination indicator
```

```
c
      if(ifix(atrib(40)).le.0)then
c        check if the recieved frame is an I-frame
c
         if(ifrm(itmp).eq.0)then
c           check if this I-frame is to be discarded because
c           of REJ or timeout recovery
c
            atrib(42) = 0.0
c           non timeout event
c
            prssdly = rcvmsg + receive + crcchk
c           processing time for receivin a frame
c
            call schdl(18,prssdly,atrib)
c           schdl LAPRCV
c
      else
c
c
         ifrm(itmp) = 0
c        reset the varaible that indicates that if an I-frame
c        is to be suspended or not due to REJ or timeout
c        recovery
      endif
      else
c
         prssdly = rcvmsg + receive + crcchk
c        processing time to receive a frame
c
         call schdl(18,prssdly,atrib)
c        schdl LAPRCV
c
      endif
c
   endif
c
c
     atrib(42) = 0.0
c
c    The following condition is the logic that would be executed when
c    the finish propagation routines schedule this subroutine.
c
     if(ncstat(inode).eq.0.and.(nodest(inode) .eq. idefer)) then
c                    this node has a packet ready to send and has sensed
c                    the channel idle.
c
c                    Save the origin node indicator.
c
           savO = atrib(2)
c
c                    Set the node marker to the poslft node.
c
           atrib(2) = float(inode)
c
c                    Remove the poslft node's packet from the defer file.
c
c
c
```

```
            call exdefr

c
c
c                   Return to the origin node.
c
            atrib(2) = savO
c
        end if
c
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME:  sndifrm                        EVENT NUMBER:  11
C ------------------------------------------------------------------
C
C PURPOSE :       This routine performs the following functions:
C                 Collect statistics for time that this packet
C                 was queued in the application layer. Next assign
C                 a sequence number to this packet and store a duplicate
C                 of it in the repeate queue in case of retransmission.
C                 Finally hand over the packet to the transmit part
C                 of the data link layer for piggybaking the acknowledgment
C                 and timer considerations.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE     CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file   ! read !         !Declare variables and
C                 !        !      !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:      EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C SLAM Network Model                   trnsmt, event 17
C                                      sndsfrm,  event 13
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C filem           ! A SLAM routine that places items in files.
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
C
      SUBROUTINE SNDIFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, ILOC, MMLE, NEXT, NNQ, ITMP
      REAL QUDLY, PRSSDLY
C
      INODE=IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE=IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      ILOC=IFIX(ATRIB(26))
C     DEFINE THE REPEATE QUEUE OF THE SOURCE NODE
C
      CALL COLCT(ATRIB(11),31)
C     COLLECT STAT.  ON AVE.  MESSAGE LENGTH
C
     ' PKTCNT(INODE) = PKTCNT(INODE) + 1
C     INCREMENT # OF FRAMES GENERATED FROM THIS NODE
C
      ATRIB(27) = PKTCNT(INODE)
C     SET THE PACKET COUNT NUMBER
C
      ATRIB(34) = TNOW
C     VAR.  TO RECORD THE LAP TIME DELAY
```

```
C
      QUDLY = TNOW - ATRIB(1)
C     COMPUTE THE QUEUEING DELAY FROM USER TO LINK LEVEL
C
      CALL COLCT(QUDLY,5)
C     COLLECT STATS. ON QUEUEING DELAY
C
      JSND(INODE)=JSND(INODE) + 1
C     INCREMENT THE SEND SEQ # FOR THIS NODE ( V(s) - PROTOCOL VERSION )
C
      IF(JSND(INODE).GT.IWNDSZ)THEN
C        CHECK FOR WINDOW ROTATION
C
         JSND(INODE) = 1
C        RESET THE SEND SEQ #
C
      ENDIF
C
      ATRIB(28) = JSND(INODE)
C     ASSIGNING THE SEQUENCE NUMBER OF THE NEXT PACKET TO BE
C     TRANSMITTED ( N(s) - PROTOCOL VERSION )
C
      CALL FILEM(ILOC,ATRIB)
C     SAVE THE PACKET IN THE REPEAT QUEUE IN CASE OF RETRANSMISSION
C
      IF(NDSTS(INODE).EQ.2)THEN
C        IF NODE IS IN RETRANSMIT STATUS INCREMENT THE NUMBER
C        OF OUTSTANDING FRAMES TO BE TRANSMITTED.
C        THIS CONDITION CAN ONLY OCCUR WHEN APPLICTION LAYER
C        DOESN'T KNOW THE CURRNET STATUS OF DATA LINK LAYER
C        SO TI WOULD SEND A FRAME TO DATA LINK LAYER WHICH
C        NORMALLY IT SHOULDN'T.
C
         BFRCNT(INODE) = BFRCNT(INODE) + 1
C        INCREMENT NUMBER OF OUTSTANDING FRAMES TO BE TRANSMITTED
C
         RETURN
C
      ENDIF
C
      IF(ICHKSTS(INODE).EQ.1)THEN
C        TRANSMITTER IS BUSY, STORE THE PACKET,WHEN TRANSMITTER
C        RETURNS TO IDLE WILL TRANSMIT THIS FRAME.
C
         JFRM(INODE) = JFRM(INODE) + 1
C        INCREMENT NUMBER OF PACKETS WAITING FOR TRANSMISSION
C
         RETURN
C
      ENDIF
C
      IF(SFRM(INODE).GE.1)THEN
C        CHECK IF A SUPERVISORY FRAME WATING FOR TRANSMISSION
C        IF SO TAKE THE PROPER STEPS TO TRANSMIT THE S-FRAME AND
C        STOP TRANSMISSION OF I -FRAMES
C
         ATRIB(42) = 0.0
C        SET A CONDITION IF THIS NODE TIMEOUT,THE SCHDL.
C        TRANS. OF I-FRAME WILL BE REMOVED FROM THE EVENT
C        CALENDER
```

```
C
          ATRIB(40) = 1.0
C         S-FRAME INDICATOR FOR THIS PACKET
C
          ICHKSTS(INODE) = 1
C         SET THE TRANSMITTER BUSY
C
          PRSSDLY = SNDSPR
C         PROCESSING TIME TO SEND A SUPERVISORY FRAME
C
          CALL SCHDL(13,PRSSDLY,ATRIB)
C         START THE TRANSMISSION OF S-FRAME AFTER THE PPROCESSING
C         TIME PERIOD
C
          RETURN
C
        ENDIF
C
        ICHKSTS(INODE) = 1
C       SET THE TRANSMITTER BUSY FOR THIS NODE
C
C
        CALL SCHDL(17,0.0,ATRIB)
C       INVOKE THE TRANSMIT PART OF THE DATA LINK LAYER FOR
C       PROPER FRAMING ,ACKNOWLEDGMENT PIGGYBACKING , AND
C       TIMER CONSIDERATIONS
C
        RETURN
        END
```

```
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rcvifrm                          EVENT NUMBER: 12
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                First, compute the probability of received frame
C                being in error then use a uniform random number
C                generator to determine if this frame is in error.
C                Second, check if this frame is in error if so discard
C                the frame . If the frame is error free, then check if
C                this frame is in sequence if so increment receive seq.
C                # . Third call subroutine SNDRCVACK to remove the
C                timeout event for the acknowledged I-frame from the
C                event calendar, and finally remove the acknowledged
C                I-frame from the retransmission buffer.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laprcv , event 18                     sndsfrm, event 3
C                                       subroutine sndrcvack
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE RCVIFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER  INODE, JNODE, JTMP, NNQ, I
C
      REAL RANNUM, UNFRM, SAVE, SAVE1, TMDLY, LPDLY, NMBTS, PRSSDLY
      DIMENSION SAVE1(42)
C
      INODE = IFIX(ATRIB(2))
C     SET THE SOURCE TO INODE
C
      JNODE = IFIX(ATRIB(25))
C     SET THE DESTINATION TO JNODE
C
      IF(IRCVCNT(JNODE).GT.IFIX(ATRIB(27)))RETURN
C        CHECK FOR DUPLICATE FRAMES
C
      IF(IFIX(ATRIB(28)).EQ.RCVSVR(JNODE)) THEN
C        PACKET CONTAINED NO ERROR , NOW MAKE SURE IF THIS PACKET HAS
C        THE CORRECT SEQUENCE NUMBER
C
         IF(IREJSTS(JNODE).GE.1)THEN
C           CHECK IF THE RECEIVING NODE HAS A REJECTED FRAME
C
            IF(IFIX(ATRIB(28)).EQ.IPKTREJ(JNODE))THEN
C              CHECK FOR THE SEQUENCE NUMBER OF THE FRAME WHICH CAUSED
```

```
C                THE REJ CONDITION
C
                 IREJSTS(JNODE) = 0
C                RESET THE REJECT VARIABLE
C
                 IPKTREJ(JNODE) = 0
C                RESET THE VARIABLE WHICH STORES THE SEQUENCE
C                # OF REJECTED FRAME
C
              ENDIF
C
           ENDIF
C
           NMBTS = ATRIB(11) - XOHBIT
C          CALCULATE # OF INFORMATION BITS
C
           INFOBTS(INODE) = INFOBTS(INODE) + IFIX(NMBTS)
C          ACCUMULATE TOATL # OF SUCCESSFUL BITS TRANSMITTED
C          BY THE TRANSMITTING NODE
C
           LPDLY = TNOW - ATRIB(34)
C          CALCULATE LAP DELAY
C
           CALL COLCT(LPDLY,6)
C          COLLECT STAT. FOR AVE LAP DELAY
C
           RCVSVR(JNODE) = RCVSVR(JNODE) + 1
C          CORRECT PACKET, INCREMENT THE RECEIVE SEQ # FOR THIS NODE
C
           KRCV(JNODE) = KRCV(JNODE) + 1
C          INCREMENT THE ACK. VARIABLE FOR THIS NODE
C
           IRCVCNT(JNODE) = IRCVCNT(JNODE) + 1
C          INCREMENT THE # OF CORRECT PACKETS RECEIVED
C          BY THIS NODE PLUS ONE ( SEQ # OF THE NEXT PACKET
C          TO BE RECEIVED BY THIS NODE )
C
           IF(RCVSVR(JNODE).LE.IWNDSZ)THEN
C             MAKE SURE THE RECEIVER SEQUENCE IS WITHIN THE SPECIFIED
C             LIMIT . THEN UPDATE THE RECEIVE SEQUENCE NUMBER WHICH
C             ACKNOWLEDGES ALL THE PACKETS NUMBERED UP TO AND INCLUDING
C             RCVSVR(JNODE) - 1
C
              ATRIB(39) = RCVSVR(JNODE)
C             UPDATE THE RECEIVE SEQ. #
C
           ELSE
C
              RCVSVR(JNODE) = 1
C             RESET THE RECEIVE SEQ #
C
              ATRIB(39) = RCVSVR(JNODE)
C             DEFINE THE RECEIVE SEQ #.
C
           ENDIF
C
           ATRIB(30) = 2.0
C          APPLICATION LAYER INDICATOR, SEND I-FRAME TO APPLICATION LAYER
C
           ATRIB(42) = 0.0
```

```
C          NON TIMEOUT EVENT
C
           PRSSDLY = BMOVE + SNDCOM
C          PROCESSING TIME TO SEND FRAME TO APPLICATION LAYER
C
           CALL SCHDL(19,PRSSDLY,ATRIB)
C          SCHDL APPLICATION LAYER
C
           CALL SNDRCVACK
C          CORRECT PACKET AT THE RECEIVER ,CALL SND_RCV_ACK
C          SUBROUTINE TO TAKE NECESSARY ACTION FOR SENDING
C          AND RECEIVING ACKNOWLEDGMENTS
C
        ELSE
C
           ATRIB(31) = 2.0
C          OUT OF SEQUENCE PACKET INDICATOR
C
           CALL SNDRCVACK
C          CALL SND_RCV_ACK SUBROUTINE TO TAKE CARE OF
C          SENDING AND RECEIVIG ACKNOWLEDGMENTS
C
           IREJSTS(JNODE) = IREJSTS(JNODE) + 1
C          OUT OF SEQUENCE FRAME , INCREAMENT THE FRAME REJ VAR.
C
           ICOUNT(INODE) = ICOUNT(INODE) + 1
C          INCREMENT NUMBER OF FRAMES IN ERROR FOR THIS
C          NODE
C
           IOTSQCNT(INODE) = IOTSQCNT(INODE) + 1
C          INCREAMENT NUMBER OF OUT SEQ. FRAMES RECEIVED
C          BY THIS NODE
C
           IF(IREJSTS(JNODE).LE.1)THEN
C             CHECK IF THERE IS ALREADY ONE REJ FRAME OUTSTANDING
C             IF NOT , THEN PROCEED TO SEND A REJ FRAME TO THE NODE
C             WHICH TRANSMITTED THE OUT OF SEQ. FRAME
C
              IPKTREJ(JNODE) = IFIX(ATRIB(28))
C             STORE THE SEQUENCE NUMBER OF THE FRAME CAUSED THE REJECT
C             CONDITION
C
              ATRIB(38) = IRCVCNT(JNODE)
C             DEFINE THE SEQ # OF THE NEXT PACKET TO BE RECEIVED
C
              ATRIB(39) = RCVSVR(JNODE)
C             REQUEST OF RETRANSMISSION OF LAST PACKET
C             RECEIVED CORRECTLY BY THIS NODE + 1
C
              ATRIB(40) = 1.0
C             S-FRM INDICATION
C
              ATRIB(41) = IRCVCNT(JNODE) - 1
C             DEFINE THE ACKNOWLEDGE VARIABLE
C
              ATRIB(42) = 0.0
C             NON TIME OUT EVENT
C
              ATRIB(2) = FLOAT(JNODE)
C             NODE THAT WAS RECEIVING BECOMES A TRANSMITTER
```

```
C
            ATRIB(25) = FLOAT(INODE)
C           NODE THAT WAS TRANSMITTING BECOMES A RECEIVER
C
            ATRIB(26) = ATRIB(2) + MAXSTA
C           DEFINE THE REPEAT QUEUE FOR TRANSMITTING NODE
C
            PRSSDLY = SNDSPR
C           PROCESSING TIME TO SEND A SUPERVISORY FRAME
C
            IF(ICHKSTS(JNODE).EQ.0.AND.NODEST(JNODE).EQ.IIDLE)THEN
C              CHECK IF THE TRANSMITTER IS IDLE FOR THIS NODE
C
               ICHKSTS(JNODE) = 1
C              SET THE TRANSMITTER BUSY
C
               CALL FILEM((NCLNR-2),ATRIB)
C              STORE THE SUPERVISORY FRAME IN A TEMPORARY BUFFER
C
               CALL SCHDL(13,PRSSDLY,ATRIB)
C              START THE TRANSMISSION OF S-FRAME AFTER THE PROCESSING
C              TIME
C
            ELSE
C              TRANSMITTER IS BUSY
C
               SFRM(JNODE) = SFRM(JNODE) + 1
C              S-FRM WAITING FOR TRANSMISSION
C
               CALL FILEM((NCLNR-2),ATRIB)
C              STORE THE REJ FRAME IN A TEMPORARY BUFFER
C
            ENDIF
C
         ENDIF
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME:  sndsfrm                          EVENT NUMBER:  13
C ------------------------------------------------------------------
C
C PURPOSE :       This routine implements the source of a node which
C                 transmits only S-FRAMES . A supervisory frame can be
C                 a RR frame which acknowledeges a frame that just received
C                 by the receiver of this node , a REJ FRAME which
C                 indicates that the receiver of this node has received
C                 an out of sequence packet , and finally a RR frame used
C                 for timeout inquiry status purpose. The distinction
C                 among different frames are provided as follows:
C                 if ATRIB(40) = 1 then the frame is a REJ FRAME
C                 if ATRIB(40) = 2 then the frmae is a RR  FRAME used
C                 for acknowledgement purposes
C                 if ATRIB(40) = 3 then the frame is a RR  FRAME used
C                 for timeout inquiry status purpose
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME                TYPE     CLASS   RANGE
C ------------------------------------------------------------------
C params.dat       ! file   ! read  !          !Declare variables and
C                  !        !       !          !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:      EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C rcvifrm , event 12                             sense, event 1
C sndifrm , event 11
C timeout , event 15
C subroutine sndrcvack
C subroutine laptrns
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C copy             ! copies the attributes of an entity from a file, and
C                  ! place them in attribute array
C rmove            ! A SLAM routine which removes an entity from a file
C schdl            ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE SNDSFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER  INODE, JNODE, NEXT, MMFE, NSUCR, ILOC, NNQ, JLOC, I
C
      REAL TEMP, PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(26))
C     DEFINE THE REPEAT QUEUE OF THE NODE WHICH
C     TRANSMITTED THE FRAME
C
```

```
C     PERFORM THE NECESSARY LOGIC TO DETERMINE WHAT KIND OF
C     SUPERVISORY IS TO BE SENT
C
      IF(IFIX(ATRIB(40)).EQ.1.OR.IFIX(ATRIB(40)).EQ.3)THEN
C        CHECK IF WE ARE TO TRANSMIT A REJ  FRAME OR A RR
C        FRAME FOR TIMEOUT INQUIRY STATUS
C
C        START THE SEARCH IN THE QUEUE THAT SUPERVISORY FRAMES ARE
C        STORED FIND THE APPROPRIATE S - FRAME AND START THE TRANS.
C        OF THIS FRAME IMMEDIATELY
C
         JLOC = NNQ((NCLNR-2))
C        DEFINE THE NUMBER OF ENTRIES IN THE FILE WHICH
C        S-FRAMES ARE STORED
C
         NEXT = MMFE((NCLNR-2))
C        START TO TRANSMIT THE S-FRAME THAT HAS BEEN WEAITING
C        FOR TRANSMISSION IN FILE NCLNR - 2
C
   50    IF(NEXT.EQ.0)THEN
C           NO ENTRIES IN SUPERVISORY QUEUE
C
            RETURN
C
         ENDIF
C
         CALL COPY(-NEXT,(NCLNR-2),ATRIB)
C        COPY THE ATTRIBUTES OF THIS ENTRY IN THE ATTRIBUTE ARRAY
C
         IF(INODE.EQ.IFIX(ATRIB(2)))THEN
C           CHECK IF THIS IS THE ENTRY THAT WE ARE LOOKING FOR
C
            IF(IFIX(ATRIB(40)).EQ.1)THEN
C              CHECK IF THE SUPERVISORY FRAME UN QUESTION IS A REJ
C              FRAME,  IF SO INCREMENT NUMBER OF REJ FRAMES TRANSMITTED
C              BY THIS NODE
C
               SFRMCNT(INODE) = SFRMCNT(INODE) + 1
C              INCREMENT NUMBER OF REJ FRAMES TRANSMITTED FROM THIS NODE
C
            ENDIF
C
            ICHKSTS(INODE) = 1
C           SET THE TRANSMITTER BUSY
C
            DO 25 I = 3,20
   25          ATRIB(I) = 0.0
C              INITIALIZE MAP ATTRIBUTES FOR THE S-FRM
C
            ATRIB(11) = MINSZ
C           DEFINE THE FRAME SIZE FOR THE S-FRM
C
            ATRIB(9) = ATRIB(11)/ RATE
C           TRANSMISSION TIME IN MICROSEC.
C
            ATRIB(35) = TNOW
C           DEFINE A VARIABLE TO COLLECT STATISTICS ON THE TIME THAT
C           TAKES THIS FRAME ARRIVE AT THE OTHER NODE'S RECEIVER
C
            ATRIB(42) = 0.0
```

```
C            NON TIME OUT EVENT
C
             PRSSDLY = SENSNG
C            PROCESSING TIME NEEDED TO HAND OVER THIS FRAME
C            TO THE LOWER PART OF THE DATA LINK LAYER
C
             CALL SCHDL(1,PRSSDLY,ATRIB)
C            SCHDL THE TRANSMISSION OF THIS FRAME
C
             CALL RMOVE(-NEXT,(NCLNR-2),ATRIB)
C            REMOVE THE WAITING S-FRAME FROM THE SUPERVISORY
C            QUEUE
C
             RETURN
C
          ELSE
C
             NEXT = NSUCR(NEXT)
C            PROCEED TO THE NEXT ENTRY IN SUPERVIS. QUEUE
C
             GO TO 50
C
          ENDIF
C
       ELSEIF(IFIX(ATRIB(40)).EQ.2)THEN
C            CHECK IF THIS NODE IS TO SEND A RR FRAME FOR
C            ACKNOWLEDEGEMENT PURPOSE
C
          ICHKSTS(INODE) = 1
C         SET THE TRANSMITTER BUSY
C
          DO 55 I = 3,20
55           ATRIB(I) = 0.0
C              INITIALIZE MAP ATTRIBUTES
C
          ATRIB(11) = MINSZ
C         SET THE PACKET SIZE FOR THIS FRAME
C
          ATRIB(9) = ATRIB(11) / RATE
C         COMPUTE LENGTH OF THIS PACKET IN MICROSEC
C
          ATRIB(35) = TNOW
C         STAT. VAR TO RECORD THE TRANSMISSION TIME FOR THIS FRAME
C
          ATRIB(40) = 2.0
C         S - FRAME INDICATOR
C
          ATRIB(42) = 0.0
C         NON TIMEOUT EVENT
C
          RRFRMCNT(INODE) = RRFRMCNT(INODE) + 1
C         INCREMENT # OF RR FRAMES TRANSMITTED FROM THIS NODE
C
          PRSSDLY = SENSNG
C         PROCEESING TIME NEEDED TO HAND OVER THIS FRAME TO THE
C         LOWER PART OF THE DATA LINK LAYER
C
          CALL SCHDL(1,PRSSDLY,ATRIB)
C         BEGIN TRANSMISSION OF THIS FRAME
C
```

```
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rcvsfrm                          EVENT NUMBER: 14
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs following functions: it
C                receives an REJ-FRM and will immediately start the
C                necessary procedures to retransmit the requested
C                frame and any other frames outstanding. To do that,
C                this subroutine will remove all the time out events
C                and scheduled transmission events from the event calender,
C                then immediately will start the retransmission process.
C                Also receives RR frames which acknowledges frames transmitted
C                by this node. Finally is to respond and take necessary
C                actions for RR timeout inquiry status frames.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C laprcv, event 18                      rexmit, event 16
C                                       sndsfrm, event 13
C                                       timeout, event 15
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE RCVSFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE,JNODE,NEXT,NSUCR,MMFE,I,NNQ,JLOC,JTMP
     1,ITMP,IMODE,IACK
C
      REAL SAVE,SAVE1,PRSSDLY
      DIMENSION SAVE(42),SAVE1(42)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      IMODE = IFIX(ATRIB(40))
C     DEFINE WHAT KIND OF SUPERVISORY FRAME IS RECEIVED
C
      GO TO (1,2,3),IMODE
C     BASED ON THE SUPERVISORY TYPE BRANCH TO THE APPROPRIATE PART
C     OF THE CODE
C
    2 CONTINUE
C     RR SUPERVISORY FRAME RECEIVED ACKNOWLEDGES ALL THE TRANSMITTED
C     PACKETS BY THIS NODE MINUS ONE ( N(r) - 1 PROTOCOL VERSION )
```

```
C
C
      CALL SRCH
C     REMOVE THE TIMEOUT EVENT FROM THE EVENT CALENDER
C     FOR THE ACKNOWLEDGED PACKET ( STOP THE TIMER )
C
      CALL RMVACKFRM
C     REMOVE THE ACKNOWLEDGED PACKET FROM THE REPEAT QUEUE
C
      RETURN
C
    1 CONTINUE
C     REJ FRAME RECEIVED,FIRST WE ARE TO CHECK IF SOURCE IS IN
C     TIMER RECOVERY CONDITION ,IF SO THE REJ FRAME WILL BE IGNORED
C     IF SOURCE ISNOT IN TIMER RECOVERY CONDITION ,REJ FRAME WILL BE
C     PROCESSED IMMEDIATELY. THE NODE THAT RECEIVES A REJ FRAME SHOULD
C     RETRANSMIT THE REQUESTED I - FRAME  AND ANY OTHER I - FRAMES
C     OUTSTANDING AS SOON AS POSSIBLE. TO DO THAT FIRST, THE RECEIVE
C     SEQUENCE NUMBER IS USED AS AN ACKNOWLEDGEMENT FOR ALL TRANSMITTED
C     I - FRAMES MINUS ONE . SECOND RESET THE TIMER AND IMMEDIATELY
C     START THE PROCESS OF RETRANSMISSION.
C
      IF(ITMOT(INODE).EQ.1)RETURN
C        CHECK IF SOURCE IS  IN TIMEOUT RECOVERY CONDITION,
C        IF SO,THEN RCVSFRM WOULD BE IGNORED
C
      IF(JCHKSTS(INODE).EQ.1)THEN
C        CHECK IF TRANMISSION OF I - FRAME IS IN PROGRESS,IF SO
C        SUSPEND THE TRANSMISSION OF I - FRAME .
C
         IFRM(INODE) = 1
C        SET A CONDITION THAT THE FRAME IN PROGRESS WOULD BE IGNORED
C        BY THE MAC LEVEL ( WILL BE DISCARDED DURING NORMAL TRANSMISSION)
C
         IABRTPKT(INODE) = IABRTPKT(INODE) + 1
C        INCREMENT NUMBER OF I-FRAMES ABORTED DURING TRANSMISSION
C        BECAUSE OF " REJ " ERROR RECOVERY
C
      ENDIF
C
      DO 25 I = 1,42
C
   25    SAVE1(I) = ATRIB(I)
C        SAVE THE ATTRIBUTES OF THIS FRAME,WHILE SEARCHING THE
C        EVENT CALENDER
C
      ATRIB(32) = 2.0
C     SET A CONDITION THAT INDICATES FROM WHAT ROUTINE THE SRCH
C     SUBROUTINE IS CALLED. NOTE THAT FOR EACH RECOVERY CONDITION
C     THE TIMER RESETTING IS DIFFERENT.
C
      CALL SRCH
C     STOP TIMER FOR TRANSMITTED FRAMES
C
      ATRIB(32) = 0.0
C     RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
      ATRIB(41) = ATRIB(38) - 1
C     SET THE ACKNOWLEDGE VARIABLE EQUAL TO RECEIVER'S VARIABLE
C     MINUS ONE
```

```
C
      CALL RMVACKFRM
C     REMOVE THE ACKNOWLEDGED PACKETS FROMS THE REPEAT QUEUE
C
      DO 46 I = 1,42
   46    ATRIB(I) = SAVE1(I)
C        RESTORE THE ATTRIBUTES VALUES
C
      ATRIB(42) = 0.0
C     NON TIMEOUT EVENT
C
C     THE NODE THAT IS RECEIVED THE REJ FRAME IS TO START
C     THE RETRANSMISSION OF REQUESTED I - FRAME(S)
C
      CALL SCHDL(16,0.0,ATRIB)
C     CALL THE REXMIT EVENT TO TAKE CARE OF I - FRAME
C     RETRANSMISSION
C     BEGIN RETRNASMISSION OF ALL OUTSTANDING FRAMES
C
      RETURN
C
    3 CONTINUE
C     RR TIMEOUT INQUIRY STATUS FRAME RECEIVED ,FIRST  CHECK IF THIS
C     FRAME IS A COMMAND OR A RESPONSE FRAME.  IF IS A COMMAND FRAME THE
C     DATA LINK LAYER MUST SEND A RESPONSE TO THE TRANSMITTING NODE AS
C     SOON AS POSSIBLE.  BUT IF RECEIVED FRAME IS A RESPONSE FRAME , THE
C     TIMEOUT EVENT WILL BE CALLED TO DEAL WITH THIS FRAME APPROPRIATELY
C
      IF(IFIX(ATRIB(33)).EQ.1)THEN
C        CHECK IF THIS FRAME IS A COMMAND FRAME, IF SO PROCESS THE
C        ACKNOWLEDEGEMENT CONTAINED IN THE FRAME AND IMMEDIATELY
C        BEGIN TO SEND A RESPONSE TO THE NODE WHICH TRANSMITTED
C        THE RR FRAME .
C
         DO 75 I = 1,42
   75       SAVE(I) = ATRIB(I)
C           SAVE THE ATTRIBUTES OF THIS FRAME WHILE IN SEARCHING
C           IN THE EVENT CALENDER AND REPEATE QUEUE
C
         ATRIB(41) = ATRIB(41) - 1
C        DEFINE THE ACKNOWLEDGEMENT SEQUENCE NUMBER ( N(r) - 1
C        PROTOCOL VERSION )
C
         CALL SRCH
C        STOP THE TIMER FOR THE ACKNOWLEDGED FRAME(S)
C
         CALL RMVACKFRM
C        REMOVE THE ACKNOWLEDGED FRAME(S) FROM THE REPEATE QUEUE
C
         DO 80 I = 1,42
   80       ATRIB(I) = SAVE(I)
C           RESTORE THE ATTRIBUTE VALUES
C
         IF(ICHKSTS(INODE).EQ.0.AND.NODEST(INODE).EQ.IIDLE)THEN
C           THIS NODE IS TO SEND A RR RESPONSE FRAME TO THE TRANSMITTING
C           NODE, FIRST CHECK IF THIS NODE IS IDLE, IF SO BEGIN THE
C           THE PROCESS TO TRANSMIT THE RR RESPONSE SUPERVISORY FRAME
C           BUT IF SOURCE IS BUSY ,THEN STORE TH RR RESPONSE FRAME
C           IN THE S - FRAME BUFFER .  THIS FRAME WILL BE TRANSMITTED
C           AS SOON AS NODE BECOME IDLE.
```

```
C
            ICHKSTS(INODE) = 1
C           SET THE TRANSMITTER BUSY
C
            ATRIB(33) = 2.0
C           SET THE FRAME TYPE A RESPONSE FRAME
C
            ATRIB(39) = RCVSVR(INODE)
C           DEFINE THE FRAME SEQUENCE NUMBER EXPECTED BY THIS NODE
C
            ATRIB(41) = IRCVCNT(INODE)
C           DEFINE THE ACKNOWLEGE VARIABLE - THIS NODE ACKNOWLEDGES
C           ATRIB(41) - 1 I- FRAMES ( N(r) - 1 PROTOCOL VERSION )
C
            ATRIB(42) = 0.0
C           NON TIMEOUT EVENT
C
            CALL FILEM(NCLNR-2,ATRIB)
C           STORE THIS FRAME IN A TEMPORARY BUFFER , AND BEGIN
C           TRANSMITTING THIS FRAME IMMEDIATELY
C
            PRSSDLY = SNDSPR
C           PROCESSING TIME NEEDED TO SEND A SUPERVISORY FRAME
C
            CALL SCHDL(13,PRSSDLY,ATRIB)
C           ACTIVATE THE TRNSMT PART OF THE DATA LINK LAYER TO
C           PREPARE THIS FRAME FOR TRANSMISSION
C
            RETURN
C
         ELSE
C
            SFRM(INODE) = SFRM(INODE) + 1
C           TRANSMIITER IS BUSY - S - FRAME WAITING FOR TRANSMISSION
C
            ATRIB(33) = 2.0
C           SET THE TYPE OF THIS FRAME TO RESPONSE
C
            ATRIB(39) = RCVSVR(INODE)
C           DEFINE THE SEQUENCE NUMBER OF NEXT FRAME EXPECTED TO BE
C           RECEIVED BY THIS NODE
C
            ATRIB(41) = IRCVCNT(INODE)
C           DEFINE THE ACKNOWLEDGE VARIABLE
C
            ATRIB(42)= 0.0
C           NON TIMEOUT EVENT
C
            CALL FILEM(NCLNR-2,ATRIB)
C           STORE THE RR FRAME IN THE S - FRAME BUFFER, AS SOON AS NODE
C           BECOME IDLE WILL TRANSMIT THIS FRAME
C
            RETURN
C
         ENDIF
C
       ELSEIF(IFIX(ATRIB(33)).EQ.2)THEN
C          CHECK IF THE RECEIVED  FRAME IS A RESPONSE RR TIMEOUT INQUIRY
C          STATUS FRAME , IF SO THE TIMEOUT ROUTINE WILL BE CALLED TO
C          PROCESS THIS FRAME
```

```
C
            ATRIB(42) = 0.0
C           NON TIMEOUT EVENT
C
            CALL SCHDL(15,0.0,ATRIB)
C           CALL THE TIMEOUT EVENT TO PROCESS THE RECEIVED SUPERVISORY
C           FRAME
C
        ENDIF
C
        RETURN
        END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: timeout                          EVENT NUMBER: 15
C ------------------------------------------------------------------
C
C PURPOSE :      This routine is invoked when a transmitting node doesn't
C                receive an acknowledgement for a transmitted I - frame
C                before the timeout priod expires . Since it can happen
C                that an I - frame has been correctly received,but the
C                acknowledgement has either been sent or lost, a RR timeout
C                inquiry status command frame is sent to the other node
C                prior to retransmission to avoid a duplication of the
C                I - frame already sent.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C trnsmt, event 17                     rexmit , event 16
C                                      sndsfrm, event 13
C                                      applclyr, event 19
C                                      subroutine srch
C                                      subroutine rmvackfrm
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE TIMEOUT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE,JNODE,NEXT,NSUCR,MMFE,I,NNQ,JLOC,JTMP
     1,ITMP,ILOC,IACK,NNRSC
C
      REAL SAVE,PRSSDLY
      DIMENSION SAVE(42)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION
C
      ILOC = IFIX(ATRIB(26))
C     DEFINE THE REPEAT QUEUE OF THE TRANSMITTING NODE
C
      IF(ITMOT(INODE).EQ.1.AND.IFIX(ATRIB(40)).EQ.0)RETURN
C        IF TRANSMITTING NODE IS IN TIMEOUT RECOVERY CONDITION
C        THE SUBSEQUENT TIMEOUTS WILL BE IGNORED UNTIL THE TIMEOUT
C        RECOVERY CONDITION IS OVER .
C
      IF(IFIX(ATRIB(40)).EQ.0)THEN
C        CHECK IF THIS ROUTINE IS INVOKED BEACAUSE OF TIMEOUT EXPIRY
```

```
C       OF AN I - FRAME . THIS ROUTINE IS ALSO SCHEDULED WHEN IT RECEIVES
C       A RESPONSE FOR THE COMMAND RR FRAME SENT SOME EARLIER TIME.
C
        ITMOT(INODE) = 1
C       TRANSMITTING NODE IS IN TIMER RECOVERY CONDITION, FURTHER
C       TRANSMISSION OF I - FRAMES IS SUSPENDED UNTIL END OF THE
C       TIMER RECOVERY CONDITION
C
        RRSTS(INODE) = RRSTS(INODE) + 1
C       INCREMENT NUMBER OF RR TIMEOUT INQURY STATUS FRAMES SENT BY
C       THIS NODE
C       BY THIS NODE
C
        IF(JCHKSTS(INODE).EQ.1)THEN
C          CHECK IF TRANSMISSION OF I - FRAME IS IN PROGRESS
C          IF SO ,THE TRANSMITTER WILL SUSPEND THE TRANSMISSION
C          OF I - FRAME AND WILL RETRANSMIT IT AFTER THE TIMER
C          RECORY CONDITION IS OVER
C
           IFRM(INODE) = 1
C          SET A CONDITION THAT WHEN SOURCE NODE IS IN TIMEOUT
C          RECOVERY CONDITION ,THE PACKET ALREADY IN TRANSMISSION
C          PROCESS WILL BE IGNORED BY THE RECEIVER
C
           IABRTPKT(INODE) = IABRTPKT(INODE) + 1
C          INCREMENT NUMBER OF I-FRAMES ABORTED DURING TRANSMISSION
C          BECAUSE OF TIMEOUT RECOVERY CONDITION
C
           ATRIB(41) = PKTCNT(INODE)
C          DEFINE THE ACKNOWLEGE VARIABLE - THIS NODE ACKNOWLEDGES
C          ATRIB(41) - 1 I - FRAMES RECEIVED BY THIS NODE ( N(r) -
C          PROTOCOL VERSINO ) .
C
           ATRIB(32) = 1.0
C          SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C          THE SRCH SUBROUTINE IS CALLED. NOTE THAT FOR EACH
C          RECOVERY CONDITION THE TIMER RESETTING IS DIFFERENT.
C          IN THIS CASE THE TIMER IS STOPPED FOR THE I _ FRAME
C          IN TRANSMISSION AND NOT FOR THE FRAMES ALREADY TRANSMITTED
C
           CALL SRCH
C          STOP TIMER FOR THE FRAME IN TRANSMISSION
C
           ATRIB(32) = 0.0
C          RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
        ENDIF
C
        IF(ICHKSTS(INODE).EQ.0.AND.NODEST(INODE).EQ.IIDLE)THEN
C          THIS NODE IS TO SEND A RR COMMAND FRAME TO THE OTHER
C          NODE ,FIRST CHECK IF THIS NODE IS IDLE ,IF SO BEGIN
C          THE PROCESS TO TTRANSMIT THE RR COMMAND SUPERVISORY FRAME,
C          BUT IF SOURCE IS BUSY ,THEN STORE THE RR COMMAND FRAME
C          IN THE S - FRAME BUFFER , THIS FRAME WILL BE TRANSMITTED
C          AS SOON AS NODE BECOME IDLE
C
           ATRIB(33) = 1.0
C          SET THE FRAME TYPE TO COMMAND FRAME
C
           ATRIB(40) = 3.0
```

```
C            RR TIMEOUT INQUIRY STATUS FRAME INDICATOR
C
             ATRIB(41) = IRCVCNT(INODE)
C            DEFINE THE ACKNOWLEDGE VARIABLE FOR THE OUTGOING SUPERVISORY
C            FRAME .  IT ACKNOWLEDEGES ATRIB(41) -1 I - FRAMES RECEIVED
C            BY THIS NODE
C
             ATRIB(42) = 0.0
C            NON TIMEOUT EVENT
C
             ICHKSTS(INODE) = 1
C            SET THE TRANSMITTER BUSY
C
             CALL FILEM((NCLNR-2),ATRIB)
C            STORE THIS FRAME IN A TEMPORARY BUFFER , AND BEGIN
C            TRANSMITTING THIS FRAME IMMEDIATELY
C
             PRSSDLY = SNDSPR
C            PROCESSING TIME NEEDED TO SEND A SUPERVISORY FRAME
C
             CALL SCHDL(13,PRSSDLY,ATRIB)
C            ACTIVATE THE TRANSMIT PART OF THE DATA LINK LAYER TO
C            PEPARE THIS FRAME FOR TRANSMISSION
C
             RETURN
C
          ELSE
C            SOURCE IS BUSY , STORE THE SUPERVISORY FRAME IN
C            THE S - FRAME BUFFER , THIS FRAME WILL BE
C            TRANSMITTED AS SOON AS NODE BECOME IDLE.
C
             ATRIB(33) = 1.0
C            SET THE FRAME TYPE TO COMMAND FRAME
C
             ATRIB(40) = 3.0
C            RR TIMEOUT INQUIRY STATUS FRAME INDICATOR
C
             ATRIB(41) = IRCVCNT(INODE)
C            DEFINE THE ACKNOWLEDGE VARIABLE FOR THE OUTGOING SUPERVISORY
C            FRAME .  IT ACKNOWLEDGES ATRIB(41) - 1 I - FRAMES RECEIVED
C            BY THIS NODE
C
             ATRIB(42) = 0.0
C            NON TIMEOUT EVENT
C
             SFRM(INODE) = SFRM(INODE) + 1
C            SOURCE IS BUSY - S -FRAME WAITING FOR TRANSMISSION
C
             CALL FILEM((NCLNR-2),ATRIB)
C            STORE THE RR COMMAND FRAME IN THE S - FRAME BUFFER , AS
C            SOON AS SOURCE BECOME IDLE ,IT WILL TRANSMIT THIS FRAME
C
             RETURN
C
          ENDIF
C
        ELSEIF(IFIX(ATRIB(40)).EQ.3)THEN
C          A RR TIMEOUT INQUIRY STATUS RESPONSE FRAME RECEIVED,
C          THE DATA LINK LAYER OF THIS NODE WOULD RESET THE
C          TIMER RECOVERY CONDITION VARIABLE AND THEN PROCESS
```

```
C          THE ACKNOWLEGEMENT CONTAINED IN THE RECEIVED SUPERVISORY
C          FRAME.  IF RECEIVED ACKNOWLEDGMENT SEQUENCE NUMBER
C          IS SMALLER THAN TRANSMITTER'S SEND SEQUENCE NUMBER
C          NODE WILL START THE RETRANSMISSION PROCESS OF ALL
C          OUTSTANDING (UNACKNOWLEDGED) I - FRAMES IMMEDIATELY
C
           ITMOT(INODE) = 0
C          RESET THE TIMER RECOVERY VARIABLE
C
           IACK = IFIX(ATRIB(41))
C          SAVE THE ACKNOWLEDGE SEQUENCE NUMBER
C
           DO 90 I = 1,42
   90         SAVE(I) = ATRIB(I)
C
           ATRIB(41) = FLOAT(IACK - 1)
C          ACKNOWLEDGE ALL THE TRANSMITTED I - FRAMES UP TO
C          RECEIVER SEQUENCE NUMBER MINUS ONE
c
           ATRIB(32) = 0.0
C          SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C          THE SRCH SUBROUTINE IS CALLED
C
           CALL SRCH
C          STOP TIMER FOR ALL ACKNOWLEDGED I - FRAMES
C
           CALL RMVACKFRM
C          REMOVE THE ACKNOWLEDEGED I - FRAMES FROM THE REPEAT
C          QUEUE
C
           DO 95 I = 1,42
   95         ATRIB(I) = SAVE(I)
C
           IF(IACK.LE.PKTCNT(INODE))THEN
C             CHECK IF THE RECEIVER SEQUENCE NUMBER IS LESS THAN OR
C             EQUAL TO SENDER'S SEND SEQUENCE NUMBER, IF SO THEN
C             SENDER WOULD RETRANSMIT ALL OUTSTANDING I - FRAMES
C
              ITMOUTCNT(INODE) = ITMOUTCNT(INODE) + 1
C             NODE IS IN TIMEOUT ERROR RECOVERY CONDITION , THEREFORE
C             INCREMENT NUMBER OF TIMEOUTS BY THIS NODE
C
              ATRIB(32) = 2.0
C             SET A CONDITION THAT INDICATES FROM WHAT ROUTINE
C             THE SRCH SUBROUTINE IS CALLED
C
              CALL SRCH
C             SOURCE IS ABOUT TO RETRANSMIT ALL OUTSTANDING I - FRAMES
C             RESET THE TIMER FOR ALL I _ FRAMES
C
              ATRIB(32) = 0.0
C             RESET THE SRCH ROUTINE INDICATOR VARIABLE
C
              CALL SCHDL(16,0.0,ATRIB)
C             BEGIN THE RETRANSMISSION OF ALL OUTSTANDING FRAMES
C
C
           ELSEIF(NNRSC(INODE).LT.1)THEN
C             THE OTHER NODE ACKNOWLEGES ALL FRAMES SENT BY THIS NODE
C             INFORM THE APPLICATION LAYER THAT DATA LINK LAYER IS
```

```
C             READY TO ACCEPT MORE I - FRAMES
C
              ATRIB(30) = 1.0
C             SET A CONDITION THAT TELLS APPLICATION LAYER THAT DATA
C             LINK LAYER IS EXPECTING I - FRAMES
C
              ATRIB(42) = 0.0
C             NON TIMEOUT EVENT
C
              PRSSDLY = BMOVE + ENQUEU + RCVCOM
C             PROCESSING TIME TO SEND AN I - FRAME FROM APPLICATION
C             LAYER TO THE DATA LINK LAYER
C
              CALL SCHDL(19,PRSSDLY,ATRIB)
C             SCHDL APPLICATION LAYER
C
           ENDIF
C
        ENDIF
C
        RETURN
        END
C
```

```
C ------------------------------------------------------------------
C          TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: rexmit                          EVENT NUMBER: 16
C ------------------------------------------------------------------
C
C PURPOSE :      This routine is to search the repeat queue, and find
C                the requested frame or a frame which caused the timeout.
C                After finding that frame , it will set the transmitter
C                send seq # equal to the seq # of the frame that just
C                found in thwe repeat queue. Then will invoke the source
C                transmitter to transmit this frame and any other frames
C                outstanding.
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C timeout, event 15                       trnsmt, event 17
C rcvsfrm, event 14
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C copy            ! A SLAM routine that copies the attributes of an entity
C                 ! into the attribute array
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
      SUBROUTINE REXMIT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, NSUCR, ILOC, MMFE, IRCVSQ, NNRSC, NNQ
C
      REAL PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(26))
C     DEFINE THE REPEAT QUEUE OF THE NODE WHICH
C     TRANSMITTED THE FRAME
C
      IF(IRXSTS(INODE).EQ.0)THEN
C        CHECK IF THIS NODE IS ALREADY IN RETRANSMISSION CONDITION
C        IF NOT SET THIS NODE IN RETRANSMIT STATUS
C
         NDSTS(INODE) = 1
C        SET THE SOURCE TRANSMITTER IN RETRANSMIT CONDITION (BUSY)
C
      ELSE
C        SOURCE ISNOT IN RETRANSMIT CONDITION
```

```
C
         BFRCNT(INODE) = NNQ(ILOC)
C        DEFINE NUMBER OF OUTSTANDING FRAMES THAT ARE TO BE
C        RETRANSMITTED
C
      ENDIF
C
      IRCVSQ = IFIX(ATRIB(39))
C     SAVE THE RECEIVER SEQ #
C
C     BEGIN SEARCH THE REPAET QUEUE ,AND FIND THE ENTRY NUMBER FOR
C     THE REQUESTED PACKET
C
      NEXT = MMFE(ILOC)
C     START THE SEARCH FOR THE REQUESTED PACKET FROM THE BEGINING
C     OF THE REPEAT QUEUE
C
C     SEARCH THE REPEAT QUEUE, AND FIND THE REQUESTED ENTRY
C
   20 IF(NEXT.EQ.0)THEN
C        END OF REPEAT QUEUE IS REACHED , NO ENTRY FOUND
C        IN THIS BUFFER WITH GIVEN INFORMATION , THEREFORE
C        IGNORE THIS SUBROUTINE AND RETURN
C
C        SINCE FOUND NO ENTRY WITH THE GIVEN INFORMATION
C        RESET ALL THE RETRANSMIT VAR.  FOR THIS NODE
C
         IFRM(INODE) = 0
C        RESET THE VARAIBLE THAT INDICATES AN I-FRAME SHOULD BE
C        SUSPENDED OR NOT
C
         BFRCNT(INODE) = 0
C        RESET THE NUMBER OF OUTSTANDING I-FRAMES WAITING
C        FOR TRANSMISSION
C
         NDSTS(INODE)=0
C        RESET NODE STATUS
C
         IRXSTS(INODE) = 0
C        RESET RETRANSMISSION VARIABLE
C
         ATRIB(30) = 1.0
C        APPLICATION LAYER INDICATOR, LAP ASKS APPLICATION LAYER
C        TO SEND I-FRAMES TO LLC LEVEL
C
         ATRIB(42) = 0.0
C        NON TIMEOUT EVENT
C
         PRSSDLY = BMOVE + ENQUEU + RCVCOM
C        PROCESSING TIME NEEDED TO SEND I-FRAME TO THE LLC LAYER
C
         CALL SCHDL(19,PRSSDLY,ATRIB)
C        SCHDL APPLICATION LAYER
C
         RETURN
C
      ENDIF
C
      CALL COPY(-NEXT,ILOC,ATRIB)
C     COPY THE ATTRIBUTES OF THIS ENTRY INTO THE ATTRIBUTE ARRAY
```

```
C
      IF(IFIX(ATRIB(28)).EQ.IRCVSQ)THEN
C        FOUND THE ENTRY THAT WAS SEARCHING FOR NOW START
C        THE PROCESS OF RETRNASMISSION
C
         IRXSTS(INODE) = 1
C        SET THE SOURCE IN RETRNASM. STATUS
C
         KTMP(INODE) = 1
C        SET A CONDITION THAT TELLS THE SOURCE THE START OF
C        RETRANSMISSION OF I-FRAMES. SO SOURCE WOULD NOT INCR.
C        THE SEND SEQ # AT THE FIRST TIME.
C
         ISND(INODE) = IFIX(ATRIB(28))
C        RESET THE SEND SEQ # EQUAL TO THE SEQ # OF
C        REQUESTED FRAME
C
         IF(ICHKSTS(INODE).EQ.0.AND.NODEST(INODE).EQ.IIDLE)THEN
C           CHECK IF TRANSMITTER IS IDLE
C
            BFRCNT(INODE) = NNQ(ILOC)
C           # OF OUTSTANDING I-FRAMES TO BE RETRANSMITTED
C
            ICHKSTS(INODE) = 1
C           SET THE TRANSMITTER BUSY
C
            NDSTS(INODE) = 2
C           SET THIS NODE IN RETRANSMISSION STATUS
C
            CALL SCHDL(17,0.0,ATRIB)
C           START THE RETRANSMISSION PROCESS IMMEDIATELY
C
         ENDIF
C
      ELSE
C
         NEXT = NSUCR(NEXT)
C        PROCEED TO THE NEXT ENTRY OF THIS FILE
C
         GO TO 20
C        CONTINUE THE SEARCH FOR THE ENTRY IN QUESTION
C
      ENDIF
C
      RETURN
      END
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: trnsmt                          EVENT NUMBER: 17
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:
C                perform the transmissin and retransmission of
C                I - frames ,piggyback the acknowledgement of the
C                outgoing frame,  start the retransmission timer
C                for the outgoing I - frames , and finally handover
C                the frame to the transmitter for transmission.
C
C
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C subroutine laptrns                      sense,    event 1
C rexmit  , event 16                      timeout,  event 15
C sndifrm , event 11                      applclyr, event 19
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C copy            ! A SLAM routine that copies the attributes
C                 ! of an entity from a file into the attribute
C                 ! array
C
C ------------------------------------------------------------------
      SUBROUTINE TRNSMT
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, ILOC, MMLE, NEXT, NPRED, NNQ, I
     1, ITMP, J, NNRSC

      REAL PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(26))
C     DEFINE THE REPEATE QUEUE FOR THIS NODE
C
      ITMP = ISND(INODE)
C     SAVE SEQ. NUM OF THE LAST PACKET TRANSMITTED
C
      IF(IRXSTS(INODE).EQ.1)THEN
C        CHECK IF SOURCE NODE IS IN THE RETRANSMIT STATUS
C
```

```
            NDSTS(INODE) = 2
C           SOURCE RETRANSMITTING
C
            IRXMTCNT(INODE) = IRXMTCNT(INODE) + 1
C           INCREMENT NUMBER OF RETRANSMITTED PACKETS
C           FROM THIS NODE
C
         ENDIF
C
         IF(KTMP(INODE).EQ.0)THEN
C           SET A CONDITION THAT WHEN SOURCE IS IN RETRANS. STATUS
C           THIS PART OF THE CODE WOULD NOT BE CONCIDERED AT THE
C           TIME
            ISND(INODE)=ISND(INODE)+1
C           INCREMENT THE SEND SEQUENCE NUMBER FOR
C           THIS NODE ( V(s) - PROTOCOL VERSION )
C
            IF(ISND(INODE).GT.IWNDSZ)THEN
C              CHECK FOR WINDOW ROTATION
C
               ISND(INODE) = 1
C              RESET THE SEND SEQ #
C
            ENDIF
C
         ELSE
C
            KTMP(INODE) = 0
C
         ENDIF
C
         NEXT = MMLE(ILOC)
C        SET THE NEXT POINTER TO THE LAST PACKET ARRIVED
C        IN THE TRANSMISSION QUEUE
C
      20 IF(NEXT.EQ.0)THEN
C           NO ENTRIES IN THIS FILE
C
            IF(IRXSTS(INODE).EQ.1)THEN
C              JUST FINISHED RETRANSMITTING ALL THE
C              REQUESTED PACKETS
C
               ISND(INODE) = ITMP
C              SET THE SEND SEQ. NUM EQUAL TO THE LAST PACKET
C              WAS TRANSMITTED
C
               IRXMTCNT(INODE) = IRXMTCNT(INODE) - 1
C              DECREMENT # OF RETRANS. FRAMES,NO FRAMES TO SEND
C              THIS VAR. SHOULD NOT HAVE BEEN INCREMINTED
C
               BFRCNT(INODE) = 0
C              RESET THE NUMBER OF OUTSTANDING I - FRAMES
C              FOR TRANSMISSION - NO PACKET TO RETRANSMIT
C
               ICHKSTS(INODE) = 0
C              SET THE SOURCE TO IDLE STATUS
C
               ATRIB(30) = 1.0
C              SET A VARIABLE THAT INFORMS THE APPLICATION
```

```
C            LAYER THAT LINK LAYER IS READY TO TRANSMIT THE
C            NEXT FRAME
C
             ATRIB(42) = 0.0
C            NON TIMEOUT EVENT
C
C
             PRSSDLY = BMOVE + ENQUEU + RCVCOM
C            COMPUTE THE PROCESSING TIME NEEDED FOR PACKET
C            TRANSFER FROM APPLICATION LAYER TO THE LINK LAYER
C
             CALL SCHDL(19,PRSSDLY,ATRIB)
C            SCHDL APPLICATION LAYER
C
             RETURN
C
          ELSE
C
C            THE ENTRY WAS NOT FOUND , RESTORE ALL THE PARAMETERS
C            BEFORE THIS SUBROUTINE WAS CALLED
C
             ISND(INODE) = ITMP
C            RESTORE THE SEQ. NUM OF THE LAST PACKET TRANS.
C
             ICHKSTS(INODE) = 0
C            SET THE SOURCE TO IDLE STATUS
C
             JFRM(INODE) = 0
C            NO FRAMES WAITING FOR TRANSMISSION
C
             ATRIB(30) = 1.0
C            SET A VARIABLE THAT INFORMS THE APPLICATION LAYER
C            THAT LINK LAYER IS READY TO TRANSMIT THE NEXT FRAME
C
             PRSSDLY = BMOVE + ENQUEU + RCVCOM
C            COMPUTE THE PROCESSING TIME NEEDED TO TRANSFER
C            ONE PACKET FROM APPLICTION LAYER TO LLC LAYER
C
             CALL SCHDL(19,PRSSDLY,ATRIB)
C            SCHDL APPLICATION LAYER
C
             RETURN
C
          ENDIF
C
       ENDIF
C
       CALL COPY(-NEXT,ILOC,ATRIB)
C      COPY THE PACKET ATTRIBUTES FROM TRANSMISSION QUEUE
C      INTO THE ATTRIBUTE ARRAY
C
       IF(IFIX(ATRIB(28)).EQ.ISND(INODE))THEN
C         CHECK IF THE COPIED PACKET IS THE NEXT PACKET TO
C         BE TRANSMITTED
C
          ATRIB(41) =KRCV(INODE)
C         PIGGYBACK THE ACKNOWLEDGEMENT TO THE OUTGOING
C         I - FRAME
C
          JCHKSTS(INODE) = 1
```

```
C          TRANMITTER BUSY ....  I - FRAME TRANSMISSION
C
           ICHKSTS(INODE) = 1
C          TRANSMITTERT BUSY ....  FRAME TRANSMISSION .  NOTE THAT
C          THIS VARIABLE IS USED FOR ALL FRAMES BUT " JCHKSTS "
C          IS USED FOR TRANSMISSION OF I-FRAMES ONLY
C
           ATRIB(42) = 1.0
C          TIMEOUT EVENT -
C
           IF(IRXSTS(INODE).EQ.1)THEN
C             IF SOURCE IS IN RETRANSMIT STATUS , DECREMENT
C             THE NUMBER OF FRAMES OUTSTANDING FOR RETRANSMISSION
C
              BFRCNT(INODE) = BFRCNT(INODE) - 1
C             DECREMENT # OF OUTSTANDING FRAMES WAITING FOR
C             TRANSMISSION
C
           ENDIF
C
           CALL SCHDL(15,ATRIB(29),ATRIB)
C          START THE TIMER FOR THE OUTGOING I - FRAME
C          IF THE TRANSMITTING NODE DOESN'T RECEIVE AN
C          ACKNOWLEDGEMENT FOR THIS FRAME BEFORE TIMER
C          EXPIRES , THE PROCEDURES FOR TIMEOUT RECOVERY
C          CONDITION WILL BE FOLLOWED
C          SCHEDULE THE TIME OUT EVENT TO OCCUR AT TIME
C          EQUAL TO ATRIB(29)
C
           ATRIB(42) = 0.0
C          RESET THE TIMER VARIABLE
C
         ELSE
C
           NEXT = NPRED(NEXT)
C          CONTINUE THE SAERCH FOR I-FRAMES IN THE
C          TRANSMISSION QUEUE
C
           GO TO 20
C
         ENDIF
C
         ATRIB(42) = 0.0
C        NON TIMEOUT EVENT
C
         TOTPKTCNT(INODE) = TOTPKTCNT(INODE) + 1
C        INCREMENT NUMBER OF I-FRAMES TRANSMITTED FROM THIS NODE
C
         PRSSDLY = SENSNG + SEND + SNDMSG
C        COMPUTE THE PROCESSING DELAY NEEDED FOR FRAMING,
C        PIGGYBACKING ACKNOW. ,AND HANDING OVER THE FRAME
C        TO THE TRANSMITTER
C        TRANSMISSION PROCESSING DELAY
C
         CALL SCHDL(1,PRSSDLY,ATRIB)
C        SCHDL THE SENSE EVENT OF THE MAC LAYER
C
         RETURN
         END
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: laprcv                              EVENT NUMBER: 18
C ------------------------------------------------------------------
C
C PURPOSE :      This ruoitne performs two functions. First, it checks if
C                frame (I - FRAMES only) has a correct CRC(no transmission
C                errors) .If so handover the frame to the upper part of
C                the data link layer. If frame is in error ,no action
C                will be taken and discard the frame. Second , based
C                on the frame type ( I - FRAME or S _ FRAME ) determine
C                which routine should be invoked to process the received
C                frame.If received frame is an I - frame schdl RCVIFRM
C                event ,but if received frame is a S - frame schdl
C                RCVSFRM event.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS ROUTINE:
C ------------------------------------------------------------------
C nodrst, event 10                      rcvifrm, event 12
C                                       rcvsfrm, event 14
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
c
      subroutine laprcv
c
      include 'params.dat'
      integer inode,jnode
c
      real prssdly,rannum,unfrm
c
      inode = ifix(atrib(2))
c     define the source node
c
      jnode = ifix(atrib(25))
c     define the destination node
c
      if(ifix(atrib(40)).eq.0)then
c        check if the received frame is an informatiom frame (I - FRAME)
c
         blkerr = 1 - ((1 - biterr)**atrib(11))
c        compute the probability of this packet
c        being in error
c
         rannum = unfrm(0.0,1.0,7)
c        using a uniform random number generator to
c        determine randomly if this packet is in error
c
         if(rannum.gt.blkerr)then
```

```
c            check if this packet is in error
c
             atrib(31) = 0.0
c            correct packet
c
             prssdly = infpkt
c            processing time needed for information frames
c
             call schdl(12,prssdly,atrib)
c            error free packet , hand over this packet to the
c            data link layer
c
          else
c
             atrib(31) = 1.0
c            packet in error
c
c            invalid frame, discard it and don't invoke the rcvifrm
c            event
c
             icount(inode) = icount(inode) + 1
c            frame in error, increment number of frames in error
c            for this node
c
             idmgpktcnt(inode) = idmgpktcnt(inode) + 1
c            increment number of damaged frames received by this node
c
          endif
c
       elseif(ifix(atrib(40)).ge.1)then
c          check if the received frame is a supervisory frame( S - FRAME)
c
          atrib(2) = float(jnode)
c         the node that was a receiver becomes a transmitter
c
          atrib(25) = float(inode)
c         the node that was a transmitter becomes a receiver
c
          atrib(26) = atrib(2) + maxsta
c         based on the source node determine the repeate queue
c
          prssdly = ackpkt
c         processing time to process the received supervisory
c         frame .
c
          call schdl(14,prssdly,atrib)
c         schdl RCVSFRM event
c
       endif
c
       return
       end
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: applclyr                        EVENT NUMBER: 19
C ------------------------------------------------------------------
C
C PURPOSE :      This routine implements the user level in a Local
C                Area Network.  It performs following functions:
C                First , releases packets from the wait node ( user buffer)
C                and send this frame to the Link Layer. Second receives
C                error free  information frames from the lower layers. Finally
C                collects statistics on the ave network delay and ave node
C                to node delay
C                channel array to determine the node's perception of the
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS ROUTINE
C ------------------------------------------------------------------
C subroutine laptrns
C subroutine rmvackfrm
C trnsmt, event 17
C rexmit, event 16
C timeout, event 15
C rcvifrm, event 12
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C colct    !  A SLAM routine that collects statistics on variables.
C ------------------------------------------------------------------
      subroutine applclyr
c
      include ' params.dat '
c
      integer inode,nnrsc
      real tmdly
c
      inode = ifix(atrib(2))
c     set the source node
c
      if(ifix(atrib(30)).eq.1)then
c        check if applclyr is asked to release frames to lower layer
c
         if(nnrsc(inode).lt.1)then
c           check if this node has any resource available
c           before it can free a packet from the user buffer
c
            call free(inode,1)
c           free one packet from the user buffer
c
         endif
c
      elseif(ifix(atrib(30)).eq.2)then
c        check if application layer is receiving a frame
```

```
c          from the lower layer
c
           xx(115) = xx(115) - 1.0
c          decrement a packet from total packets in the system
c
           tmdly = tnow - atrib(1)
c          calculate user to user delay
c
           call colct(tmdly,(inode+6))
c          collect delay statistics for this node
c
           call colct(tmdly,29)
c          collect delay statistics for total system delay
c
        endif
c
        return
        end
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: colisn
C
C PURPOSE :      This routine is called whenever a collision is detected.
C                Here, statistics are collected and the backoff algorithm
C                is called.
C
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |         | Declares variables and
C                   |       |      |         | extablishes the common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                 ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C event 6, etrans                   event 9, nodset
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C search            | User routine removes scheduled succes event from the
C                   | event calendar.
C schdl             | SLAM routine that schedules events.
C calc              | User routine that implements the backoff algorithm.
C ------------------------------------------------------------------
      subroutine colisn
      include 'params.dat'
      real sav1,sav2,delay
      integer inode
c
c               Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c               Save the prop left marker.
c
      sav1 = atrib(3)
c
c               Save the prop right marker.
c
      sav2 = atrib(4)
c
c               Increment the number of colisns.
c
      icoll(inode) = icoll(inode) + 1
      if ( nodest(inode) .ne. intrmv ) then
c               the colisn did not occur in the interframe spacing, so we must
c               find the succes event that was due to occur and remove it from
c               the event calendar.
c
               call search
      end if
```

```
c
c               Return the prop left marker.
c
      atrib(3) = sav1
c
c               Return the prop right marker.
c
      atrib(4) = sav2
c
c               Increment the number of attempts for this packet.
c
      atrib(6) = atrib(6) + 1
c
c               Set node state to jamming.
c
      nodest(inode) = ijamng
c
c               Set a(5) to indicate a propagation event.
c
      atrib(5) = 0
c
c     To model a bus which detects the collisions and tramsmits
c     jam signals, we must remove two link delays from the time when the
c     end of jam is seen on the bus (with respect to dumb bus model)
c     and prevent their advance scheduling of the idle condition.
c
c               Prepare to schedule end of jam signal after propagation to bus.
c
      delay = lkdly(inode) + rjmtim - 2.0*lkdly(inode)
c
c               Schedule the end of the jam signal.
c
      call schdl(6,delay,atrib)
c
c               Set a(5) to indicate a transmission event.
c
      atrib(5) = 1
c
c               Calculate and wait the backoff time.
c
      call calc
      return
      end
C
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: search
C
C PURPOSE :      This routine removes scheduled SUCCES events from
C                the event calendar in the event of a collision.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |         | Declares variables and
C                   |       |      |         | establishes the common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                 ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C (none)                            subroutine colisn
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C copy              | SLAM routine that copies attributes from elements in
C                   | in a file.
C rmove             | SLAM routine that removes entries from the files.
C ------------------------------------------------------------------
      subroutine search
      include 'params.dat'
      integer next,mmfe,nsucr,inode
c
c               Set inode to the node that generated the packet.
c
      inode=atrib(2)
c
c               Set the pointer "next" to the first location in the event
c               calendar, which is the next event due to occur.
c
      next = mmfe(nclnr)
   10 if ( next .eq. 0 ) then
c                       there are not any events due to occur and the routine
c                       should not have been called
               write(nprnt,20)
   20          format('Error in search ... no more entries.')
               return
      end if
c
c               A negitive sign in front of the rank specified tells slam that
c               the entry (-next) is a pointer to the location rather than the
c               specific rank, copy the attributes of the entry pointed to by
c               next into the atrib array.
c
      call copy(-next,nclnr,atrib)
c
      if ( (atrib(2).eq.inode).and.(atrib(5).eq.1) ) then
c                       this entry is a transmission event that was scheduled
c                       by inode remove this entry from the event calendar and
```

```
c                       return to the colisn subroutine.
c
                call rmove(-next,nclnr,atrib)
        else
c                       The entry copied out was not a transmission event
c                       scheduled by inode, so we should continue the search.
c                       Increment next to the following entry in the event
c                       calendar
c
                next = nsucr(next)
c
c                       Return to copy this entry from the event calendar and
c                       continue the process, until the event has been found.
c
                go to 10
        end if
        return
        end
```

```
C ------------------------------------------------------------------
C         TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: calc
C
C PURPOSE :      This routine determines the backoff time that a node
C                must wait before it may attempt to transmit.  Once the
C                backoff is calculated, a SENSE event is scheduled to
C                occur after this backoff time.  In addition, if a
C                packet has experienced too many collisions, here is
C                where the packet is terminated.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |        | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |        | Declare variables and
C                   |       |      |        | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C event 1, sense                     subroutine colisn
C-----------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C schdl             | SLAM routine that schedules events.
C-----------------------------------------------------------------
      subroutine calc
      include 'params.dat'
      integer iranum,inode,nretry,k
      real unfhi,rannum,bakoff,unfrm,prssdly
c
c               Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c
c               Set the number of RETRANSMISSION ATTEMPTS.
c
      nretry = ifix(atrib(6))
c     maximum collision reached
      if ( nretry .eq. mxcoll ) then
c        check if this frame is I-frame, if so this frame will
c        be discarded because of excessive collisions
         if(ifix(atrib(40)).le.0)then
c
c                       too many colisns have occured!!!  So, increment the
c                       number of packets lost to excessive colisns.
c
                excoll(inode)=excoll(inode)+1
c
c                       Set the node state to terminate.
c
                nodest(inode) = iterm
c
                return
```

```
c
c          check if this is an S-frame, if so we will not discard
c          this frame , and its transmission through the channel will
c          enforced be resetting its attempt limit
           elseif(ifix(atrib(40)).gt.0)then
c
c              reset the attempt limit for this S-frame
               atrib(6) = 1.0
c
               nretry = ifix(atrib(6))
c
              go to 99
c
           endif
c
        else
   99   continue
c
c                       There has not been an excess of colisns, so set
c                       maximum on range for backoff
c
c
c       See page 40 of CSMA/CD standard book 802.3 and page 2-4 of the
c       INTEL LAN components user's manual for a better description of the
c       backoff algorithm that we are about to implement.
c
                k = min(nretry,mxboff)
c
c                       Determine upper bound of uniform distribution.
c
                unfhi = 2.0**float(k) - 1.0
c
c                       Determine the random number from uniform distribution.
c
                rannum=unfrm(0.0,unfhi,iseed)
c
c                       This number must be in integer.
c
                iranum=int(rannum+0.5)
c
c                       Determine the correct backoff amount.
c
                if(iranum.eq.0) then
c                               Enforce the min backoff of one waittime.
c
                        bakoff=waitim
                else
c                               The backoff is a multiple of the slottime.
                        bakoff=float(iranum)*slttim
                end if
        end if
c
c               Set some variables for book keeping.
c
        if(tnow.eq.0.0) nbkoff = 0
c
        nbkoff = nbkoff + 1
c
        if(tnow.eq.0.0) smboff=0.0
c
```

```
      smboff = smboff + bakoff
C
C           Schedule the node to sense the channel when the backoff
C           time has expired.
C
      prssdly = collsn + deffrng
C
      call schdl(1,bakoff + prssdly,atrib)
      return
      end
C
```

```
C ------------------------------------------------------------------
C          TISL CSMA/CD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME:  exdefr
C
C PURPOSE :       In this routine, packets that were placed in the defer
C                 file because the channel was senesed busy in SENSE are
C                 removed from the defer file, then the packets are presented
C                 for retransmission immediately.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME               TYPE     CLASS  RANGE
C ------------------------------------------------------------------
C                    |        |      |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat         | file   | read |         | Declare variables and
C                    |        |      |         | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                    ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C event 1, sense                       event 10, nodrst
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C rmove              | SLAM routine that removes items from a file.
C schdl              | SLAM routine that schedules events
C ------------------------------------------------------------------
      subroutine exdefr
      include 'params.dat'
      real sav1,sav2
      integer inode,inrank,nfind
c
c               Set inode to the node that generated the packet.
c
      inode = ifix(atrib(2))
c
c               Save the end of prop left marker.
c
      sav1 = atrib(7)
c
c               Save the end of prop right marker.
c
      sav2 = atrib(8)
c
c               set inrank equal to the rank of the first entry found in the
c               defer file whose atrib(2) is exactly equal to inode.
c
      inrank = nfind(1,( nclnr-1 ),2,0,atrib(2),0.0)
c
      if (inrank .eq. 0) then
c                       There was not an entry in the defer file from inode
c                       and this routine should not have been called.
              write(nprnt,40)
   40         format('Rank error in exdefr...entry not found')
c
      end if
```

```
c
c               Remove the inrank entry from the defer file and place it's
c               attributes in the atrib buffer.
c
        call rmove(inrank,( nclnr-1 ),atrib)
c
c               Set a(5) to indicate a transmission event.
c
        atrib(5) = 1
c
c               Schedule an immediate sense of the channel.
c


c
        call schdl(1,0.0,atrib)
c
c               Return the end of prop left marker to a(7).
c
        atrib(7) = sav1
c
c               Return the end of prop right marker to a(8).
c
        atrib(8) = sav2
        return
        end
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: laptrns
C
C PURPOSE :      In this routine, packets are terminated after either a
C                successful transmission of after an excessive number of
C                collisions. Also the termination statistics collection
C                is done. Finally start to transmit the next frame . To
C                transmit the next frame , source would transmit any S -frames
C                waiting for transmission before transmitting any I - frames.
C                Also for transmitting I - frames priority is given to
C                I - frames waiting for retransmission.
C                If source node has no S - frames and I - frames waiting
C                for transmission schdule the application layer to send
C                more I - frames to the data link layer.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |         | Declare variables and
C                   |       |      |         | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C event 17 , trnsmt                          event 10, nodrst
C event 13 , sndsfrm
C event 19 , applclyr
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C free              | SLAM routine to free resources.
C colct             | SLAM routine to collect statistics on variables.
C-------------------------------------------------------------------
      subroutine laptrns
c
      include 'params.dat'
      integer inode,badpak,nretry,jnode,nnrsc,i,iloc,nnq
c
      real tsys,taccss,tchnl,totsys,temp,prssdly
c
      inode = ifix(atrib(2))
c     define the source node
c
      jnode = ifix(atrib(25))
c     define the destination node
c
      iloc = ifix(atrib(26))
c     define the repeat queue number
c
      ichksts(inode) =  0
c     set the transmitter idle
c
      jchksts(inode) = 0
```

```
c       set the transmitter idle(I-frames only)
c
        badpak=0
c
c       Increment the number of txmitted packets counter.
c
        icnt(inode) = icnt(inode) + 1
c
        if ( ifix(atrib(6)) .eq. mxcoll ) then
c          The packet had experienced excessive colisns, so set
c          badpak to indicate an unsuccesful packet
           badpak=1
c
           jtermn(inode) = 0
c
           ifrm(inode) = 0
c
           ibadpak(inode) = ibadpak(inode) + 1
c          increment # of I - frames discarded by the MAC level
        end if
c
        if ( badpak .eq. 1 ) then
c          The packet was unsuccesful, so do not include the time
c          to txmit in the throughput.
c
           tgood=0.0
c
c          Add the number of bits in this packet to the total
c          number of unsuccesful bits.
c
           bitsbd=bitsbd + atrib(11)
         else
c          the packet was succesfully txmitted, so calculate the
c          amount of time required to txmit the packet.
c
           tgood=tnow-atrib(12)
c
c          Add the number of bits in this packet to the total
c          number of succesfully txmitted bits.
c
           bitsgd=bitsgd + atrib(11)
        end if
c
c       Add the time to send the packet to the total time spent sending
c       packets succesfully from the specific node.
c
        timegd(inode)=timegd(inode)+tgood
c
        nretry = ifix(atrib(6))
c
c       Increment the number of packets succesful in the specific
c       number of attempts required by this particular packet.
c
        atemps(nretry+1) = atemps( nretry+ 1 ) + 1
c
        if ( ifix(atrib(17)) .eq. 1 ) then
c          The packet was succesful on its first attempt to access
c          the network.  So, increment the number of packets
c          succesfully transmitted from inode on there first
c          attempt to access the network.
```

```
c
         frstat(inode)=frstat(inode)+1
c
      end if
c
c     calculate the amount of time required to access the network
c     for this particular packet.
c
      taccss=atrib(12)-atrib(14)
c
c     Collect statistics on the amount of time to access the net.
c
      call colct(taccss,1)
c
      if (  badpak .eq. 0 ) then
c        The packet was succesfully txmitted, so calculate the
c        total channel delay.
c
         tchnl=tnow-atrib(12)
c
         call colct(tchnl,2)
c
      end if
c
c     calculate the total delay.
c
      totsys=tnow-atrib(35)
c
c     Collect statistics on the total delay through the MAC .
c
      call colct(totsys,4)
c
c     source just finished transmitting a frame, before
c     transmitting the next frame, source is to check some
c     conditions before transmitting next in sequence I-frame.
c     those conditions are as follows:
c     Any S-frames to send ?
c     Is source node in retransmit status?
c
      if(ndsts(inode).gt.0.or.sfrm(inode).ge.1)then
c        check if the source node is in retransmission status,or
c        has a S-FRM waiting to be transmitted
c
         if(sfrm(inode).ge.1)then
c           S-FRM waiting for transmission
c
            atrib(40) = 1.0
            S-FRM indicator
c
            sfrm(inode) = sfrm(inode) - 1
c           decrement number of S-frames waiting for transmission
c
            prssdly = sndspr)
c           compute the processing time to send a supervisory
c           frame
c
            ichksts(inode) = 1
c           set transmitter busy
c
            call schdl(13,prssdly,atrib)
```

```
c            schdl the sndsfrm event for transm. of S-FRM
c
             nodest(inode) = iidle
c            set the node status idle for this node
c
             return
c
          elseif(ndsts(inode).gt.0)then
c            source node is in retransmit status ,start
c            of requested frames
c
             if(itmot(inode).ne.1)then
c               before retransmit make sure that node isnot in
c               timer recovery condition
c
                if(ndsts(inode).eq.1)then
c                  check the status of retransmission recovery
c
                   bfrcnt(inode) = nnq(iloc)
c                  node is about to start retransmission process,
c                  define the number of outstanding frames to be
c                  retransmitted
c
                endif
c
                if(bfrcnt(inode).gt.0)then
c                  check if any more frames outstanding for
c                  retransmission
c
                   do 90 i = 3,20
   90                 atrib(i) = 0
c                     initialize MAC attributes
c
                   ichksts(inode) = 1
c                  set the transmitter busy
c
                   call schdl(17,0.0,atrib)
c                  schdl trnsmt event for transmission of I-FRMS
c
                   nodest(inode) = iidle
c                  set node status to idle
c
                   return
c
                endif
c
             endif
c
          endif
c
c     else
c
c              source node has no S-FRM waiting and isnot in
c              retrans. status. Also the transmitted packet
c              was a I-FRM . Start the process of packet trans.
c
      endif
c
       if(itmot(inode).ne.1)then
c        if node is not in timer recovery condition, then start
```

```
c          sending I-frames
c
           if(jfrm(inode).ge.1)then
c             packets waiting for transmission at the link level
c             don't release any packet from wait node(application
c             level) until the waiting packets are transmitted
c
              jfrm(inode) = jfrm(inode) - 1
c             Decrement number of packets waiting for transmission
c             for this node
c
              do 85 i = 3,20
   85           atrib(i) = 0.0
c               initialize MAC attributes
c
              ichksts(inode) = 1
c             set the transmitter busy
c
              call schdl(17,0.0,atrib)
c             schdl trsnmt event to start the transmission process
c
           else
c
              atrib(30) = 1.0
c             set the application layer indicator that tells
c             application layer to send I-frames to data link layer
c
              atrib(42) = 0.0
c             non timeout event
c
              prssdly = xx(118) + xx(119) + xx(121)
c             calculate processing time needed for application
c             layer to send I-frame to data link layer
c
              call schdl(19,prssdly,atrib)
c             schdl application layer
c
           endif
c
        endif
c
c       set the node node to idle state
        nodest(inode)=iidle
c
        return
        end
C
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: sndrcvack
C ------------------------------------------------------------------
C
C PURPOSE :     This subroutine takes the necessary action for receiving
C               and sending acknowldegements. When a node receives an
C               I - frame , it containes a piggybacked acknowledgement
C               which is used to ack. all I - frames transmitted by this
C               node . If a node has no I - frame to send ( can't piggyback
C               the acknowledgement) then would send a RR supervisory
C               frame to acknowledge the I - frame just received by this
C               node.
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME                TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !         !Declare variables and
C                 !       !      !         !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS ROUTINE:      EVENTS SCHEDULED BY THIS ROUTINE:
C ------------------------------------------------------------------
C rcvifrm, event 12                      sndsfrm, event 13
C                                        subroutine srch
C                                        subroutine rmvackfrm
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C schdl           ! A SLAM routine that schedules event.
C ------------------------------------------------------------------
C
      SUBROUTINE SNDRCVACK
C
      INCLUDE ' PARAMS.DAT '
C
      INTEGER I, INODE, JNODE, ITMP, NNQ
C
      REAL SAVE1, SAVE2, PRSSDLY
      DIMENSION SAVE1(42), SAVE2(42)
C
      INODE = ATRIB(2)
C     DEFINE THE SOURCE NODE
C
      JNODE = ATRIB(25)
C     DEFINE THE DESTINATION
C
      DO 10 I = 1,42
   10    SAVE2(I) = ATRIB(I)
C        SAVE THE ATTIRBUTES FOR THIS FRAME
C
      ATRIB(2) = FLOAT(JNODE)
C     CHANGE THE STATUS OF THIS NODE FROM RECEIVER TO TRANSMITTER
C     THE NODE THAT WAS RECEIVING PACKET BECOMES A TRANSMITTER
C
      ATRIB(25) = FLOAT(INODE)
C     THE NODE THAT WAS TRANSMITTING BECOMES A RECEIVER
C
```

```
      ATRIB(26) = ATRIB(2) + MAXSTA
C     DEFINE THE REPEAT QUEUE BASED ON THE TRANSMITTER
C
      IF(IFIX(ATRIB(41)).EQ.0)GO TO 20
C        CHECK IF THIS IS THE FIRST I - FRAME RECEIVED BY THIS NODE
C        IF SO I - FRAME CONTAINES NO ACKNOWLEDGEMENT THEREFORE
C        BEGIN THE PROCESS TO SEND A RR FRAME
C
      DO 30 I = 1,42
   30    SAVE1(I) = ATRIB(I)
C
      IF(IFIX(ATRIB(41)).GT.IAKVAR(JNODE))THEN
C        CHECK IF THE RECEIVED ACKNOWLEDGMENT IS GREATER THAN
C        ACKNOWLEDGEMENT STATE VARIABLE ( V(a) PROTOCOL VERSION ), IF
C        SO PROCESS THE RECEIVED ACKNOWLEDGEMENT.
C
         CALL SRCH
C        REMOVE THE TIMEOUT EVENT FOR THE ACK. PACKET FROM THE
C        EVENT CALENDER
C
         CALL RMVACKFRM
C        REMOVE THE ACK. PACKET FROM THE REPEAT QUEUE
C
      ENDIF
C
      DO 35 I = 1,42
   35    ATRIB(I) = SAVE1(I)
C
   20 CONTINUE
C     CHECK IF THIS NODE HAS NO I - FRAME TO SEND , IF SO BEGIN TO
C     SEND A RR FRAME WHICH ACKNOWLEDGES THE FRAME JUST RECEIVED
C     BY THIS NODE
C
      IF(NDSTS(JNODE).LE.1)THEN
C        CHECK IF THE RECEIVING NODE IS IN NORMAL I - FRAME TRANSMISSION
C        CONDITION, IF SO TAKE THE PROPER STEPS TO SEND OR NOT TO SEND
C        A RR FRAME
C
         IF(JFRM(JNODE).LE.0.AND.IFIX(ATRIB(31)).EQ.0.AND.NNQ(JNODE)
     1       .LE.0)THEN
C           THIS NODE HAS NO FRAMES WAITING FOR TRANSMISSION
C           TAKE THE NECESSARY STEPS TO TRANSMIT A "RR" FRAME
C           TO THE OTHER NODE . THE RR FRAME WOULD ACKNOWLEDGE
C           ALL THE PACKETS UP TO RECEIVER'S SEQ. NUMBER MINUS
C           ONE. THE RR FRAME WOULD BE SENT TO THE OTHER NODE
C           IF AND ONLY IF A PACKET HAS BEEN RECEIVED CORRECTLY
C
            IF(ICHKSTS(JNODE).EQ.0.AND.NODEST(JNODE).EQ.IIDLE
     1          .AND.ITMOT(JNODE).EQ.0)THEN
C              MAKE SURE THE SOURCE ISNOT TRANSMITTING AND ALSO
C              ISNOT IN ERROR RECOVERY CONDITION
C
               ATRIB(40) = 2.0
C              RR SUPERVISORY FRAME INDICATOR
C
               ATRIB(41) = ATRIB(27)
C              SET THE SEQUENCE NUMBER OF LAST PACKET RECEIVED CORRECTLY
C              EQUAL TO THE ACKNOWLEDGE VARIABLE
C
               ATRIB(42) = 0.0
```

```
C                NON TIMEOUT EVENT
C
                 ICHKSTS(JNODE) = 1
C                SET THE TRANSMITTER BUSY
C
                 PRSSDLY = ACKPKT
C                PROCESSING DELAY TO SEND A SUPERVISORY FRAME
C
                 CALL SCHDL(13,PRSSDLY,ATRIB)
C                START SENDING THE RR-FRAME AFTER THE PROCESSING TIME
C
              ENDIF
C
           ENDIF
C
         ELSEIF(NDSTS(JNODE).EQ.2)THEN
C           THE RECEIVING NODE IS IN RETRANSMIT STATUS, IF THIS NODE HAS NO
C           OUTSTANDING FRAMES THEN PREPARE TO SEND A RR FRAME TO ACKNOW.
C           I - FRAME JUST RECEIVED BY THIS NODE
C
           IF(ITMOT(JNODE).EQ.0.AND.IFIX(ATRIB(31)).LE.0)THEN
C             MAKE SURE THAT NODE IS NOT IN TIMER RECOVERY CONDITION
C             AND THE RECEIVED I-FRAMEIS VALID ( NO TRANSMISSION
C             ERROR OR OUT OF SEQUENCE)
C
              IF(ICHKSTS(JNODE).EQ.0.AND.NODEST(JNODE).EQ.IIDLE
     1           )THEN
C                MAKE SURE THE TRANSMITTER IS NOT BUSY
C
                 ATRIB(40) = 2.0
C                RR SUPERVISORY FRAME INDICATOR
C
                 ATRIB(42) = 0.0
C                NON TIMEOUT EVENT
C
                 ATRIB(41) = ATRIB(27)
C                SET THE SEQUENCE NUMBER OF LAST PACKET RECEIVED
C                CORRECTLY EQUAL TO THE ACKNOWLEDGE VARIABLE
C
                 ICHKSTS(JNODE) = 1
C                SET THE TRANSMITTER BUSY
C
                 PRSSDLY = ACKPKT
C                PROCESSING DELAY TO SEND A SUPERVISORY FRAME
C
                 CALL SCHDL(13,PRSSDLY,ATRIB)
C                START SENDING THE RR-FRAME
C
              ELSEIF(BFRCNT(JNODE).LE.1)THEN
C                SINCE TRANSMITTER IS BUSY AND THE SOURCE NODE IS IN
C                RETRANSMIT STATUS, CHECK IF THIS IS THE LAST I-FRAME
C                BEING TRANSMITTED
C
C                TRANSMITTER IS BUSY, STORE THE SUPERVISORY FRAME IN
C                A TEMPORARY BUFFER, AS SOON AS SOURCE RETURNS IDLE
C                WILL TRANSMIT THIS  FRAME
C
                 SFRM(JNODE) = SFRM(JNODE) + 1
C                S-FRAME WAITING FOR TRANSMISSION
C
```

```
                ATRIB(40) = 2.0
C               S-FRAME INDICATOR
C
                ATRIB(41) = ATRIB(27)
C               SET THE ACKNOWLEDGE VARIABLE
C
                ATRIB(42) = 0.0
C               NON TIMEOUT EVENT
C
                CALL FILEM(NCLNR-2,ATRIB)
C               STORE THE SUPREVISORY FRAME IN THE S-FRAME BUFFER
C
             ENDIF
C
          ENDIF
C
       ENDIF
C
C
       DO 40 I = 1,42
   40     ATRIB(I) = SAVE2(I)

C          RESTORE THE ATTRIBUTE VALUES
C
       RETURN
       END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: rmvackfrm
C
C PURPOSE :      This routine is called when a frame have arrived correctly
C                at the receiver. This subroutine removes the acknowledged
C                frame from the repeat queue . Also collects statistics on the
C                average buffer holding time . Finally if the source node
C                is in retransmit status and has finished retransmitting all
C                outstanding I - frames would reset the retransmit variables
C                and schedules the application layer event to send more
C                I - frames to the data link layer.
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME              TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |        | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |        | Declare variables and
C                   |       |      |        | establish common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C applclyr, event 19                    subroutine sndrcvack
C                                       rcvsfrm, event 14
C                                       timeout, event 15
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C colct             | SLAM routine to collect statistics on variables.
C rmove             | SLAM routine to remove  an entity from a file
C copy              | SLAM routine to copy attributes of an entity from
C                   | a file
C-------------------------------------------------------------------
      SUBROUTINE RMVACKFRM
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, MMFE, NSUCR, ISQNM, NNQ, ILOC
     1, NNRSC, JLOC, ITMP, JTMP
C
      REAL BFHLDTM, PRSSDLY
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      ILOC = IFIX(ATRIB(26))
C     DEFINE THE REPEAT QUEUE OF THE NODE
C     WHICH TRANSMITTED THE FRAME
C
      ISQNM = IFIX(ATRIB(41))
C     DEFINE THE ACK. FRAME
C
```

```
      ITMP = NNQ(ILOC)
C     DEFINE # OF OUTSTANDING FRAMES IN THE REPEAT QUEUE
C
      JTMP = 0
C
C     BEGIN TO REMOVE THE ACK. PACKETS FROM THE REPEAT QUEUE, SEARCH
C     THE ENTIRE REPEAT QUEUE FOR ACK. PACKETS AND TAKE THE DELAY
C     STAT. OF THE ACK. PKTS
C
      DO WHILE(JTMP.LE.ITMP)
C
         NEXT = MMFE(ILOC)
C        SET THE POINTER " NEXT " TO THE FIRST LOCATION IN
C        THE RETRANSMISSION BUFFER WHERE THE ACK. FRAME IS STORED
C
    5    IF (NEXT.EQ.0)RETURN
C             NO ENTRIES IN THIS FILE
C
         CALL COPY(-NEXT, ILOC, ATRIB)
C        COPY THE ATTRIBUTES OF THE ENTRY POINTED TO BY
C        NEXT INTO THE ATTRIBUTE ARRAY.
C
         IF(IFIX(ATRIB(27)).LE.ISQNM)THEN
C           CHECK IF COPIED ENTRY IS LESS THAN OR EQUAL TO THE ACK.
C           PACKET
C
            CALL RMOVE(-NEXT, ILOC, ATRIB)
C           REMOVE THIS ENTRY FROM THE REPEAT QUEUE SINCE
C           THIS PACKET HAS BEEN ACKNOWLEDGED
C
            BFHLDTM = TNOW - ATRIB(1)
            BFHLDTM = BFHLDTM + DEQUEU
C           CALCULATE THE ER HOLDING TIME FOR THIS PACKET
C
            CALL COLCT(BFHLDTM, 30)
C           COLLECT STATS. ON THE TOTAL AVE. BUFFER HOLDING TIME
C
            IAKVAR(INODE) = IFIX(ATRIB(27))
C           UPDATE THE ACKNOWDLEGE VARIABLE
C
            IF(IRXSTS(INODE).EQ.1.AND.NNQ(ILOC).LE.0)THEN
C              CHECK IF THIS NODE IS IN RETRANSMIT STATUS, IF
C              TRUE ALSO CHECK IF THE LAST FRAME HAS BEEN ACK.
C              BEFORE RESETTING THE RETRANSMIT VARIABLES.
C
               BFRCNT(INODE) = 0
C              RESET THE # OF OUTSTANDING FRAMES FOR TRANSMISSION VARIABLE
C
               IRXSTS(INODE) = 0
C              RESET THE RETRANSMIT STATUS FOR THIS NODE
C
               NDSTS(INODE) = 0
C              SET THE SOURCE NODE IDLE
C
               ATRIB(30) = 1.0
C              APPLICATION FRAME INDICATOR, LAP EXPECTS MORE I-FRAMES
C
               ATRIB(42) = 0.0
C              NON TIMEOUT EVENT
C
```

```
                PRSSDLY = BMOVE + ENQUEU + RCVCOM
C               PROCESSING TIME FOR APPLICATION LAYER TO
C               SEND THE NEXT I-FRAME TO LLC LAYER
C
                CALL SCHDL(19,PRSSDLY,ATRIB)
C               SCHDL APPLICATIOON LAYER
C
             ENDIF
C
          ELSE
C
C            THE ENTRY COPIED ISNOT THE ACKNOWLEDEGED PACKET,
C            CONTINUE THE SEARCH UNTIL THAT PACKET IS FOUND
C
             NEXT = NSUCR(NEXT)
C            POINT TO THE NEXT ENTRY IN REPEAT QUEUE
C
             GO TO 5
C
          ENDIF
C
          JTMP = JTMP + 1
C
       ENDDO
C
       RETURN
       END
C
```

```
C ------------------------------------------------------------------
C         TISL LAPD Simulation System Discrete Event Model
C ------------------------------------------------------------------
C SUBROUTINE NAME: srch
C
C PURPOSE :      This routine removes scheduled TIMEOUT event(s) from
C                the event calendar in the event of a acknowledged
C                I-frame, or timeout recovery condition, REJ recovery
C ------------------------------------------------------------------
C PARAMETER DEFINITIONS:
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C                   |       |      |         | (none)
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat        | file  | read |         | Declares variables and
C                   |       |      |         | establishes the common.
C ------------------------------------------------------------------
C EVENTS SCHEDULED:                  ROUTINES SCHEDULEING THIS SUBROUTINE:
C ------------------------------------------------------------------
C (none)                             subroutine sndrcvack
C                                    rcvsfrm, event 14
C                                    timeout, event 15
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED:
C ------------------------------------------------------------------
C copy              | SLAM routine that copies attributes from elements in
C                   | in a file.
C rmove             | SLAM routine that removes entries from the files.
C ------------------------------------------------------------------
      SUBROUTINE SRCH
C
      INCLUDE ' PARAMS.DAT '
      INTEGER INODE, JNODE, NEXT, MMFE, NSUCR, IRCV, I, NNQ, JTMP, JLOC
     1, IMODE, ITMP
C
      REAL SAVE
      DIMENSION SAVE(42)
C
      INODE = IFIX(ATRIB(2))
C     DEFINE THE SOURCE NODE
C
      JNODE = IFIX(ATRIB(25))
C     DEFINE THE DESTINATION NODE
C
      IRCV = IFIX(ATRIB(41))
C     DEFINE THE ACK. FRAME
C
      IMODE = IFIX(ATRIB(32))
C     INDICATE FOR WHAT RECEOVERY CONDITION THIS SUBROUTINE
C     IS CALLED:  IMODE = 0 - ACKNOWLEDGE I-FRAME(S)
C                 IMODE = 1 - TIMEOUT RECOVERY CONDITION
C                 IMODE = 2 - REJ RECOVERY CONDITION
C
      DO 5 I = 1,42
    5    SAVE(I) = ATRIB(I)
C
```

```
      JLOC = NNQ (NCLNR)
C     DEFINE THE BUFFER CONTENTS OF THE EVENT CALENDER
C
      JTMP = 0
C
C     BEGIN TO REMOVE THE TIMEOUT EVENT FOR THE ACK. PACKET(S) FROM
C     THE EVENT CALENDER
C
      DO WHILE(JTMP.LE.JLOC)
C
         NEXT = MMFE(NCLNR)
C        SET THE POINTER " NEXT " TO THE FIRST LOCATION IN THE EVENT
C        CALENDER , WHICH IS THE NEXT EVENT DUE TO OCCUR
C
   10    IF (NEXT.EQ.0)THEN
C           THERE ARE NOT ANY EVENTS DUE TO OCCUR AND THE ROIUTINE
C           SHOULD NOT HAVE BEEN CALLED
C
            DO 7 I = 1,42
    7          ATRIB(I) = SAVE(I)
C
            RETURN
C
         ENDIF
C
         CALL COPY(-NEXT,NCLNR,ATRIB)
C        COPY THE ATTRIBUTES OF THIS ENTRY INTO THE ATTRIBUTE ARRAY
C
         ITMP = IFIX(ATRIB(2))
C        SAVE THE TRANSMITTING NODE
C
         IF(IMODE.EQ.0.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(42)).EQ.1
     1       .AND.IFIX(ATRIB(27)).LE.IRCV)THEN
C           CHECK IF THIS ENTRY IS AN ACKNOWLEDGED I-FRAME
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           REMOVE THE TIMEOUT EVENT FOR THE ACKNOWLEDGED
C           I-FRAME FROM THE EVENT CALENDAR
C
         ELSEIF(IMODE.EQ.1.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(42)).EQ.1
     1       .AND.IFIX(ATRIB(27)).EQ.IRCV)THEN
C           CHECK IF THIS ENTRY IS FOR TIMEOUT RECOVERY
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           TIMEOUT RECOVERY CONDITION, STOP THE TIMER FOR I-FRAME
C           IN TRANSMISSION AND NOT FOR THE FRAMES ALREADY TRANSMITTED
C
         ELSEIF(IMODE.EQ.2.AND.ITMP.EQ.INODE.AND.IFIX(ATRIB(42)).EQ.1
     1       )THEN
C           CHECK IF THIS ENTRY IS FOR REJ RECOVERY CONDITION
C
            CALL RMOVE(-NEXT,NCLNR,ATRIB)
C           REJ RECOVERY CONDITION , STOP THE TIMER FOR ALL
C           TRANSMITTED I-FRAMES
C
         ELSE
C           THE ENTRY COPIED OUT ISNOT A TIMEOUT EVENT
C           SCHEDULED BY SOURCE NODE SO WE CONTINUE THE SEARCH
C
            NEXT = NSUCR(NEXT)
```

```
C            PROCEED TO THE NEXT ENTRY IN THE EVENT CALENEDER,
C            CONTINUE THE SEARCH UNTIL THE DESIRED ENTRY IS FOUND
C
             GO TO 10
C
          ENDIF
C
          JTMP = JTMP + 1
C
        ENDDO
C
        DO 8 I = 1,42
      8    ATRIB(I) = SAVE(I)
C
        RETURN
        END
```

```
C ------------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------------
C PROGRAM NAME: otput.for      AUTHOR: M.Kashefi          DATE: 12-15-85
C ------------------------------------------------------------------------
C VERSION NUMBER : 1.0           AMENDMENT DATE(S) :
C ------------------------------------------------------------------------
C PURPOSE :     Report specific performance indicators as part of the
C               TISL LAP-D/CSMA-CD simulation system.
C ------------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME           ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C ------------------------------------------------------------------------
C                 !       !      !        ! (none)
C ------------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------------
C                 ! otputlap print user statistics for LAPD model
C                 ! otputmap print user statistics for CSMA/CD model
C ------------------------------------------------------------------------
c       This subroutine is part of SLAM summary reporting and involves a
c       call to another routine called output which has got the actual
c       summary reporting program. This subroutine is called at the
c       end of each simulation run by SLAM processor.
c
        subroutine otput
c
        call otputmap
c       output  MAP user statistics
c
        call otputlap
c       output LAP user statistics
c
        return
        end
C
```

```
C<FF>
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: otputmap.for      AUTHOR: b.j.wood          DATE: 8-15-85
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0            AMENDMENT DATE(S) :
C ------------------------------------------------------------------
C PURPOSE :     Report specific performance indicators as part of the
C               TISL CSMA/CD simulation system.
C ------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME             | TYPE  | CLASS| RANGE  |     DESCRIPTION
C ------------------------------------------------------------------
C                   |       |      |        | (none)
C ------------------------------------------------------------------
C                NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat       | file  |      |        | Declare variables and
C                   |       |      |        | establish common.
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME           |          DESCRIPTION
C ------------------------------------------------------------------
C                   | (none)
C ------------------------------------------------------------------
      subroutine otputmap
      include 'params.dat'
      integer i,totpak,totfrt,totdcd,totcol
      real tottim,stathu,totthu,frtper,ftaper,disper,colpms,peratt
     &     ,rateof,offld,avebof
c
c     We want to execute this routine only when we reach the end of the
c     last in a series of simulatin runs.
c
c     if(nnrun.lt.ifix(xx(27)))return
c             if TRUE, we are not at the end of the simulation, so
c             we should exit this routine.  Else...
c
      totcol= 0.0
      tottim= 0.0
      totfrt= 0.0
      totdcd= 0.0
      totpak= 0.0
c
c     Print the specifications for this simulation.
c
      write(6,4000)
4000  format(1x,'CSMA/CD Performance Analysis',/)
c
      write(6,4001)
4001  format(1x,'Network Parameters',/)
c
      write(6,4002) rate
4002  format(1x,'Line rate in Mbps = ',f10.4,/)
c
      write(6,4003) slttim
4003  format(1x,'The slot time in microsec. = ',f10.4,/)
c
      write(6,4004) waitim
```

```
4004    format(1x,'The interframe spacing in microsec. =',f10.4,/)
c
        write(6,4014) rjmtim
4014    format(1x,'Jam time in microsec. = ',f10.4,/)
c
        write(6,4005) mxcoll
4005    format(1x,'Attempt limit = ',i5,/)
c
        write(6,4006) mxboff
4006    format(1x,'Backoff limit = ',i5,/)
c
        write(6,4007) maxsz
4007    format(1x,'Maximum frame size in bits = ',f10.4,/)
c
        write(6,4008) minsz
4008    format(1x,'Minimun frame size in bits = ',f10.4,/)
c
        write(6,4009) xohbit
4009    format(1x,'Number of overhead bit/packet = ',f10.4,/)
c
        do 9879 i = 1,maxsta
        write(6,4020) i,mxdly(i)
4020    format(1x,'Max lk delay for ',i4,' node in microsec = ',f14.7,/)
9879    continue
c
        write(6,4021)eedly
4021    format(1x,'End-to-End bus delay in microsec = ',f14.7,/)
c
        write(6,4012)
4012    format(1x,'Traffic Parameters',/)
c
        do 7170 i = 1, maxsta
                write(6,4010) i,xx(i)
4010            format(1x,'Average length for node ',i5,' = ',f10.4,/)
c
                write(6,4011) i,1000000.00/xx(49+i)
4011            format(1x,'Average arrival rate for node ',i5,'
     &          = ',f10.4,/)
7170    continue
c
        write(6,4018) sumbit
4018    format(1x,'Total number of bits generated = ',f15.3,/)
c
c   Calculate the offered load.
c
        rateof = sumbit/tnow
        offld  = rateof/rate
c
        write(6,4019) offld
4019    format(1x,'Offered load = ',f10.4,/)
c
c
        write(6,4013)
4013    format(1x,'*************************************************',///)
c
c
c        We wish to report the average number of backoffs.
c
         if (nbkoff.eq.0.0) then
c                If TRUE, there were no backoffs, so we simply report
```

```
c                this fact.
c
                 write(6,7879)
7879             format(1x,'Number of backoffs = 0.0',/)
        else
c                otherwise, we report the average.
c
                 avebof = smboff/float(nbkoff)
                 write(6,7878) avebof
7878             format(1x,'average backoff time = ',f14.4,/)
        endif
c
        do 101 i=1,maxsta
                 totpak=totpak+icnt(i)
  101   continue
c
c       Skip to the top of the next page.
c
        write(6,1905)
 1905   format('1    ')
c
   10 format('    ')
c
c<<<<<<<<<<<<<<<<<<  page 1  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c       Calculate throughput at each station in network.
c
        do 102 i=1,maxsta
                 stathu=timegd(i)/xx(25)
                 write(6,1827) i,timegd(i),stathu
 1827            format(' time sending good packets from station'
     &                  ,i3,' = ',f18.5,',  throughput = ',f7.5)
                 tottim=tottim+timegd(i)
  102   continue
c
        totthu=tottim/xx(25)
c
c       Print throughput statistics.
c
        write(6,10)
        write(6,10)
        write(6,9898) tottim,totthu
 9898   format(' the total time spent with good packets = ',f16.2,
     1          '   total throughput = ',f5.3)
        write(6,10)
        write(6,10)
c
c       Report number of collisions experienced at each station.
c
        do 103 i=1,maxsta
                 write(6,60) i,icoll(i)
   60            format(' total # of collisions that occured at station '
     1                  ,i5,' = ',i8)
                 totcol=totcol+icoll(i)
  103   continue
        write(6,10)
        write(6,10)
c
        colpms=1000*float(totcol)/xx(25)
        write(6,1000) totcol,colpms
```

```
 1000   format(' total # of collisions = ',i8,
     1          '  collisions per millisecond = ',f9.4)
        write(6,1905)
c
c<<<<<<<<<<<<<<<<  page 2  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c       Report the number of packets that were successful on their first
c       access to the network.  Packets in this catagory do not defer
c       or experience collisions.
c
        do 104 i=1,maxsta
               write(6,1957) i,frstat(i)
 1957          format(' # of packets successful on the first'
     1                ,' attempt to access net from ',i3,' = ',i5)
               totfrt=totfrt+frstat(i)
  104   continue
        write(6,10)
        frtper=100*(float(totfrt)/float(totpak))
        write(6,8585) totfrt,frtper
 8585   format('          total # of 1st access = ',i9,
     1          '       % of 1st access = ',f8.4)
        write(6,10)
        write(6,10)
c
c       This reports the number of packets that were successfully
c       transmitted after a particular number of attempts to access the
c       network.  Note that the number of packets successful on
c       the first attempt includes the number of packets successful
c       with zero attempts (packets reported in previous section).
c
        do 106 i=1,mxcoll
                      peratt=100*float(atemps(i))/float(totpak)
                      write(6,1111) i,atemps(i),peratt
 1111                 format(' # of packets successful on attempt',
     &                       ' number ',i2 ,' = ',i8,
     &                         '     percentage = ',f6.3)
  106   continue
        write(6,10)
c
c       Note, packets successful on 1st attempt = packets successful
c       on first access + packets successful after only one attempt.
c
        ftaper=100*(float(atemps(1))/float(totpak))
        write(6,2233) ftaper
 2233   format('              % successful on 1st attempt = ',f8.4)
c
        write(6,1905)
c
c<<<<<<<<<<<<<<<<<<  page 3  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
c
c       Report the number of packets transmitted.
c
        do 107 i=1,maxsta
               write(6,30) i,icnt(i)
   30          format(' total # of packets transmitted from station '
     1                ,i5,' = ',i8)
  107   continue
        write(6,10)
c
        write(6,1807) totpak
```

```
 1807   format(' total # of packets transmitted = ',i13)
        write(6,10)
        write(6,10)
c
c       Report the number of packets that were discarded from the system.
c       These were packets that experienced more than mxcoll collisions.
c
        do 108 i=1,maxsta
                write(6,7777) i,excoll(i)
 7777           format(' # of discarded packets from ',i3,' = ',i5)
                totdcd=totdcd+excoll(i)
  108   continue
        write(6,10)
        disper=100*(float(totdcd)/float(totpak))
        write(6,9696) totdcd,disper
 9696   format('     total # of discarded packets = ',i9,
     1          '     % of packets discarded = ',f8.4)
        write(6,10)
        write(6,10)
c
c       Report totat number of successful bits transmitted.
c
        write(6,2222) bitsgd
 2222   format(' total # of successful bits transmitted = ',i15)
        write(6,10)
c
c       Report the total number of bits that were discarded.
c
c       write(6,3333) bitsbd
 3333   format(' total # of unsuccessful (discarded) bits = ',i8)
c
        return
        end
```

```
C<FF>
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: otputlap.for      AUTHOR: M.Kashefi         DATE: 12-15-85
C ------------------------------------------------------------------
C VERSION NUMBER : 1.0           AMENDMENT DATE(S) :
C ------------------------------------------------------------------
C PURPOSE :     Report specific performance indicators as part of the
C               TISL LAP-D simulation system.
C ------------------------------------------------------------------
C                      PARAMETER DEFINITION
C  NAME            ! TYPE  ! CLASS! RANGE  !     DESCRIPTION
C ------------------------------------------------------------------
C                  !       !      !        ! (none)
C ------------------------------------------------------------------
C                NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat       ! file  !      !        ! Declare variables and
C                  !       !      !        ! establish common.
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME          !         DESCRIPTION
C ------------------------------------------------------------------
C                  ! (none)
C ------------------------------------------------------------------
      SUBROUTINE OTPUTLAP
C
      INCLUDE ' PARAMS.DAT '
      INTEGER I
      REAL THRUPUT,LOAD
      DIMENSION LOAD(25)
C
C     IF(NNRUN .LT. IFIX(XX(27))) RETURN
C
      WRITE(NPRNT,1) XX(111)
    1 FORMAT(/,' TIMEOUT PERIOD = ',F14.2,4X,' MICROSEC ')
C
      WRITE(NPRNT,2)BITERR
    2 FORMAT(/,' BIT ERROR RATE = ',E11.4)
C
      DO 35 I = 1, MAXSTA
C
      WRITE(NPRNT,3)I,TOTPKTCNT(I)
    3 FORMAT(////,' TOTAL # OF PACKETS TRANSMITTED FROM NODE ',I3
     1,' = ',I7)
C
      WRITE(NPRNT,5)I,PKTCNT(I)
    5 FORMAT(/,' TOTAL PACKETS GENERATED FROM NODE ',I3,' = ',I6)
C
      WRITE(NPRNT,6)I,IABRTPKT(I)
    6 FORMAT(/,' TOTAL PACKETS ABORTED DURING TRANSMISSION FROM NODE
     1',I3,' =',I5)
C
      WRITE(NPRNT,8)I,IBADPAK(I)
    8 FORMAT(/,' TOTAL I - FRAMES DISCARDED BY THE MAC LAYER
     1',I3,'=',I5)
C
      WRITE(NPRNT,10)I,ICOUNT(I)
   10 FORMAT(/,' # OF PACKETS IN ERROR FROM NODE ',I3,1X,' = ',I5)
```

```
C
      WRITE(NPRNT,15)I,IOTSQCNT(I)
   15 FORMAT(/,' # OF OUT SEQUENCE PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,20)I,IDMGPKTCNT(I)
   20 FORMAT(/,' # OF DAMAGED PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,25)I,IRXMTCNT(I)
   25 FORMAT(/,' # OF RETRANSMITTED PACKETS FROM NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,30)I,ITMOUTCNT(I)
   30 FORMAT(/,' # OF TIME OUTS BY NODE ',I3,1X,' = ',I5)
C
      WRITE(NPRNT,40)I,INFOBTS(I)
   40 FORMAT(/,' TOTAL NUMBER OF INFORMATION BITS TRANSMITTED FROM
     1 NODE ',I3,' = ',I13)
C
      WRITE(NPRNT,50)I,RRFRMCNT(I)
   50 FORMAT(/,' TOTAL NUMBER OF "RR" PACKETS TRANSMITTED FROM
     1 NODE ',I3,' = ',I5)
C
      WRITE(NPRNT,60)I,SFRMCNT(I)
   60 FORMAT(/,' TOTAL NUMBER OF  "S" PACKETS TRANSMITTED FROM
     1 NODE ',I3,' = ',I5)
C
      WRITE(NPRNT,65)I,RRSTS(I)
   65 FORMAT(/,' TOTAL NUMBER OF " RR TIMEOUT INQUIRY  STATUS " PACKETS
     1 TRANSMITTED FROM NODE',I3,' =',I5)
C
      LOAD(I) = (INFOBTS(I) * ( 1/RATE)) / XX(25)
C     CALCULATE THROUGHPUT FOR THIS NODE
C
      THRUPUT = THRUPUT + LOAD(I)
C     CALCULATE TOTAL THROUGHPUT
C
       WRITE(NPRNT,45)I,LOAD(I),LOAD(I) * ( RATE * 1.0E+6)
   45  FORMAT(//,' THROUGHPUT FOR NODE ',I3,' = ',F8.5
     1,'=',E10.3,'   BITS/SEC  ')
C
   35 CONTINUE
C
      WRITE(NPRNT,55)THRUPUT,THRUPUT * ( RATE * 1.0E+6 )
   55 FORMAT(////,' TOTAL THROUGHPUT = ',F9.6
     1,'=',E10.3,'  BITS/SEC ')
C
      RETURN
      END
```

```
C ------------------------------------------------------------------
C          TISL LAP-D Simulation System Discrete Event Model
C ------------------------------------------------------------------
C EVENT NAME: alloctrsc
C ------------------------------------------------------------------
C
C PURPOSE :      This routine performs the following functions:  when
C                source is not in retransmit status/max window/timeout
C                recovery, then it will seize a resource and lets an
C                entity proceed from network model to the discrete
C                event model .
C ------------------------------------------------------------------
C NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C NAME               TYPE    CLASS  RANGE
C ------------------------------------------------------------------
C params.dat      ! file  ! read !        !Declare variables and
C                 !       !      !        !establish common block
C ------------------------------------------------------------------
C ROUTINES SCHEDULING THIS EVENT:     EVENTS SCHEDULED BY THIS EVENT:
C ------------------------------------------------------------------
C SLAM                                                  none
C ------------------------------------------------------------------
C SUBROUTINES REQUIRED
C ------------------------------------------------------------------
C seize           ! A SLAM routine that allocates resoource to a particular
C                 ! node
C ------------------------------------------------------------------
      SUBROUTINE ALLOC(I,IFLAG)
C
      INCLUDE 'PARAMS.DAT'
      INTEGER NNQ,NNRSC,ILOC,IFLAG,I,JLOC,KLOC
C
      IFLAG = 0
C
D     WRITE(NPRNT,15)
   15 FORMAT(/,' ALLOCATE SUB.  ')
C
      ILOC = IWNDSZ
C     DEFINE WINDOW SIZE
C
      JLOC = I + MAXSTA
C     DEFINE THE REPEAT QUEUE FILE NUMBER
C
      KLOC = NNQ(JLOC)
C     DEFINE NUMBER OF I-FRAMES IN THE REPEAT QUEUE
C
C     ALLOCATE RULE  - SEIZE RESOURCE FOR SOURCE NODE
      IF(NNRSC(I).LE.0.OR.NDSTS(I).EQ.2.OR.KLOC.EQ.ILOC.OR.
     1ITMOT(I).EQ.1) RETURN
C
        CALL SEIZE(I,1)
        IFLAG = -1
C
      RETURN
      END
```

```
C ------------------------------------------------------------------
C       TELECOMMUNICATION AND INFORMATION SCIENCES LABORATORY
C ------------------------------------------------------------------
C PROGRAM NAME: userf.for      AUTHOR: v.s.frost        DATE: 8-15-85
C ------------------------------------------------------------------
C VERSION NUMBER : 1.1         AMENDMENT DATE(S) : 9-1-85 by b.j.wood
C ------------------------------------------------------------------
C PURPOSE :     This function calculates packet lengths for
C               packets created in the TISL CSMA/CD simulation.
C ------------------------------------------------------------------
C                       PARAMETER DEFINITION
C  NAME           ! TYPE   ! CLASS! RANGE   !    DESCRIPTION
C ------------------------------------------------------------------
C imode           !integer! read  ! 1,2,3   ! Determines which rule wil be
C                 !        !      !         ! used to calculate packet size
C ------------------------------------------------------------------
C               NON-LOCAL VARIABLES -- FILE DEFINITIONS
C ------------------------------------------------------------------
C params.dat      ! file   !      !         ! Declare variables and
C                 !        !      !         ! establish common.
C ------------------------------------------------------------------
C                     SUBROUTINES REQUIRED
C    NAME         !         DESCRIPTION
C ------------------------------------------------------------------
C                 ! (none)
C ------------------------------------------------------------------
c
c       This function calculates the length of data packets created in
c       the SLAM driver file.  Userf has three modes as described be-
c       low.  In each mode, the packet size is calculated and the
c       network overhead (xx(83)) is added.  This packet size is then
c       made to conform to the maximum and minimum data packet size
c       permitted by the network.  In each case, the packet size that
c       is returned is in units of BITS!  The average data packet size
c       is given in the SLAM driver file as xx(inode) where inode is
c       the node number of the node that is creating this packet.
c
        function userf(imode)
c
        include 'params.dat'
        integer inode,nnow,imode
        real avelen,userf,xleng,xrand,unfrm,expon
c
c               Get average length from the xx array for the current node.
c
        inode = ifix(atrib(2))
        avelen = xx(inode)
c
c                Initialize the bit counter
c
        nnow = ifix ( tnow )
        if ( tnow. le. 0 ) sumbit = 0.0
c
c                Go to appropriate mode code.
c
        go to(1,2,3)imode
c
c
c       Mode 1: Fixed length packets.
c
```

```
    1   userf = avelen + xx(83)
        if(userf.gt.maxsz) userf = maxsz
        if(userf.lt.minsz) userf = minsz
        sumbit = sumbit + userf
        return
c
c
c       Mode 2: Exponential length data packets
c
    2   userf = expon(avelen,1) + xx(83)
        if(userf.gt.maxsz) userf = maxsz
        if(userf.lt.minsz) userf = minsz
        sumbit = sumbit + userf
c
        return
c
        end
```