To The Graduate Council:

I am submitting herewith a dissertation written by Saran Harinsuta entitled "Intelligent Local Area Network Planning." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Engineering Science.

William G. Sullivan, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of the Graduate School

INTELLIGENT LOCAL AREA NETWORK PLANNING

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Saran Harinsuta

May 1990

DEDICATED

To my parents, อำพล และ ลำยอง หะรินสุต

ACKNOWLEDGEMENTS

The author would like to express his gratitude and sincere appreciation to Professor William G. Sullivan, Chairman of the Committee, for his guidance, criticism, encouragement, patience, time and energy in his supervision of this research.

The author also wishes to thank the members of the committee, Professor Asa O. Bishop Jr., Professor Elden L. DePorter, Professor William T. Snyder, Professor Osama Soliman, and Professor Robert E. Uhrig for their valuable comments, suggestions, encouragement and support.

Finally, the author wishes to express his deepest gratitude and admiration to his parents for love they have given and sacrifices they have made for their children throughout their lives.

ABSTRACT

The objective of this research is to develop an analysis tool for evaluating a token bus topology, supporting the Manufacturing Automation Protocol (MAP). Furthermore this research describes an easy-to-use graphical interface for a complex simulation program. Specifically, the creation of a graphical user interface for MAP network simulation program is illustrated. Simulation of MAP sub-networks interconnected by bridges can be performed without further programming. This software, called ILAN, features a unique graphical user interface with message definition facility and an error-checking inference engine which advises users of any instances of duplicated or omitted information. ILAN's problem-definition phase is simplified by implementing a single graphical user interface for all inputs.

Two main procedures within ILAN consist of a local area networks simulation and an intelligent graphical interface with a unique graphical message definition facility. The simulation procedure includes the ability to model different numbers of multiple MAP sub-networks interconnected by bridges and different numbers of associations between stations using the token bus protocol. Higher layers in the protocol stacks are modeled as queues having variable delays for different messages at different stations. In order to aid users with the difficulty of inputting parameters needed to simulate complex MAP networks, a graphical user interface is developed. By means of this graphical interface, users can describe the MAP hardware configurations as well as message definitions to be simulated. Sequences of messages to be transmitted can be displayed on the screen as users describe them.

TABLE OF CONTENTS

CHAPTER		PAGE
1. INTRODUCTION		1
1.1	Problem Statement	3
1.2	Objectives of the Research	7
1.3	Methodology	7
1.4	Verification and Validation	14
1.5	Overview	15
2. LITERATURE REVIEW		16
2.1	Simulation Overview	16
2.2	Artificial Intelligence Overview	16
2.3	Intelligent Simulation Systems	17
2.4	Graphical Simulation Systems	20
2.5	Traditional Approach to Network Analysis	25
2.6	Analysis of a MAP Network	27
2.7	Summary	29
3. DESCRIPTION OF ILAN		30
3.1	Architecture	30
3.2	Scope and Limitations	30
4. CONSTRUCTION OF SIMULATION PROCEDURE		34
4.1	Overview of ILAN Simulation Procedure	34
4.2	ILAN Simulation Programs	34
4.3	Program Initialization	36
4.4	Entities and Attributes	38
4.5	Events	50
4.6	Fast and Efficient Token Simulation Codes	53
5. CONSTRUCTION OF USER GRAPHICAL INTERFACE		57
5.1	Overview	57
	5.1.1 Menu	59
	5.1.2 Drawing Data Base	59
	5.1.3 Blocks and Attributes	61
5.2	AutoLISP Procedures	64
6. ILLUSTRATION		74
6.1	Executing ILAN	74
6.2	Hardware Definition	74
6.3	Message Definition	98
6.4	Other Parameter Definition	109

CHAPTER PAGE

7. VERIFICATION AND VALIDATION 114
7.1 Verification and Validation Overview 114
7.2 Verification of Token Bus Protocol 115
7.3 Verification of Higher Layers 128
7.4 Validation of Simulation Procedure 136

8. SIMULATION OF AN AUTOMOTIVE PLANT 145
8.1 System Description 145
8.2 Construction of Traffic Flow Data 154
8.3 Input to ILAN 157
8.4 Simulation Results 162

9. CONCLUSIONS 165
9.1 Graphical User Interface 165
9.2 Simulation Procedure 165
9.3 Summary of Contributions 166
9.4 Further Research 168

LIST OF REFERENCES 170

APPENDICES 179
APPENDIX A 180
APPENDIX B 185
APPENDIX C 204
APPENDIX D 223
APPENDIX E 242
APPENDIX F 252

VITA 258

LIST OF TABLES

TABLE		PAGE
2-1.	INSIGHT Nodes	22
4-1.	Association Status	41
4-2.	Attribute Coding	49
4-3.	Restrictions Under Different Operating Systems	50
5-1.	Template Files	63
5-2.	Object Lists	66
5-3.	Parameters for INLAST	72
7-1.	Simulated Mean Token Rotation Times vs. Arrival Rates	116
7-2.	S&S Mean Token Rotation Times and Their Variances vs. Arrival Rates	120
7-3.	S&S Sample of Mean Token Rotation Times vs. Arrival Rates	121
7-4.	Sample of Mean Token Rotation Times and Their Totals	126
7-5.	ANOVA Table for Mean Token Rotation Times	127
7-6.	Simulated Mean Transmission Delays vs. Message Arrival Rates	130
7-7.	Queuing Sample Mean Transmission Delays vs. Arrival Rates	132
7-8.	Sample of Mean Transmission Delays and Their Totals	134
7-9.	ANOVA Table for Mean Transmission Delays	135

LIST OF FIGURES

FIGURE		PAGE
1-1.	Saturn Logical Network Configuration	4
1-2.	Logical Ring of Token Bus	11
2-1.	ANISIM Graphical Output	20
2-2.	INSIGHT Model Description	23
2-3.	SIMNET Nodes	24
3-1.	System Architecture of ILAN	31
4-1.	Simulation Flowchart for ILAN	35
4-2.	Example of Some Message Type Characteristics	40
4-3.	Message Group for Association Status 0	43
4-4.	Message Group for Association Status 7	44
4-5.	Message Spawning	46
4-6.	Token Re-scheduling Codes	55
5-1.	ILAN Root Menu	60
5-2.	Relationships of ILAN Procedures	65
5-3.	INLAST Listing	71
6-1.	Initial Screen of Hardware	75
6-2.	Draw Hardware Screen	77
6-3.	NODE-1N Screen	78
6-4.	NODE-1N Input Text Screen	80
6-5.	Drawing of Station 1	84
6-6.	Attribute Display	86
6-7.	Station 2	88
6-8.	NODE-mN Input Text Screen	89
6-9.	Bridge Insertion	90
6-10.	Input for Sub-network Number 1	92
6-11.	Input for Sub-network Number 2	93
6-12.	Drawing of Sub-networks 1 and 2	94
6-13.	Connections of Sub-networks 1 and 2	95
6-14.	Deletion of Station 2	97
6-15.	New Message Definition	99
6-16.	Drawing of Message Number 1	102
6-17.	Message Numbers 1 and 2	104
6-18.	Definition of Message Number 2	105
6-19.	Drawing of Message Number 2	106
6-20.	Edit Message	108
6-21.	Message Number 2 Deleted	110
6-22.	Deleted Arrow Line	111
6-23.	Other Parameter Input	112
7-1.	Simulated and S&S Mean Token Rotation Times vs. Arrival Rates	123
7-2.	Mean Transmission Delays vs. Message Arrival Rates	133
7-3.	Experiment without Load	137

FIGURE PAGE

7-4. Experiment with Load 138
7-5. Simulation Message Sequences 140
8-1. Nodes Number 60-62 Read Requests and Their Scan Times 147
8-2. Nodes Number 63-66 Read Requests and Their Scan Times 148
8-3. Traffic Flows for Read Requests and Responses 149
8-4. File Transfer Traffic Flows 153
8-5. Construction of Node Number 60 158
8-6. Definition of Message Numbers 7 and 8 160
B-1. Flowchart of Subroutine L7ARR 186
B-2. Flowchart of Subroutine L2ARR 189
B-3. Flowchart of Subroutine CHNGQ 191
B-4. Flowchart of Subroutine TRANSM 194
B-5. Flowchart of Subroutine L2REC 196
B-6. Flowchart of Subroutine L7REC 198
C-1. LOCATI Procedure 204
C-2. STATION Procedure 205
C-3. BRIDGE Procedure 206
C-4. NETWORK Procedure 208
C-5. CONNECT Procedure 209
C-6. CONEC1 Procedure 210
C-7. DLINE Procedure 211
C-8. DRAWMS Procedure 213
C-9. DELMS Procedure 215
C-10. DROTPA Procedure 216
C-11. DELETE Procedure 217
C-12. INLAST Procedure 218
C-13. EVALU Procedure 219
C-14. FIND Procedure 222

CHAPTER 1

INTRODUCTION

The acceleration in the growth of computers and other intelligent devices in the market place has forced users to consider local area networks (LANs). These networks are communication systems built for the purpose of providing transfer of data between a wide variety of devices such as personal computers, mainframes, terminals, mass storage devices and printers. A typical LAN has the following characteristics:

1. The network spans a limited geographic area of 15 kilometers or less.
2. A single organization owns the network, thus eliminating the need to follow government regulations or payment of heavy subscription charges.
3. The network has a high speed data transmission rate that ranges from 1 to 50 megabits per second (Mb/s).

Several transmission media for LANs exist: twisted pair wire, coaxial cable, radio, microwave, infrared and fiber optic. A variety of data rates and bandwidths are associated with each type of transmission medium. Likewise, different methods for inserting the signal on the transmission medium, such as broadband, baseband and carrierband, also exist. Furthermore, there are several different types of LAN access methods. An access method is a set of rules in which a device on the network can signal its intention to transmit data and receives data that is transmitted to it with little or no interference to other devices on the network. The pattern of interconnecting devices on a network is the architecture or topology of the network. Topology is closely related

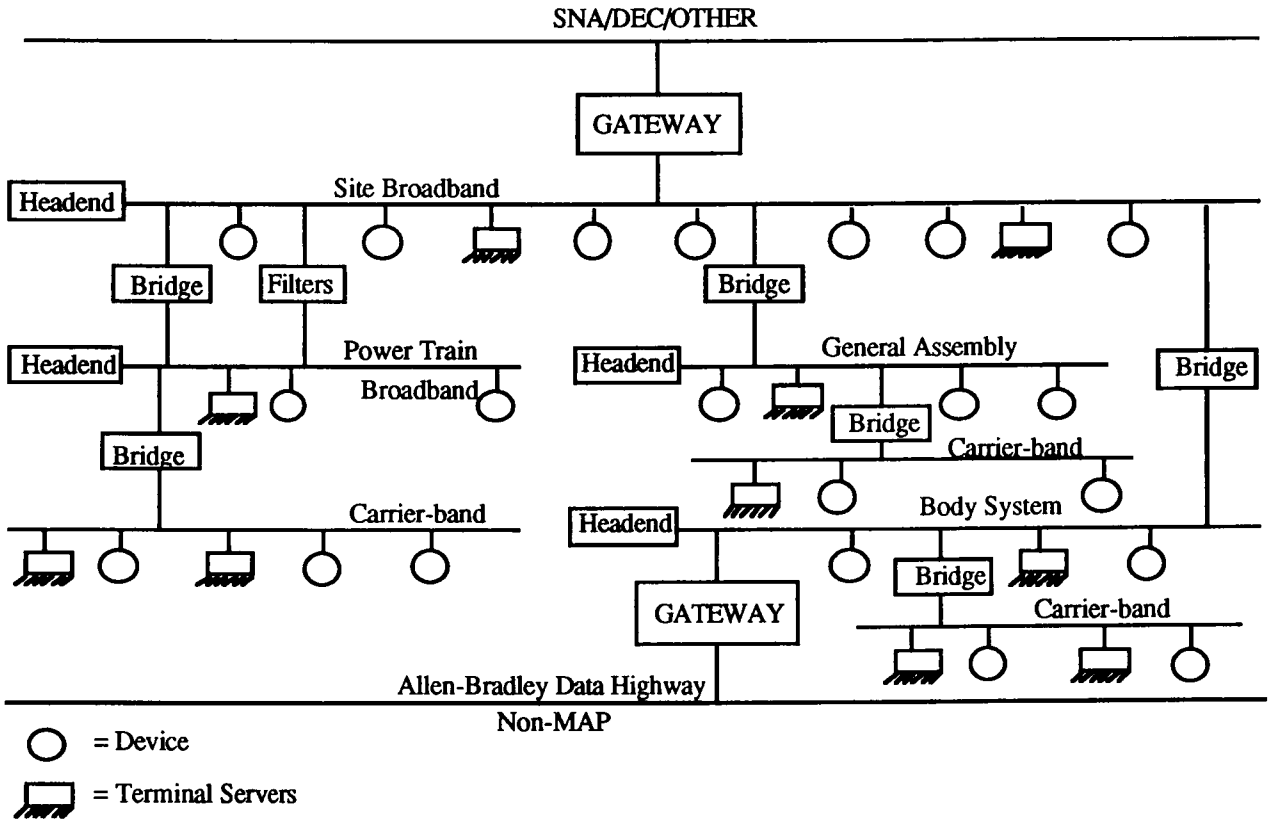
to the access method. It is often chosen as a function of the design of the network. For example, a contention network such as a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) frequently uses a bus topology and coaxial or fiber optic cable. The most prevalent access methods are CSMA/CD [66, 21], token passing ring [4] and token bus [31]. These protocols are Institute of Electrical and Electronic Engineers (IEEE) standards. In December 1980, the IEEE computer society formed the Project 802 committee to establish LAN standards [92]. All three of the above access methods were chosen resulting in three subcommittees: IEEE802.3 (CSMA/CD) [50], IEEE802.4 (token bus) [51] and IEEE802.5 (Token Ring) [52].

During the late 1970s and early 1980s, General Motors (GM) experienced computer communication problems. Only 15 percent of its 40000 programmable devices were able to communicate with one another. Of those that could communicate as much as 50 percent of the total expenses of automation had to be spent on special wiring, custom hardware and interfaces [44]. With an accelerated increase in installed programmable devices, GM realized that the largest single obstacle to the future automation of plants is the communication between these devices. As a result, a task force was formed to investigate use of the Open System Interconnection (OSI) seven-layer reference model developed by the International Standard Organization (ISO). In 1982, GM initiated the development of a communication protocol called Manufacturing Automation Protocol (MAP) [55, 56, 57] based on the OSI model. Other companies were invited to join, forming a MAP Users' Group. Existing standards were selected for the appropriate layers. For example, layers one and two of MAP uses the IEEE

802.4 token bus protocol. The signal insertion techniques specified for MAP include broadband and carrierband.

1.1 Problem Statement

During the last few years, LANs have become one of the fastest expanding segments in the computer and communications marketplace [12]. During the same period, MAP was developed. Since several years were needed to complete a full MAP implementation, migration paths were introduced such that users can upgrade from version 1.0 through 2.x to 3.0. Since MAP version 3.0, which contains the full MAP implementation, is completed, MAP has stabilized and matured. It has gained acceptance in the manufacturing industry. Several applications of MAP have been shown to work very well by companies such as John Deere, Eastman Kodak and General Motors [43]. Now that different programmable devices can communicate on a MAP network, the next logical question to answer is: How do we connect all of these devices so that they can perform efficiently and effectively? The answer is very complex due to the size of the problem. In a large plant such as Saturn's plant at Spring Hill, Tennessee, there are several MAP sub-networks spanning several buildings that connect hundreds of programmable devices, terminal servers, disk servers, etc. Each building may also contain several MAP sub-networks, and each sub-network may be connected by bridges, routers or gateways. Figure 1-1 shows a simplified logical network configuration for Saturn's Spring Hill plant. The possibility for multiple headend remodulators exists for each of the headend remodulators in Figure 1-1. An Intelligent Local Area Network (ILAN) evaluation tool is needed to analyze a



Note: Does not represent actual planned installations but is shown to indicate the various types of devices which will be attached to the broadband and baseband networks.

Fugure 1-1. Saturn Logical Network Configuration

LAN that utilizes MAP. Potential savings are possible through design guidelines derived by the computer simulation tool. Given the information about the signal propagation characteristics of the signal path, traffic rates between stations, type of devices, location of the devices and which network these stations are connected to, ILAN determines the performance and utilization of each sub-network, as well as an evaluation of programmable devices and bridges.

Although MAP is a deterministic protocol, evaluation of its performance is still needed. The deterministic nature of a MAP network is in assuring that each node on the network gets to transmit messages within a certain specified time period. However, in multiple sub-networks with heavy traffic situations, a MAP network may fail to meet other requirements even when each node on the network can transmit messages within the specified time period. For example, the transmission time for transmitting a specific time-critical message may be fast enough for some traffic patterns but not others. An analysis tool would help in predicting whether different traffic patterns or hardware configurations influence the transmission times for these messages.

There is a need for an evaluation tool which leads to monetary savings by considering the following types of questions:

1. How do changes in traffic load or device placement affect the overall performance of the existing network ?
2. How does the annual re-tooling, replacing and adding equipment, common in manufacturing industries such as the automobile industry, affect the performance of the network?

3. Where does addition of carrierband sub-networks improve the message transmission time and the utilization of the network?

Since an evaluation of the performance of a MAP network relies heavily on the input parameters of the network, it is assumed that these parameters can be determined with reasonable accuracy. Examples of some of the input parameters for ILAN include:

1. information such as types of printers, computers and storage devices to be connected in the form of the delays in the protocol stacks for transmitting and receiving each message type at each station;
2. the message transmitting rate (also called the message arrival rate), its statistical distribution and the standard deviation for each station at normal and/or peak operations;
3. the response (back to the station that transmitted the message or the other stations on the MAP network) when the message is received at each station;
4. the geographic location of each station (although no recommendations are given as to the best locations for these stations);
5. the probability of transmitting messages from one station to another ;
6. the message type to be transmitted (either a file transfer, read request, read response, write request or write response);
7. the message length (number of bytes contained in a file) for each message transmitted on the network (The message length for a read request, write request or read response can be expressed in term of number of data elements. The message length for a write response is 1 byte.)

1.2 Objectives of the Research

1. Develop Analysis Tool: The purpose of this research is to develop an analysis tool which can be customized by users to predict the performance of their MAP networks.
2. Verify Analysis Tool: Results from the analysis tool are verified by comparing them to results from analytical techniques. The ability to model the performance of a real system is verified by using input data collected from GM's General Assembly network.
3. Validate Analysis Tool: Computed results from the analysis tool are validated by comparing them against measured results from an experimental MAP network.
4. State Findings: Contributions in areas of simulation and intelligent graphical user interface are presented.

1.3 Methodology

The methodology in ILAN software development consists of constructing three main functions: simulation of a given MAP network, a unique message definition, and an intelligent graphical interface. Each of these tasks are described in more detail below.

- To develop an analysis tool for evaluating the performance of a general MAP network. This includes developing the simulation program as well as determining a set of parameters that influence the performance of a MAP network. These parameters include the number of multiple MAP sub-networks interconnected by bridges and the number of associations between stations using the token bus protocol for layers one to two. Higher layers will be modeled as queues having variable delays for different messages at different stations.

- To model message sequences between stations graphically. The transmission of data between stations can be described by message sequences. Data can be generated, internally, at each origination station to be transmitted to another destination station. When a data message arrives at the destination station, it may create a new message to be transmitted to another station.
- To develop an easy-to-use intelligent graphical interface for describing the MAP hardware configurations as well as the message sequences to be used by the simulation procedure.

Several approaches are taken in order to build the three functions described above. These techniques include Artificial Intelligence (AI) [102] and simulation [105].

Artificial Intelligence and the Graphical Interface

AI is used in manipulating and maintaining a graphical database for ILAN graphical user interface. Artificial Intelligence is chosen over conventional approaches based on numerically-based algorithms because of these advantages:

1. The inference engine is separated from the database or knowledge base. Modification of the data or the knowledge representation can be easily implemented without any changes in the inference engine.
2. The manipulation of symbols can be easily accomplished. This is necessary in order to build different MAP configurations.

Simulation

Computer simulation technique is used to analyze the performance characteristics of the MAP network under consideration. There are several advantages in using a simulation technique rather than an analytical technique. These advantages include:

1. More accurate results under any given input load can be determined.
2. A complex network model can be structured in more detail.
3. Each station in a LAN is not restricted to the same traffic load and distribution.

A computer simulation model is used to evaluate performance characteristics of the network such as the maximum message delay between each station, the average number of messages waiting to be transmitted at various bridges and stations, the utilization of the network. The simulation program is written in SIMAN [79] because it is a general purpose simulation language that is available in DOS and OS/2 operating systems. Furthermore, it has several modeling orientations, adding to the flexibility in writing a simulation procedure. There are three modeling orientations within SIMAN. These orientations are the process orientation, the discrete event orientation and the continuous orientation. The process orientation and the discrete event orientation are used to model discrete change systems. Continuous change systems are modeled with continuous orientation using algebraic difference/ differential equations. Since a computer network is a discrete change system, either the process orientation, the discrete event orientation or the combination of the two can be used to model the system. In the process orientation, the model is constructed by depicting the functional operations with a block diagram, which is a linear sequence of blocks representing specific process functions such as time delays and queues. On the other hand, the discrete event orientation consists of a set of user-written FORTRAN subroutines defining the instantaneous state transitions of the system at each event time. The user-written subprograms may call the SIMAN subprogram library for standard discrete

event modeling functions such as file manipulating, event scheduling, observation recording.

The ILAN simulation procedure is written in discrete event instead of process orientation of SIMAN because more detail and control can be included. Neither a continuous nor a combination of discrete and continuous model is selected because both require a different level of model abstraction. One of the goals in designing ILAN software is to produce a user-friendly analysis tool. A discrete event simulation system is easier to operate because the user is not required to describe the problem in terms of equations.

Several significant components that influence the performance characteristics of a MAP network are modeled in the simulation program. Brief descriptions of some of the more important components are reported below:

1. Token passing scheme -- The IEEE 802.4 token bus access method for local area networks is selected for layer 1 of the MAP network. In the token access method used by MAP during normal, steady-state operation, only the station that has the token has the right to access the medium. A "free token" consisting of a predetermined bit pattern, without any user messages, is passed from a station to the station that has the next lower station address. In order to form a logical ring, the token is then passed from the station with the lowest station address to the station with the highest address. When a station receives a free token and it has a message to transmit, it keeps the token and transmits the message. The destination station copies the message and lets the message continue on the medium to other stations. The station

has a limit on the amount of time it can hold the token. This timer value is called "token holding time" which can be initialized by the network manager. The station can hold the token and continue transmitting until either the time runs out or there are no more messages to transmit. The token is released by passing it to the next station on the logical ring. Thus, if the numbers within the squares in Figure 1-2 represent station addresses then the order of token passing is 121, 110, 94, 80, 15, 121 ...; the physical locations of each station have no impact on the order of the logical ring (dashed line).

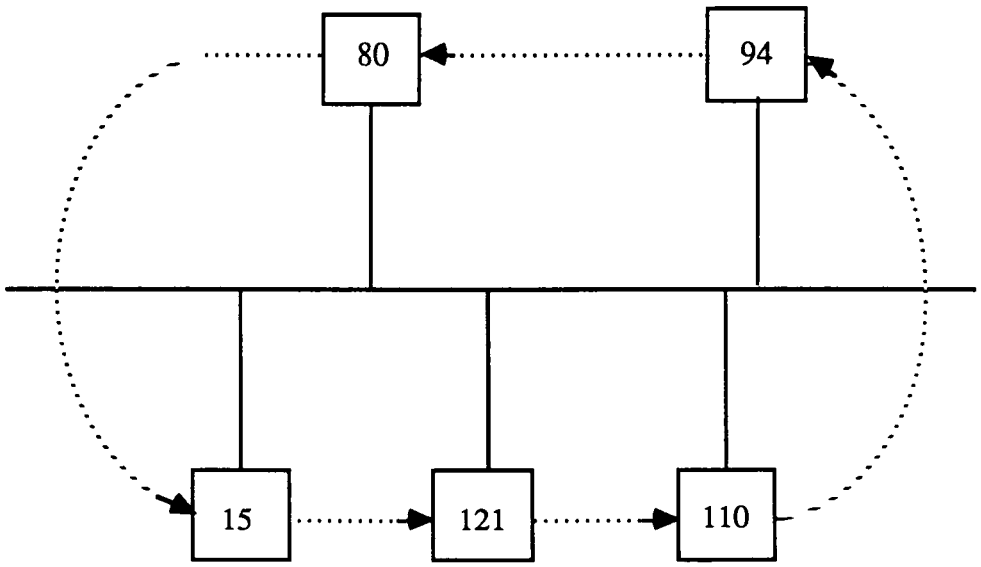


Figure 1-2. Logical Ring of Token Bus

2. Bridges -- A bridge is assumed to be an active station on each side of the sub-network that it is connected to. The IEEE 802.1 Medium Access Control (MAC) bridge [7, 10] is assumed to be used to inter-connect MAP sub-networks. This type of bridge uses a spanning tree algorithm to determine a route from the source station to the destination station without looping [106, 49]. This bridge is transparent to end stations because it requires no participation by an end station in order for the station to make use of it. It automatically initializes, configures and runs itself without any intervention from the network manager. A forwarding database is maintained by the bridge for each of its ports. A typical bridge has two ports, one on each of the two sub-networks that it connects making the inter-network data transfer between the two sub-networks possible. When the bridge receives a frame, it checks the destination address of the frame to see if that address is contained in the forwarding database. If the destination address of the frame is found, the frame is forwarded to the appropriate sub-network. Otherwise, the frame is discarded.
3. Filters -- Frequency filters can be used for a one-way linkage from one network to another. In Figure 1-1, a one-way filter from "Site Broadband" to "Power Train Broadband" would allow a logical token ring to be extended from the "Site Broadband" to "Power Train Broadband." However, the headend at the "Power Train Broadband" could use a different frequency from the "Site Broadband." The filter would limit the token

passing activities only to the "Power Train Broadband" when such a frequency is used.

4. Cable Media: Carrierband and Broadband -- The major differences between these two cable media are transmission speed, existence of headend remodulator and limitation in the number of nodes [17]. The transmission speeds allowable for a broadband system are 1, 5 and 10 Mb/sec. The transmission rate for the carrierband network is 5 Mb/sec. For carrierband networks no headend remodulator is required. The signal from the transmitting station does not have to go through the headend remodulator and be re-transmitted from the headend remodulator back to the receiving station. Thus, the carrierband has only half the propagation delay of the broadband (on average). The size of the carrierband network is limited by the cable characteristics. If a longer length of a cable run of the carrierband network is needed, a device called a repeater can be used.
5. Stations -- Devices on the network that receive and/or transmit data are called stations. Examples of these devices are personal computers, mainframe computers, and robots.
6. Associations -- An association is an establishment of logical connection between two host applications. Before two host applications can exchange information using the Access Control Service Element (ACSE) [56] in the seventh layer, an association between them must be established. The maximum number of associations for a particular station depends on the buffer size of the network adapter at that station. Thus, the number of

associations for each station may be different depending on the type of network adapters used. For simulation purposes, the maximum number of associations for each station can be changed before executing the simulation program. The decision is left to the simulation users as to when to establish and release the association. When a request to establish the association reaches a station that has already reached the maximum number of associations, a denial is transmitted back to the station that requested association. The station that requested association will request association again at later times until the association is established.

1.4 Verification and Validation

The verification of the simulation model is accomplished in two stages. In the first stage, the simulation results of the token passing scheme are compared to the results of a published analytical method. In the second stage, simulated results for the upper layers of the network are compared to results obtained from the M/D/1 queuing model. If the difference in the simulation results and the analytical results for both stages are not statistically significant, then the simulation procedure is verified.

The simulation model is used to predict the performance of General Motor's General Assembly MAP network in Spring Hill, Tennessee. Using real input data, the ILAN software is tested for its usefulness in the real world. In an effort to verify that the simulation model functions correctly, the simulation results are analyzed to determine if it behaves in a logical manner.

The validation of the simulation procedure is performed by comparing its results against measured results obtained experimentally from a MAP network. If the difference in the simulation results and the experimental results are not statistically significant then the simulation procedure is validated.

1.5 Overview

Chapter 1 introduced the problem of developing an analysis tool for analyzing a MAP network. In addition, the objectives and the methodology were also described. In chapter 2, a literature review is presented. A survey of existing systems in intelligence simulation, graphical simulation, network analysis and MAP analysis are covered in chapter 2. In chapter 3, ILAN's system description is given. In chapter 4, the simulation model built into ILAN is presented. Chapter 5 describes the development of graphical user interface of ILAN. The manipulation of graphical user interface is described in chapter 6. Validations are performed in chapter 7. Chapter 8 describes the simulation of the General Assembly Network. Finally, chapter 9 presents the conclusion and recommendations for further research.

CHAPTER 2

LITERATURE REVIEW

2.1 Simulation Overview

In this dissertation the term "simulation" refers to digital computer simulation. Pritsker [81] describes simulation as follows:

> Simulation is the establishment of a mathematical-logical model of a system and the experimental manipulation of it on a digital computer.

Computer programs can be written such that parameters of interest imitate the real systems. The primary reason for building a simulation program rather than a simulator in a laboratory is that it is generally less time consuming and less costly. Simulation can be applied to complex problems that cannot be solved through mathematical analysis. It is used by various disciplines in diverse ways [81].

Several simulation languages exist. A simulation language provides a convenient facility and structure for which a simulation program can be built. The ILAN simulation procedure is written in discrete event orientation of SIMAN simulation language. The reasons for selecting the discrete event orientation of SIMAN simulation language are described in Section 1.3.

2.2 Artificial Intelligence Overview

Artificial Intelligence (AI) started as a sub-field of computer science. Its ultimate goal was to transform computers into a Turing machine [98]. A Turing

machine answering the same questions as a human will produce answers that are indistinguishable from the answers produced by the human. While this goal has yet to be reached, AI has come a long way. During the last decade, AI research has reached the point where it is applicable to the commercial community [71]. Now, applications in AI have sprung up in virtually every field from engineering, chemistry, and medicine to linguistics.

Problems are usually represented as symbols or s-expressions (symbolic expressions) in AI instead of sets of equations performing mathematical (i.e. algorithmic) computations. A symbol is a string of characters that stand for some real-world concept [93]. Most AI systems are implemented in LISP or LISP based languages [22]. There are arguments for [30] and against [41, 54] using LISP.

AI languages considerably cut down on the programming development time of prototype systems. A LISP-based language that has drawing capabilities is chosen for the development of ILAN graphical input facility. It is selected because it is flexible, easy to learn and easy to use.

2.3 Intelligent Simulation Systems

In simulation, there is a strong trend toward developing more powerful modeling environments. The goal is to aid users in the definition of simulation models such that they can be constructed quickly and easily. Several simulation program generators were first developed in the 1970's. Mathewson's DRAFT uses a family of generators to produce source codes in a number of different simulation languages [60, 61, 62]. Davies' approach is to build an independent language generator [20]. Other

simulation program generators soon followed [53, 86, 97]. Most program generators generate programs in well known simulation languages such as GASP [81] or SIMULA [19] from graph diagrams such entity/activity diagram [35] or petri nets [78]. More recently, efforts have been concentrated in two separate modeling methodologies. These are intelligent simulation systems and graphical simulation systems.

Several authors have noted the similarities between simulation and AI [74, 89, 100, 71, 72]. For example, the object-oriented programming in AI came from object-oriented capabilities of the SIMULA simulation language. The current trend is to merge AI with simulation. AI languages have been found to be well suited for implementing simulation programs. For example, there is no need to write link list procedures to store and retrieve events and their attributes in LISP because these procedures are part of the LISP language. Several simulation programs have been written in LISP [84, 26] and Prolog [26, 1, 15, 29]. Applications of expert systems in simulation are found to be well suited for representing knowledge used in making decisions within a simulation model. For example, TAC ASSESSOR and its successor ADM/MTM (Automated Decision Module/McClintic Theater Mode) [47] are large military simulation programs that incorporated roughly two thousand rules in fifteen knowledge bases.

Another application of expert systems in simulation is the analysis of simulation models. TIM (The Insight Master) [34] is an expert system that aids students in debugging simulation models. Overstreet describes a specification language from which an experimental model can be built [75]. Elmaghraby developed an expert system to select a simulation language that matches the user's model with available computer resources utilizing a rule-based knowledge system and a relational database

[25]. NATURALIST [33] is an expert system having the ability to explain the structure and domain of a simulation for inventory control problems.

A significant development in the application of AI to simulation is automatic simulation model generation. Early applications were restricted to generating small models. De Swaan Arons described an expert system that could construct several oscillato r models [22]. As simulationists begin to understand more AI tools, more sophisticated systems were developed. Advanced Information Processing (AIP) interrogates the user and its database to produce simulation models [46]. Further applications of AI have produced other breakthroughs. Natural Language Understanding System (N.L.U.S.) [23, 77] made contributions to the natural language processing field of AI as well as simulation by generating simulation programs from natural language.

By interfacing simulation with AI, more powerful systems than the old program generators are realized. Now several iterations of different simulation models can be constructed, executed and analyzed until a specified goal is reached. A few of these systems have been developed. Ben-Arieh describes a knowledge-based simulation and control system [8] written in Prolog. This system consists of a static database, a dynamic database, a modeling knowledge part and a simulation driver. A manufacturing system is controlled by interrogating the knowledge base in order to make a decision. The decision is simulated for evaluation and improvement. Improvement of the decision triggers a change in the knowledge base.

The development of these new simulation systems are time consuming and involve major changes to existing simulation programs. The development of ILAN

demonstrates a methodology of constructing intelligent graphical interface quickly with little change to the existing simulation programs.

2.4 Graphical Simulation Systems

A graphical simulation system may include a graphical output facility as well as a graphical input. A graphical output facility provides visual animation of simulation output. The initial systems were limited to showing moving alpha-numeric characters or generic moving elements processed in symbolic queues and servers. ANISIM [2] outputs its simulation results using symbolic queues and servers. Figure 2-1 shows outputs of ANISIM. The items enter the system from a source and leave the system by going to a sink. Palme shows moving pictures of the system using alpha-numeric characters on a DEC system 10/20 display as the events changed [76]. Other early graphical output facilities were developed by Hurrion. These facilities include visual interactive simulation methods for job-shop scheduling [36] and industrial management problems [37] executing on a micro-computer [38]. A majority of the graphical

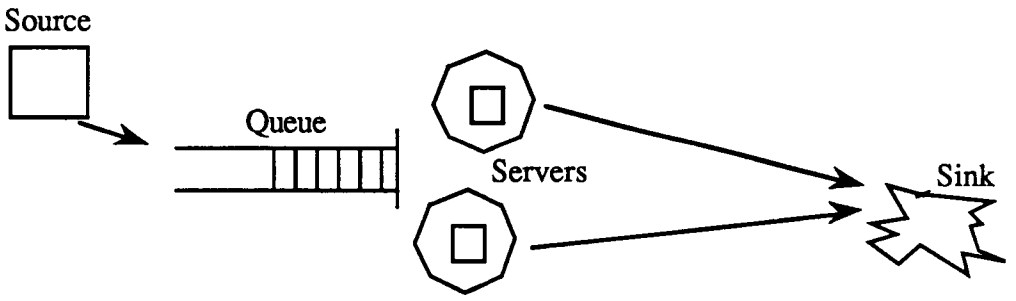
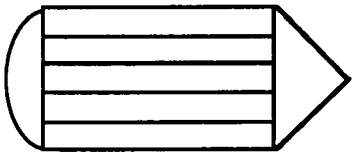


Figure 2-1. ANISIM Graphical Output

simulation systems provide graphical output but not the graphical input [73]. More recent simulation systems that include the graphical output facility for specific systems include animation of an assembly system [45] and aircraft movement (AIRMOV) [96]. The quality of graphical simulation output has improved considerably. Animated objects and drawings of simulation output can be very sophisticated. Examples of general current simulation graphical output facilities include CINEMA [67] and GPSS/PC [18].

A graphical input facility provides the user with the ability to describe the simulation problem visually by inserting and manipulating objects on the computer screen. There are two distinct approaches to graphically describing the simulation problem. These methods are process representation and iconic representation. In process representation, a simulation problem is described using an intermediate graphical representation such as an activity-cycle or network diagram. A good example of this is the graphical network modeling facility in INSIGHT [39, 83]. Table 2-1 shows the meaning of some of the INSIGHT graphic nodes. Figure 2-2 shows a graphical description of a simple simulation problem in INSIGHT. Radios are sent to an inspection station. Some radios fail inspection and are sent to a repair station. After the repairs are completed, radios are returned for re-inspection. Another graphical process representation is SIMNET II.5 [95]. Figure 2-3 shows SIMNET nodes. A graphical process representation of SLAM II simulation language is TESS [32]. Other graphical process representations include IDEF$_2$ [68], GMSS [103], GMBS [104] and GPSS/PC. These graphical inputs can be directly translated to equivalent statements in

Table 2-1. INSIGHT Nodes

Node Type	Node Symbol	Node Description
Branch	Branch Parameter 	Connects two nodes according to their precedence and specifies how transactions branch
Source	Description 	Creates transactions and schedules their arrivals

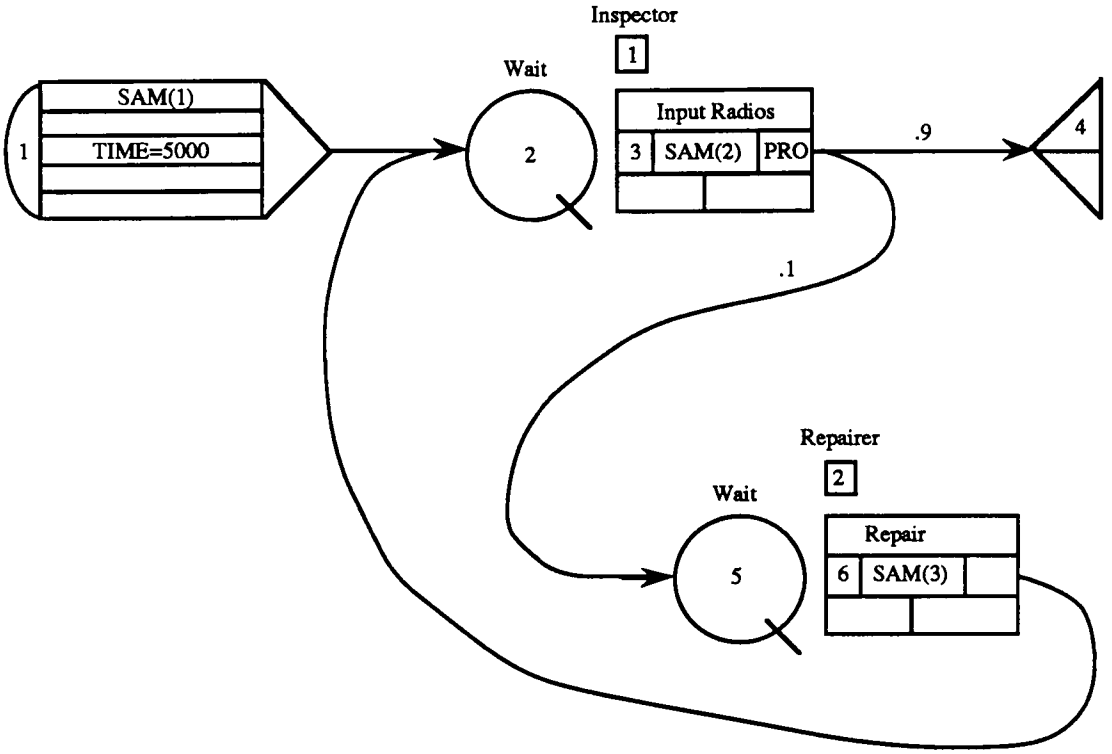


Figure 2-2. INSIGHT Model Description

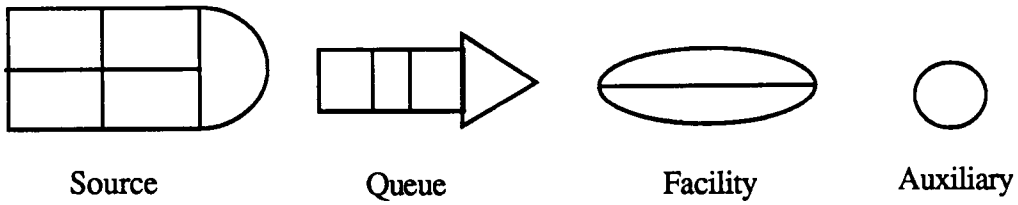


Figure 2-3. SIMNET Nodes

their corresponding queuing network simulation languages. These languages are similar to GPSS, SLAM II and the process orientation of SIMAN.

In iconic representation, a simulation problem is defined by placing and linking icons. Examples of generic model facilities include SimKit [27] and the Performance Analysis Workstation [65]. Problem oriented model facilities include Graphic Workflow Simulator [24] and XCELL+ [16]. The Workflow Simulator is suitable in cases where a layout of the problem can be drawn on the screen of a graphic device and the workflow superimposed on it (i.e. in manufacturing applications). XCELL+ is well suited for modeling manufacturing processes with automatic guided vehicle systems and conveyor systems. It contains a graphical system in which a model is constructed by placing and connecting elements selected from a set of element types. The element types include receiving area, shipping area, buffer, process, work center, auxiliary resources, maintenance facility and links. The iconic representation approach has considerable advantages over process representation. This representation can be used as the basis for visual output. Since the iconic representation closely relates to the real system, its representation is easily used to describe the simulation problem.

The concept of iconic graphical input facility is utilized in ILAN because it closely matches the real system to be represented. Existing graphical input facilities are designed to work only with their specific simulation systems. ILAN employs techniques (as outlined in Chapter 1 Section 1.3) which contain AI techniques that can be incorporated to develop graphical facilities for other existing simulation systems quickly and easily.

2.5 Traditional Approach to Network Analysis

Several analytical and simulation models for analyzing different cable access methods are available. Most of the analytical models are based on queuing theory. Numerous simulation models are used to analyze CSMA, and Ethernet protocols exist [101, 64, 40, 9]. Kuehn developed an analytical model for analyzing the non-exhaustive token passing strategy of the token bus access method [48]. A non-exhaustive token passing strategy permits the station with the token to transmit messages until time runs out. An exhaustive token passing strategy permits the station with the token to transmit messages until its buffers are empty. Other analytical models for the token bus protocol that are capable of describing exhaustive as well as non-exhaustive token passing strategies were developed and were used to compare different token holding strategies [99, 12]. Sethi and Saydam (S&S) developed a different analytical model for the token ring access method [87]. Chauhan and Sethi [11] found that S&S's analytical model can be applied to token bus protocol. They compared analytical results from S&S, Ulug and Kuehn to their own simulation results.

Comparisons of token bus protocol with other LAN protocols were conducted [85]. However, using these models to evaluate LAN designs presents a difficult task. To deal with this situation, a software package written in BASIC was developed by Mok [69]. The user inputs the network configuration and the type of protocol. The system calculates the performance from analytical models and outputs the results in either tabular or graphical form. Another system capable of analyzing different types of protocols is the Local Area Network Design Algorithm (LANAGO) [63]. LANAGO was written in C using analytical models to evaluate either bus, ring or irregular LAN topologies.

A simulation model of token bus protocol was developed using the process orientation of SIMAN [80]. The model is restricted to simulating only four stations on a single network. A very limited number of stations can be modeled using the process orientation of SIMAN running under DOS. Furthermore, long execution times are experienced in light traffic situations.

The advantage of using analytical analysis to evaluate LAN performance is that it can be easily implemented. The problem with using an analytical model is that it is less flexible. Another disadvantage is that these models tend to oversimplify the problem and predict incorrect results in some ranges [11].

The construction of the ILAN simulation procedure adds the ability to simulate multiple token bus sub-networks interconnected by bridges. This capability is not found in any existing token bus analytical model. Furthermore, fast and efficient token passing codes are developed for ILAN simulation procedure. With this capability, the

time needed to simulate a token bus protocol for light traffic situations can be decreased significantly. No other token bus simulation models have this capability.

2.6 Analysis of a MAP Network

Although much information on simulation and analytical models of token bus protocol exists, little information is available on the analysis of MAP networks. Suppan-Borowka [94] analyzed a MAP backbone network through simulation techniques. However, his model did not include the associations between stations. The only two types of messages allowed in the model are "single message" and "message with reply." The reply is transmitted back to the originating station upon reception of the message that requires a reply. The simulation model does not consider the possibilities of generating response messages to be transmitted to different stations other than the originating stations.

Marathe and Smith [59] measured time for establishing associations between a host computer and other devices using Industrial Network Incorporated (INI) network adapters on a single network at various network loads. A queuing model was developed to determine the probability of waiting to establish an association when the network adapter on the host computer reached its limits. Each network adapter has a limited number of associations that can be connected to it. When this limit is reached, no more associations can be established to the adapter until at least one of the established associations is disconnected. This simple queuing model was based on the number of associations available on the host computer, and it neglected the effect of the token bus protocol. Multiple messages must be transmitted back and forth between the

two stations to establish an association. In multiple sub-network situations, the time needed to establish an association between two stations on different sub-networks is significantly influenced by the token bus protocol. Each message transmitted to a destination on different networks needs to be transmitted intermediately to one or more bridges. The time to transmit each message from the bridge is also controlled by the token bus protocol. When there are several nodes on the network and the traffic is heavy, the total time spent in waiting for tokens before each message at each node can be large and cannot be neglected. Thus a simple queuing model that does not consider the token bus protocol is not always valid. Moreover, the time delay for establishing an association between two stations on different sub-networks also depends on loads of both sub-networks and cannot be considered as a constant.

Some simulations of MAP networks [13] have been modeled using a general network simulation language called Network II.5 [70]. These models consider the time for establishing associations to be constant. Although message responses as well as multiple sub-networks can be modeled, the limits on the number of associations between stations cannot be easily included in the simulation. Model description can be input interactively. However, with limited graphical interface input, the model description task is cumbersome and can take several hours to complete even for a small problem.

The ILAN simulation procedure is not limited to modeling "single message" and "message with reply." With the concept of "spawning," a message in ILAN may transmit a message which may spawn other messages at the destination station. Using the spawning concept, the sequences of messages needed to establish associations can

be simulated. Multiple sub-networks interconnected by bridges, the token bus protocol as well as the effect of the limited number of associations at each station can be modeled.

2.7 Summary

From the literature, it is found that AI languages considerably cut down on the programming development time of prototype systems. A LISP language with drawing capabilities is used in the development of ILAN graphical input facility because of its flexibility. An iconic graphical input facility is needed to assist users in the problem-definition phase in ILAN as outlined in Chapter 1 Section 1.3. Though other graphical input facilities exist, they are designed to work only with their specific simulation language. The techniques used in developing such a facility in ILAN are beneficial in the development of other graphical input facilities because they can be conveniently interfaced with existing simulation systems. Based on the literature, there is little interaction between AI and graphical input environments. A graphical input facility can be developed by using AI languages such as LISP.

Through the literature survey, it is found that there are several simulation results for the token bus networks but there is no analysis procedures for a general MAP network. The concept of spawning messages is to be included in the present research so that the sequences of messages and establishment of associations can be simulated. The challenge here is to write an efficient simulation program for a generic MAP network with its underlining token bus protocol.

CHAPTER 3

DESCRIPTION OF ILAN

3.1 Architecture

Figure 3-1 shows the system architecture of ILAN. The two main sub-systems of ILAN consist of a front-end processor and a simulation program. The front-end processor is the user interface that processes the input of the system, which is the user-specified problem-definition. It includes the type of stations to be connected, the geographical locations of each station and the traffic load distributions among stations. The graphical interface front end processor (Block B of Figure 3-1) includes inferences for drawing a data base for the simulation problem-definitions written in AutoLISP [6] language within AutoCAD [5]. After the problem-definition is completed through the input given by the users (Block A), ASCII simulation input files (Block C) are produced and are read by the simulation program when it is executed. The simulation procedure is the SIMAN simulation program (Block D) for a general MAP network.

3.2 Scope and Limitations

The following limitations apply to the ILAN system:

1. The type of protocol under the investigation in layers one and two of the network is limited to the token bus protocol. This protocol is used in MAP [55] which is based on the ISO's seven layers architecture of the OSI reference model. Although the token bus protocol is not as widely used as

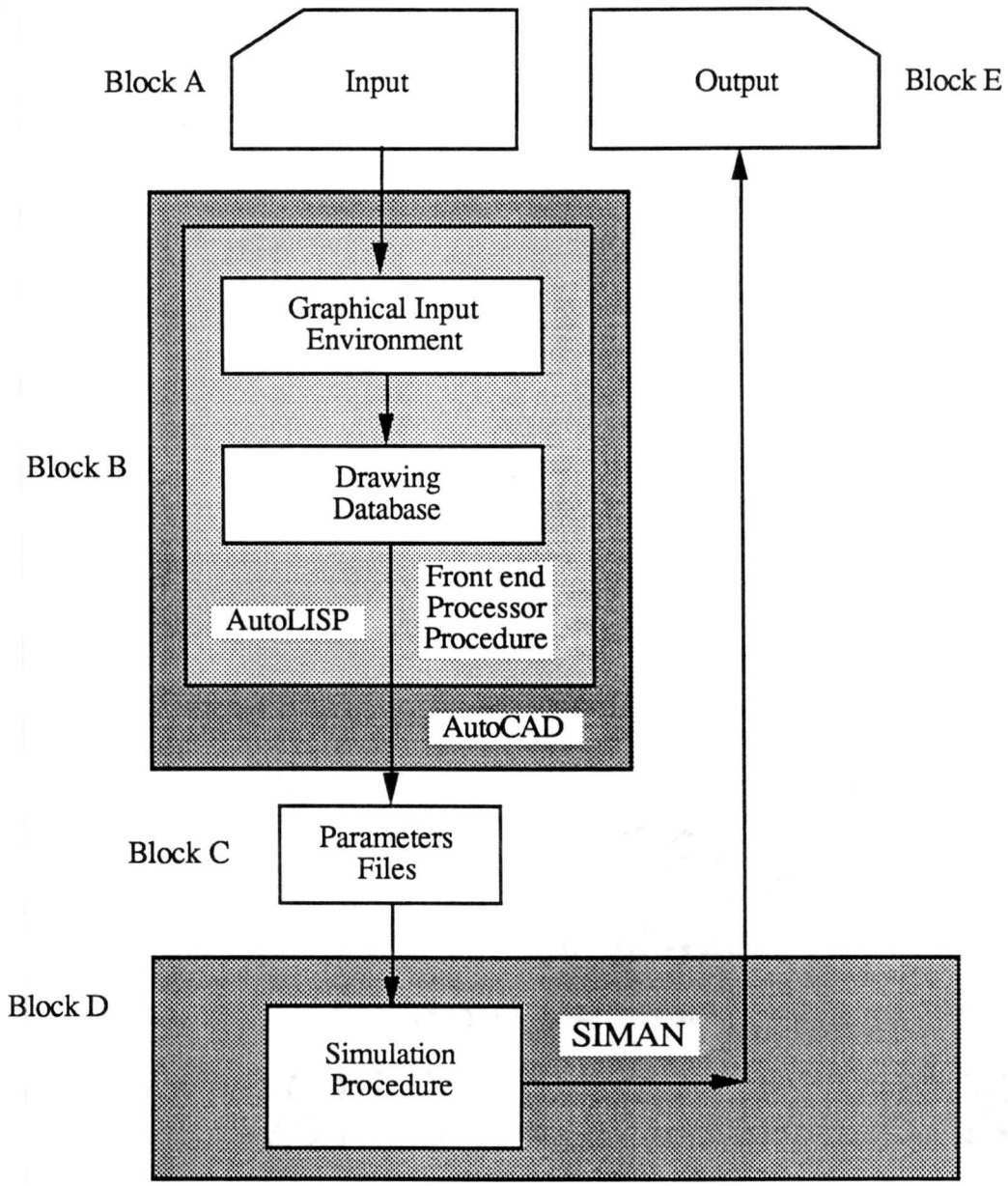


Figure 3-1. System Architecture of ILAN

other protocols such as token ring or CSMA/CD, it is an appropriate protocol for the manufacturing environment where each station cannot wait very long before transmitting a message. The cost of installing the token bus protocol is higher than other protocols because it is more sophisticated. However, experts predict that the cost will come down very fast and the demand for MAP will sharply rise in the future [44]. The performance simulation is limited to layers one and two (the physical and data link layers, respectively). The IEEE802 standards (IEEE802.3, IEEE802.4 and IEEE802.5) conform to these layers. They provide services for medium access, error control, flow control and sequencing.

2. Delays in the protocol stacks associated with receiving and transmitting in the MAP network is simulated using user-described time for each message type at each station. Waiting queues are also established for the stacks as well as for the token bus at each station.
3 It is assumed that the distribution of data packet traffic and size for each station is known at the data link layer.
4. Only one priority of the token bus protocol is considered by the simulation model.
5. The bridges are assumed to have correct information about the locations of all stations on the network.
6. It is assumed that the network is running under normal and steady state conditions. There are no active stations dropping out of the network, nor are any new stations permitted to become active on the logical ring.

7. Devices as well as terminal servers are modeled collectively as stations on the network with no functional differences between them. However, each station may have different message arrival rates and time delays for the protocol stacks for different types of applications. It is assumed that traffic from the terminal servers to other stations in the network is known.
8. Any retransmission due to transmission error is approximated by increasing the number of transmissions by the number of transmission errors. The user supplies the percent of total transmissions that is transmission error. The values of minimum, mean and maximum of message arrival rates for each station is increased by this percentage. Users may also model transmission error more realistically as part of message transmission sequences between stations.

CHAPTER 4

CONSTRUCTION OF SIMULATION PROCEDURE

4.1 Overview of ILAN Simulation Procedure

This chapter expands on the architecture of Block D of Figure 3-1 that was described in Chapter 3. Some of the aspects of the ILAN simulation procedure such as the initialization process, the entities and attributes as well as events are explained. One of the purposes of ILAN is to develop an efficient MAP simulation program. Details of the fast and efficient token passing simulation codes are described at the end of this chapter. Another purpose of including a simulation procedure in ILAN is to gain the ability to model sequences of messages accurately. The ways in which messages are related to entities and attributes in ILAN are described in this chapter.

4.2 ILAN Simulation Programs

A general descriptive flowchart illustrating the complete ILAN simulation program, consisting of user-written subprograms and functions, is shown in Figure 4-1. Paired arrows at a given block represent calls to and returns from other functions or subprograms. Moving counter-clockwise, the first of any two paired arrows represents the direction of the call for a given block. For example, the first arrow in counter-clockwise direction, for the block labeled "Subroutine FILEND" at the lower right hand corner of Figure 4-1, points to itself. Thus, the call is made from subroutine BATCH to subroutine FILEND. The second arrow in counter-clockwise direction from the

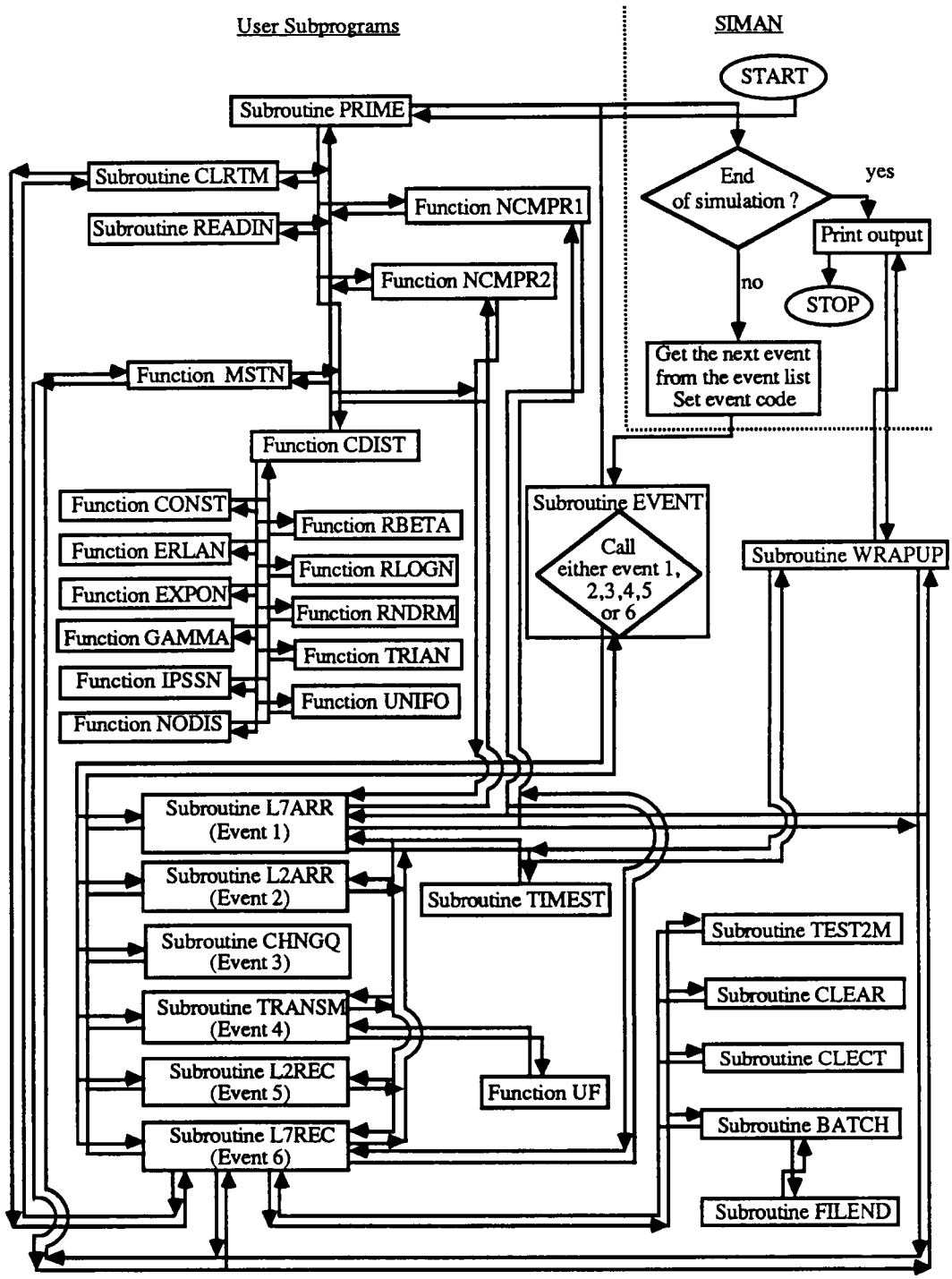


Figure 4-1. Simulation Flowchart for ILAN

block labeled "Subroutine FILEND" represents the return from subroutine FILEND back to subroutine BATCH after the execution in subroutine FILEND has been completed. Description of common variables can be found in Appendix A.

4.3 Program Initialization

At the start of the program, subroutine PRIME is executed. Subroutine PRIME calls subroutine READIN which reads input parameters for the simulation program from the input files STAT.TXT, MESS.TXT, NETW.TXT and OTHE.TXT. Information about every station on the network is stored in file STAT.TXT. The file MESS.TXT contains facts related to every message type. Details concerning sub-networks are stored in NETW.TXT file. The rest of the input parameters are stored in OTHE.TXT. These ASCII data input files are produced by the graphical user interface part of ILAN written in AutoLISP running under AutoCAD. Details of these files as well as their formats can be found in Chapter 5. These input parameters, read from the input files, are used in calculating values for other parameters as necessary. Those parameter values that are needed later in the other subroutines are stored in variables that are declared in the user common blocks.

After reading parameters from the input file, control is returned to subroutine PRIME. The main attributes for the first external message arrival at each station are determined. These attributes consist of the message type and the number remaining to be spawned. Other attributes can be determined after the message arrives at the station. At least one node on the network must have an external message arrival; otherwise there is nothing for the simulation program to simulate. The message type for each station is

calculated by function NCMPR2 from the variable PAPPL(I,K), which is a cumulative probability distribution that station K is using message type I. The number remaining to spawn new messages for this message is calculated by function NCMPR1 from the variable PSPAW(I,K), which is the cumulative probability of producing I-1 spawns from message K. These arrivals of external messages are scheduled for each station according to their time between arrivals. Function CDIST determines the time between message arrivals at each station. The statistical distribution of time between message arrivals function is called by function CDIST with the appropriate parameter set for each station. The token arrivals are then scheduled for each network. Subroutine PRIME initializes user variables to zero and calls subroutine CLRTM to clear all users' statistical variables specified in subroutine TIMEST.

After completing the execution of subroutine PRIME, control is returned to SIMAN. SIMAN determines whether the end of the simulation is reached. If the current simulation time TNOW is greater than or equal to the maximum simulation time specified by a user in SIMAN's experimental frame, or if there is no more event in the event list, then the end of simulation is reached. SIMAN outputs the results of the simulation and calls subroutine WRAPUP. The user's output is printed from subroutine WRAPUP. In addition, subroutine WRAPUP calls subroutine TIMEST to output queue statistics. Otherwise, the next event code is read and the subroutine EVENT is called with the appropriate event code as its parameter.

4.4 Entities and Attributes

Entities are objects that may change their characteristics within the boundaries of a system [81]. There are three types of entities in the ILAN simulation system. These entity types are token, message and frame. Different entity types have different attributes associated with them. These attributes describe the characteristics of the entity.

Only two attributes are needed to describe a token entity in a multi-subnetwork environment. These attributes are the sub-network number and the station number on which the token is located. Since there are multiple tokens in a multi-sub-network system (one token for each sub-network), the sub-network number attribute for the token distinguishes one token from another. The station number attribute locates where the token resides so that the next station to receive the token can be determined.

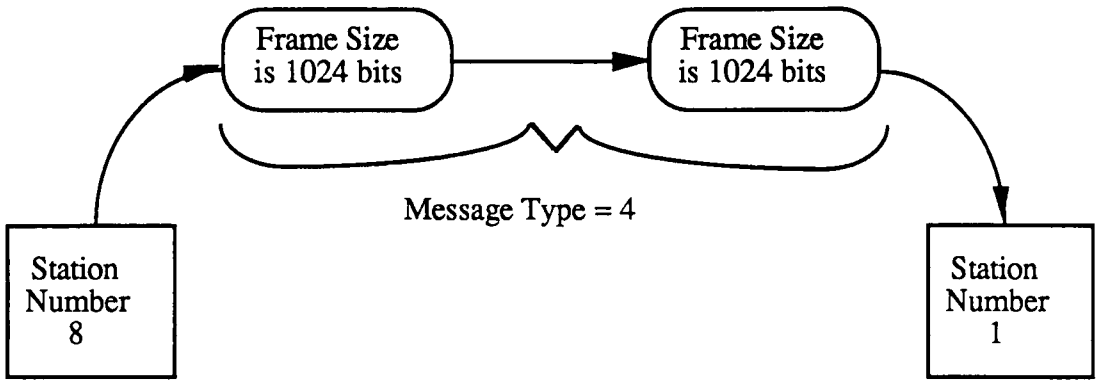
Several attributes are needed to describe a message entity. These attributes include start time, origination station number, destination station number, message I.D., message type, group message type, number of times remaining to spawn new messages, association status number, sub-network destination number and message creation type. The start time is the time that the "transmit command" is issued at the highest layer in the origination station. The origination station number initiates the transmit command to instigate the message. The destination station number specifies where the message is to be received. The message I.D. is set to 1 to identify that the entity is a message and not a token or a frame. Messages may be of different types and have different sets of allowable characteristics. These sets include frame sizes, number of multiple frames within the message and destination stations.

Figure 4-2 shows an example of the relationships between a message type and some of its characteristics, which are frame size, number of multiple frames, destination and origination station numbers. Note that the values for frame sizes, number of frames and destination station numbers are probabilistic for each message type. Thus, the frame size of 1024, number of frames of 2 and destination station number of 8 for message type 4 in this particular instance may be changed for other instances. Also, the characteristics listed for the message type in Figure 4-2 are not exhaustive. Other characteristics of message types are described below.

An association status may be assigned to a message. Table 4-1 lists the meanings of association status numbers. The sequences of messages may be attached to message groups to permit reuse in different origination and destination stations. However, within a given message group, messages can only be transmitted back and forth between the destination and the origination stations. A message group is a special message type that may have a sequence of other messages attached to it. The message type of the message group is transmitted after the sequence of messages attached to it if its association status is not 7. If its association status is 7, then it is transmitted before the sequence of message types attached to it. Through examples, the relationships between message type, message group and association status for ILAN simulation procedure are show below.

Example 4-1

For illustration purposes, suppose that it takes only two messages (message type numbers 10 and 11) to establish an association. Suppose that an origination



Characteristics of Message Type 4

Frame Size = 1024 bits
Number of Frames = 2
Destination Station Number = 1
Origination Station Number = 8

Figure 4-2. Example of Some Message Type Characteristics

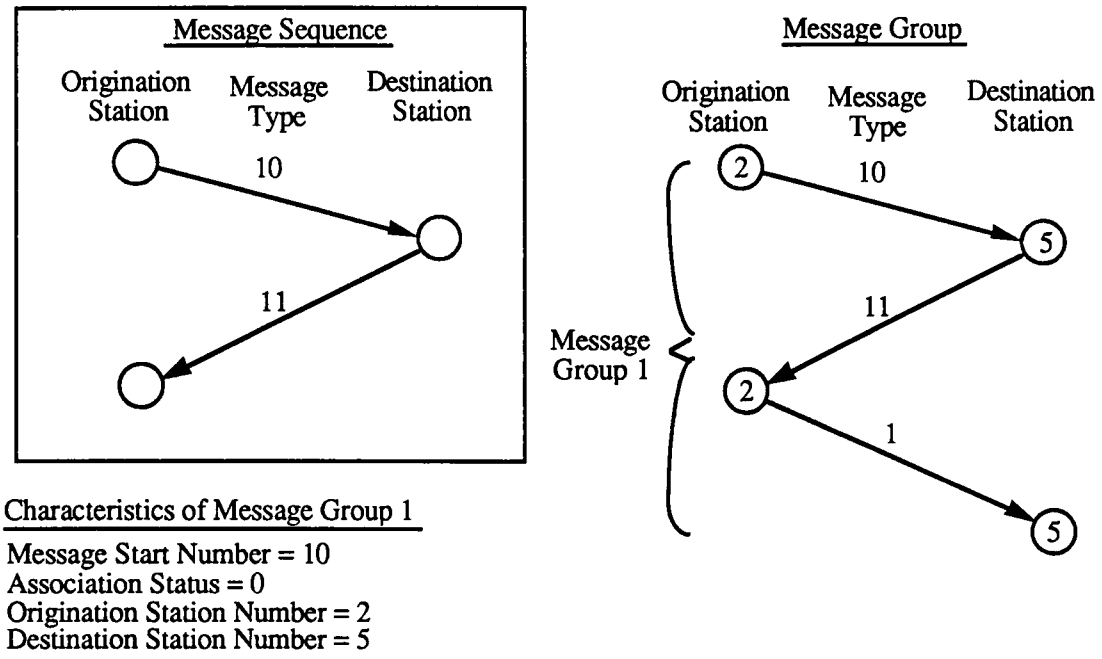
Table 4-1. Association Status

Association Status	Meaning
1	Request Association Connection
2	Request Association Connection Granted
3	Request Association Connection Denied
4	Request Association Disconnection
5	End of Association Connection Establishment
6	End of Association Disconnection
7	Message Group that sends data before requesting Association Disconnection

station wants to establish an association and transmits a message type 1. Figure 4-3 shows the relationships between the message group, the message sequence and the association status of the message group needed for this example. Here, message group 1 has message type 10 as the start message type number, association status of 0, origination station number of 2 and destination number of 5. The transmissions of message type 11 is the response of the reception of message type 10. The message sequence transmissions are transmitted before the message type 1 is transmitted because the message association status for the message group 1 is not 7. Thus, declaring that message group 1 is to be transmitted requires that the sequence of transmissions for message types 10 and 11 be completed (establishing associations between the two stations) before message type 1 is transmitted. The origination and destination stations used are determined from characteristics of message type 1.

Example 4-2

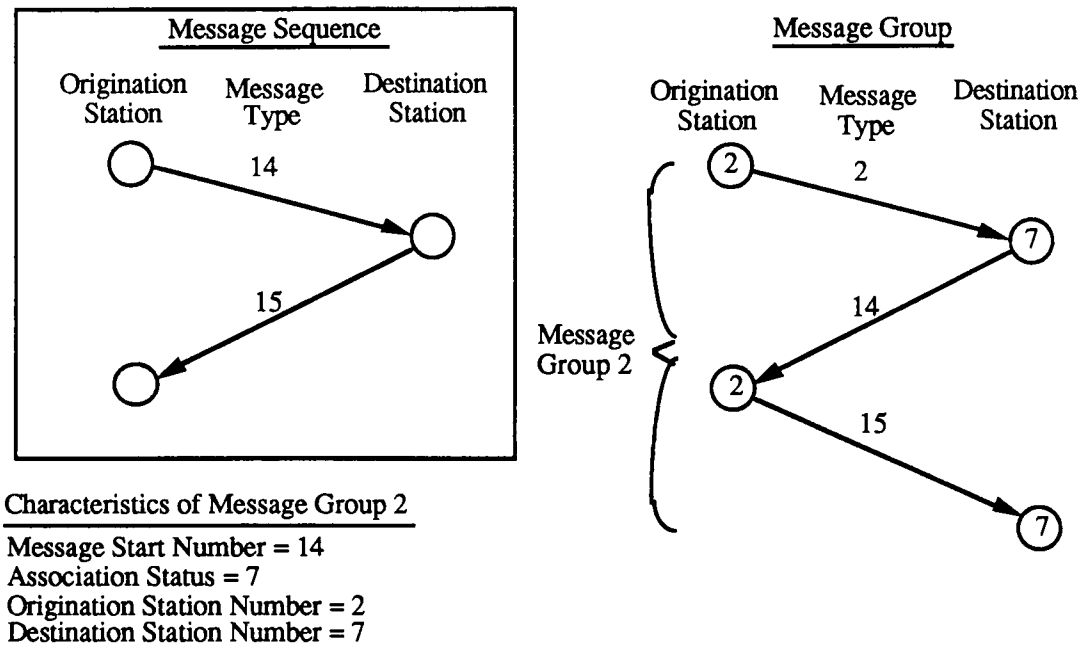
For illustration purposes, suppose that it takes only two messages (message types 14 and 15) to disconnect an association. Suppose that an origination station wants to disconnect an association after transmitting a message type 2. Figure 4-4 shows the relationships between the message group, the message sequence and the association status of the message group that is needed for this example. Here, message group 2 has message type 14 as the start message type number, association status of 7, origination station number of 2 and destination number of 7. The transmission of message type 15 is the response of the reception of message type 14. The message sequence transmissions are transmitted after the message type 2 is transmitted because



Characteristics of Message Group 1

Message Start Number = 10
Association Status = 0
Origination Station Number = 2
Destination Station Number = 5

Figure 4-3. Message Group for Association Status 0



Characteristics of Message Group 2

Message Start Number = 14
Association Status = 7
Origination Station Number = 2
Destination Station Number = 7

Figure 4-4. Message Group for Association Status 7

the message association number for the message group 2 is 7. Thus, declaring that message group 2 is to be transmitted means that the sequence of transmissions for messages types 14 and 15 are transmitted after message type 2 is transmitted. The origination and destination stations used are determined from characteristics of message type 2.

A message may be either created externally with time between creations specified by some statistical distribution or created internally by spawning from another message. Each message created externally is assigned a total maximum number of internal messages that may be spawned from it, and the number of times to spawn internal messages is limited to 24. Note that if the message spawned is a message group, then all of the messages transmitted as the result of transmitting the message group is counted as one spawn. The characteristics of spawning new messages, externally created messages, and internally created messages are demonstrated in Figure 4-5. Message group 1 of Figure 4-5 has the same characteristics as that of Figure 4-3. Message type 4 is externally created with the number remaining to be spawned of 2. After transmitting message type 4 and internally creating message group 1 by spawning, the number remaining to be spawned is 1. After transmitting message group 1, which contains message sequence types 10, 11 and 1, message type 12 is internally created by spawning, and the number remaining to be spawned is zero.

When the destination station is on multiple sub-networks, the destination sub-network is selected randomly among the available destination sub-networks that the destination station is on. The message creation type identification number equals one if the message is externally generated. Otherwise, it is zero.

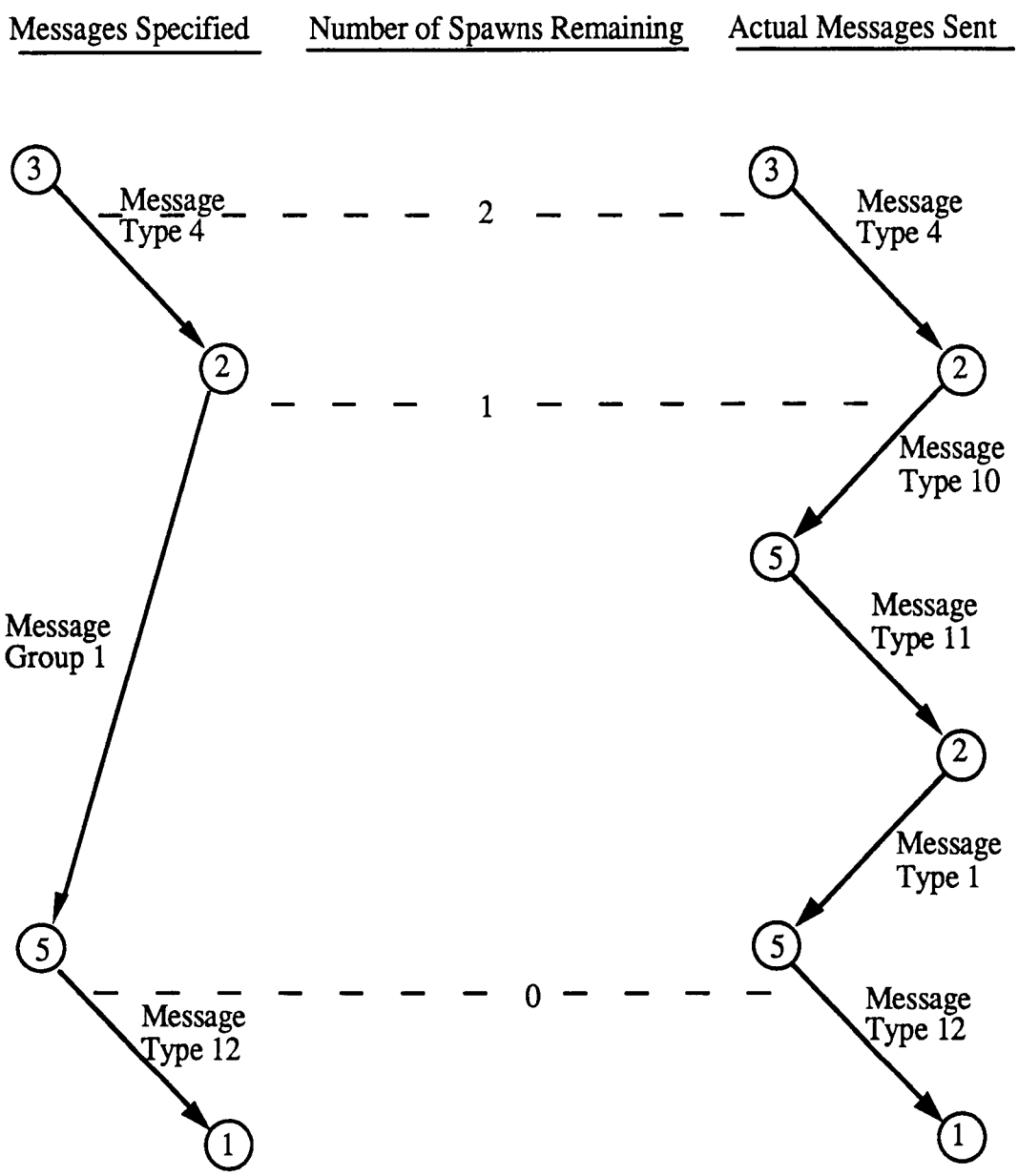


Figure 4-5. Message Spawning

Attributes similar to those used to describe message entities are used to represent frame entities. These attributes are start time, origination station number, message number, group message number, network destination number, number to spawn, and association status. Attributes used to describe a frame include last frame status, frame size type and intermediate station number. Since a message may be split into multiple frames, a last frame status that has a value of one represents the last frame for the message transmission. The frame size type identifies the size of the frame. When the destination and origination stations are not on the same sub-network, the frame must be transmitted to an intermediate node (a bridge) which, essentially, is a node that resides on one or more sub-networks.

For SIMAN running under IBM PC DOS, a maximum of 64 K bytes is available for data space in the experimental frame. Memory allocation in the experimental frame includes the maximum number of attributes for all the entities in the system, the maximum number of queues in the system and the parameters used to define statistical distributions. The frame entity requires the largest number of attributes (twelve). Thus, twelve attributes must be allocated to each entity. At medium to high load it is possible to have an average of twenty or more frames waiting to be transmitted at each station. Thus, 2000 entities in the system are not uncommon for a 100-station network. If the maximum number of entities in the system is 2000, memory allocation of 24000 variables is needed for the maximum number of attributes for all entities in the system. Since a single precision real variable occupies 4 bytes, a total of 96 K bytes is needed. The 64 K bytes of available space is exceeded.

To overcome the severe limitation of attribute space allocations, an attribute coding scheme is developed. Multiple attributes are coded and stored into a single precision real variable. Four single precision real variables are used to store the needed twelve attributes. Now, only 32 K instead of 96 K is needed. Table 4-2 shows the coding scheme used. Since only four attributes are needed for token entities, no coding is required to store these attributes. The asterisk sign (*) followed by a constant in the factors column in Table 4-2 represents a multiplication of the attribute in the same row by the constant. The plus sign (+) indicates the addition of the attributes in the row to the attribute in the next row. If the next row contains an asterisk sign (*) and a constant, the attribute of the next row is multiplied by the constant before adding. For example, message entity attribute variable four equals the number of messages left to spawn + association status number * 25 + sub-network destination number * 250 + message creation type * 6250.

A further limitation of 640 K bytes of addressable memory in DOS posts some restrictions on the size of the system that can be simulated. Larger models can be simulated with minor changes under IBM's OS/2. These changes involve changing the array of variables in the user common blocks and the file number of the event file in the user-written subroutines. The file number for the event file is the maximum number of queues specified in the experimental frame plus 1. For the same attribute coding, the restriction on the size of network simulation models running under DOS and OS/2 are tabulated on Table 4-3.

Table 4-2. Attribute Coding

Entities	Attribute Variables	Attribute Names	Factors
Token	1	Sub-network number	
	2	Station number	
	3	Time before last increment	
	4	Number of stations the token is scheduled to move through.	
Message	1	Start time	
	2	Origination station number Destination station number Message I.D.	+ * 1000. + * 2000000.
	3	Message type Group message type	+ * 1000.
	4	Number of time to spawn Association status number Sub-network destination no. Message creation type	+ * 25. + * 250. + * 6250.
Frame	1	Start time	+
	2	Origination station number Destination station number	* 1000. + * 1000000.
	3	Message type Group message type Frame size type	+ * 1000. + * 250000.
	4	Intermediate station number Sub-network destination no. Number of time to spawn Association status number	+ * 1000. + * 25000. + * 625000.

Table 4-3. Restrictions Under Different Operating Systems

Maximum number of	DOS	OS/2
Stations	100	999
Frame sizes	39	39
Applications per station	39	999
Message groups	39	999
Number of spawns per application	24	24
Sub-networks	24	24

4.5 Events

The system behavior is simulated by a series of events. "Events usually cause changes in the status of the system ..." [81]. Events represent either the start or the end of activities. Activities are tasks that usually require time to complete. Pegden [79] describes the relationship between events and activities as follows:

> Activities that occur over time must be modeled using two events. The first event models the state changes that occur at the start of the activity, and then schedules the second event to occur at the end of the activity. The second event models the state changes that occur at the end of the activity.

Further explanations about events in simulation can be found in simulation text books such as GASPIV [81] or SIMAN [79].

TRANSM, L2REC and L7REC, representing events 1 to 6, respectively. Each of these events is briefly discussed below. The simulation flow chart in Figure 4-1 shows the relationships of these subroutines to the rest of the functions and subroutines in the ILAN simulation model. More detailed discussions and flow charts of these event subroutines can be found in Appendix B.

Event L7ARR is the arrival of a message or frame at the station in the highest layer of the protocol stack. The message is split into frames. The attributes for each frame are determined. If the transmitting protocol stack of the station is not busy, then the frame is scheduled to arrive at layer 2 of the same station. The arrival time at layer 2 is determined from the stack delay for transmitting the frame belonging to a given message type from a particular station. The transmitting stack queue is set to busy to indicate that it is processing the frame. Otherwise, the frame is inserted in the waiting queue of the transmitting stack.

The arrival of frame in layer 2 at the station is called L2ARR. Since a frame has just arrived at layer 2, the transmitting stack is no longer busy. Therefore, the waiting queue for the stack is set to not busy. The frame is inserted in the appropriate queue at this station, waiting for transmission to its destination station.

The token passing event is represented by event CHNGQ. The sub-network of the token that arrives in this event is determined. The token is scheduled, with the appropriate amount of time delay, to arrive at the next station that has something to transmit. If every station of the sub-network has nothing to transmit, then it is assumed that the token is passed from station to station on the sub-network until there is some data to be transmitted. The time that the sub-network has a message to be transmitted is

that the token is passed from station to station on the sub-network until there is some data to be transmitted. The time that the sub-network has a message to be transmitted is not known but can be approximated. The number of stations that the token visited during this idle period is calculated. From it, the station at which the token resides just before there is a message to be transmitted is determined. The token is scheduled to arrive at this station at that time.

TRANSM event represents the "try to transmit frames" event. If the station has a frame to be transmitted and the time that the token has spent in this station has not exceeded the token holding time, then the frame is transmitted by scheduling it to arrive at layer 2 of the destination station. If the station has more frames to be transmitted, they are transmitted until there is nothing to transmit or until the token holding time is exceeded. At the end of transmission, the token is passed to the next station by scheduling the CHNGQ event at the current time, TNOW.

The reception of a frame at layer 2 is event L2REC. If the receiving stack of the station is busy then the frame is placed in the waiting queue for the stack. Otherwise, the frame is scheduled to be received in layer 7 of the same station and waiting queue for the stack is set to busy.

Event L7REC is the reception of a frame at layer 7. Processing of the frame through the reception stack has just been completed. Thus the reception stack is no longer busy. The reception stack is set to not busy. If this station is a bridge, then this frame is to be transmitted from this station to its final destination station. This is accomplished by scheduling the L7ARR event for the frame. If the frame should

spawn another message, then new attributes for the newly spawned message are set up, and the L7ARR event is scheduled for the new message.

4.6 Fast and Efficient Token Simulation Codes

The load on a MAP network is usually very light. When there is little or no traffic, most of the time is spent in passing the token from one station to another on the network. Normally the passing of the token is accomplished by scheduling the token to arrive at the next node on the network. When the token is received at the node, the messages waiting to be transmitted at the node are transmitted until the token holding time limit runs out or there is nothing to transmit. The token is then passed to the next node on the logical ring.

In order to efficiently simulate the token passing for light traffic situations, ILAN looks ahead at the next sequence of nodes in the logical ring to see if they have anything to transmit before passing the token. The token is scheduled to the next node in the same sub-network that has something to transmit. If all nodes on the sub-network have nothing to transmit, the token is moved as close to the time that there is some data to be transmitted as possible. This is accomplished by determining the time of the next occurrence for event L7ARR, L2ARR or L7REC and calculating the location of the token at this time. The token is scheduled to arrive at this node at the same time as the next L7ARR, L2ARR or L7REC arrival. An easier scheme of passing the token at every event's occurrence is not much better than passing the token to each and every node, especially in multiple sub-networks environment. The reasons for selecting events number 1, 2 and 6 are as follows:

2. In event L2ARR, a frame waits to be transmitted at a given node. When the token arrives at its scheduled node there is at least one node that has something to transmit.
3. Event L7REC may represent a frame arrival at a bridge node that belongs to the sub-network on which the token resides. If the frame is forwarded to the same sub-network, then event L7ARR and consequently event L2ARR are scheduled.

However, in some cases, the scheduled token may overshoot the arrival of messages. For example, suppose that all nodes on the sub-network under investigation have nothing to transmit. The event list contains event TRANSM, scheduled to arrive at time 200, and event L2ARR, scheduled to arrive at time 1000 for this sub-network. If the token is passed between nodes that have nothing to transmit, then the location of the token can be determined at time 1000. Suppose that the token is calculated to be at node number 10 at time 1000. The token is scheduled to arrive at node number 10 at time 1000. Suppose that at time 200 a frame is transmitted (event TRANSM) and that the frame arrives at layer 7 of the destination node at time 210 and it spawns another message. If the spawned message arrives at layer 2 of the same sub-network at time 222, then the token was overshot considerably. A correction scheme needs to be included.

The adjustment is incorporated in event L7ARR. This is accomplished by the code listed in Figure 4-6. When a frame arrives at layer 7 of a given sub-network, the

```
      AQL = 0.0
      DO 400 I = 1, NSTA
        AQL = NQ (I) + AQL
400   CONTINUE
      AQL = AQL / NSTA
      IF (AQL .LT. 2.0) THEN
        KNXT = LFR(301)
        NCNT = 0
410     CONTINUE
        IF (NEVENT(KNXT) .EQ. 3 .OR. NEVENT(KNXT) .EQ. 4) THEN
          DO 420 K = 1, NETOL
            RK = K
            IF (A(KNXT,1) .EQ. RK) GOTO 430
420       CONTINUE
          GOTO 440
430       CONTINUE
          NCNT = NCNT + 1
          NNET = A(KNXT,1)
          IF (TEVENT(KNXT) - TNOW .GT. OVH(NNET) + NET(NNET)
            .AND. A(KNXT,4) .GT. 1.) THEN
            CALL REMOVE(KNXT,2)
            NNST = A(KNXT,2)
            NLOC = NQLOC + AMOD(A(KNXT,4),NET(NNET))
            NORI = MOD(NORI,NET(NNET))
            OVEQ = (TNOW - A(KNXT,3)) / OVH(NNET)
            TMVE = OVEQ
            OVEQ = AMOD(OVEQ, NET(NNET))
            NDES = NORI - OVEQ
            NDES = MOD(NDES,NET(NNET))
            IF (NDES .LE. 0) NDES = NET(NNET) + NDES
            NGLO = NQGLO(NNET,NDES)
            DES = NGLO
            CALL SETA (KNXT,2,DEST)
            CALL SETA (KNXT,4,TMVE)
            CALL SCHED(KNXT,3,0.)
          END IF
440     CONTINUE
        IF (KNXT .NE. LLR(301) .OR. NCNT .LT. NETOL) THEN
          KNXT = LSUCC(KNXT)
          GOTO 410
        END IF
```

Figure 4-6. Token Re-scheduling Codes

token for the same sub-network in the event list is found. If the token is scheduled to move more than one full rotation without transmitting any messages, then its current position is determined and it is re-scheduled to arrive there at this time.

In the first five lines of Figure 4-6, the current average queue length (AQL) is determine by totalling the queue lengths and dividing the total by the number of queues on the sub-network. The rest of the codes are executed only if the average queue length is less than 2. In medium to heavy traffic (average queue length greater than 2) there is no need to re-schedule that token since it is not likely to skip many nodes. To try to re-schedule the token in heavy traffic situations would slow the program down considerably.

KNXT is set to be the first entity in the event list. If this is event CHNGQ or event TRANSM, then the program determines whether the event is a token. It is a token if the value of its first attribute is a sub-network number. NCNT is set to be the counter for the number of tokens found so far. NNET is set to the value of the sub-network number for the token. If the token moves more than one full rotation without transmitting messages, then the node that has the token at this time is determined and the token is scheduled to arrive there. If KNXT is not the last entity on the event list and not all of the tokens have been found, then KNXT is set to the next entity on the event list and looped back to determine whether it is event CHNGQ or TRANSM.

CHAPTER 5

CONSTRUCTION OF USER GRAPHICAL INTERFACE

5.1 Overview

The purpose of this chapter is to describe in detail Block B of Figure 3-1. AutoCAD was selected instead of other graphical environments for the following reasons:

1. It is desirable to use an AI language such as LISP to manipulate non-numeric information and reduce the development time of prototype systems. AutoCAD provides a general LISP-based programming language called AutoLISP which is easy to learn and use.
2. AutoCAD provides simple ways of customizing its menu thus eliminating the tasks of writing programs to implement and manipulate them. A customized menu within AutoCAD is written such that the AutoLISP procedures can be easily selected and executed using a mouse as a pointing device.
3. AutoCAD provides a command line interface which is an alternate way of accessing AutoLISP functions. These AutoLISP functions can be executed by selecting them from the menu or by typing the name of the function from the AutoCAD's command line. Experienced users may be able to execute some AutoLISP functions faster through the command line without having to go through several levels of the menu.

4. All AutoCAD drawing functions are available through the command line. Experienced AutoCAD users can manipulate the drawing using familiar functions.
5. Drawings in AutoCAD are automatically converted into drawing databases that can be manipulated through the AutoLISP program. The desired information from the database can be easily extracted to other programs such as the ILAN simulation procedure.
6. Drawing entities and their attributes in AutoCAD are well suited for defining simulation problems graphically. Drawing entities such as a bridge or a station can be pre-constructed with their associated attributes defined and stored as blocks in AutoCAD. These blocks can be inserted anywhere in other drawings at latter time. The values of attributes for each block are specified at the time of the insertion. For example, attributes for a station block include the station number, the sub-network number that the station is on, etc.
7. AutoCAD drawing functions are easy to use. Unlike drawing functions available in other languages such as C or FORTRAN, there is no need to set up parameters and pointers before each execution. AutoLISP is embedded within the AutoCAD program. Through AutoLISP, users can access and update AutoCAD's drawing entity data, manipulate AutoCAD's variables, execute AutoCAD's commands and control AutoCAD's graphics screen and device input [91].

5.1.1 Menu

Menu items are enclosed within square brackets; [DRAW] represents a menu item. Only these menu items are displayed on the menu on the right part of the AutoCAD screen. When the user selects a menu item, the commands that follow the menu item are executed. These commands may contain menu codes and macros.

Figure 5-1 shows the root menu of the customized AutoCAD menu for ILAN. The meaning and use of each menu item is discussed in more detail later in Chapter 6. The customized AutoCAD menu listing file can be found in Appendix D. Further explanations of the AutoCAD menu and macro can be found in *INSIDE AutoCAD* [82] and *CUSTOMIZING AutoCAD* [90].

5.1.2 Drawing Data Base

AutoCAD automatically keeps track of information about its drawing in its database. AutoLISP can access and manipulate data in the database. If a line is drawn from coordinate (1,1) to coordinate (10,10) using AutoCAD's drawing editor, then the information stored in the database for the drawing looks like this:

((-1 . <Entity name : 60000014>) (0 . "LINE") (8 . "0") (10 1.000000 1.000000) (11 10.000000 10.000000))

The data is stored in the association list format. For AutoCAD's database, the value of -1 is associated with the entity name. The entity name is a unique record location of the information stored. The value of 0 is associated with the entity type. For the above

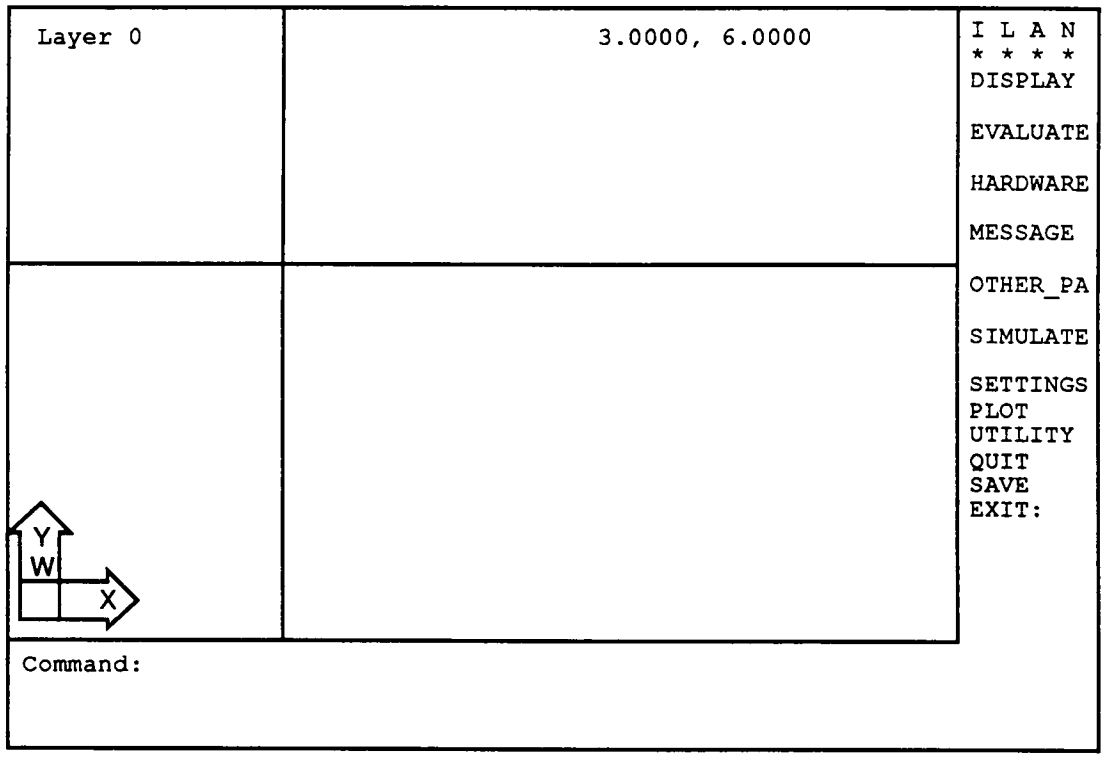


Figure 5-1. ILAN Root Menu

example, the entity type is a line and the value of 8 is associated with layer name. In this case, the layer name is "0." The value of 10 is associated with the X and Y coordinates of the starting point of the line and the value of 11 is associated with the X and Y coordinate of the end point of the line. The length of a record is a variable depending on the type of its entity.

5.1.3 Blocks and Attributes

AutoCAD allows strings of characters to be stored in a drawing as attributes. Attributes can be defined using AutoCAD's ATTDEF command. An attribute can be declared to be invisible. Normally, invisible attributes are not drawn in the drawing. However, these invisible attributes can be displayed in the drawing by executing AutoCAD's attribute display command, called ATTDISP. Through the ATTDISP command, normally visible attributes can be made invisible and vice versa. Constant values can be assigned to attributes at the time of the attribute definition. Several attributes can be grouped together and attached to other drawings to form a block. A block can be stored in a separate drawing file and inserted into other drawings. AutoLISP user-written functions can interact with the users for values of attributes before the block that contains these attributes is inserted into the drawing. More information concerning AutoCAD's attributes and blocks can be found in *AutoCAD Database Book* [42].

All input to ILAN is stored in the AutoCAD drawing database as attributes. All objects in the drawing such as the stations, the bridges, and the headend remodulators are blocks that had been pre-constructed and stored in other files. When the user issues

a command to draw an object, either by typing the procedure name on the command line or by selecting the command from the menu that executes that procedure, the procedure asks the user for values of attributes of the object. After all of the values of the attributes are determined, the procedure inserts the object and its known attributes values in the drawing.

A report of the values of the attributes in the drawing can be generated using the attribute extraction command, called ATTEXT. The user must supply the name of the template file and the name of the extracted attributes file. The ATTEXT command extracts attributes in the drawing that fit the format given in the template file and writes them in the extracted attributes file using the format contained in the template file. Table 5-1 contains four template files. These files are STATIONS.TXT, NETWORK.TXT, MESSAGE.TXT and OTHERPA.TXT. The corresponding extracted attributes files that use these template files are STAT.TXT, NETW.TXT, MESS.TXT and OTHE.TXT, respectively.

Each template file contains the names of the attributes and their corresponding formats. The first character of the format string is either "N" or "C," indicating a numerical or a character output. The total output field width is specified in the next three digits that follow the first character field. The last three digits specify the number of decimal places for the output. (000 is specified for character output.) Thus "N008001" is equivalent to "F8.1" in a FORTRAN format statement, corresponding to the total field width of 8 characters with 1 decimal place. Integer format can be specified using numerical output with the last three digits of "000."

Table 5-1. Template Files

STATION.TXT		NETWORK.TXT		MESSAGE.TXT		OTHERPA.TXT	
Attribute	Format	Attribute	Format	Attribute	Format	Attribute	Format
PSENDMS	C100000	NTYPE	C002000	MESSAGE	C003000	FSIZE	C100000
PARAMTR	C100000	DRATE	C010000	PNXTMES	C100000	PERAC	C010000
SDIST	C006000	THT	C010000	PDEST	C100000	PERCO	C010000
MAXAS	C003000	NETN	C010000	START_M	C004000	PERRO	C010000
BRDG	C008000			ASSOC_S	C002000	DASA	C010000
STATION	C004000			PFRMSZ	C100000	PREA	C010000
D27	C100000			PFRAMES	C100000	MINF	C010000
D72	C100000			PSPAWN	C100000	T&RMOD	C020000
QDIS	C002000			APDELAY	C100000	HED	C010000
DIST	C100000					MORF	C002000
NET	C100000						

Before a simulation program can be executed, four files have to be generated. These files contain values of attributes from the drawing which are used as input parameters for the simulation program. The file named STAT.TXT contains information concerning stations and bridges. The information about each sub-network is stored in the file named NETW.TXT. Attribute values of messages are extracted into the MESS.TXT file. Any other parameters are extracted into a file called OTHE.TXT.

5.2 AutoLISP Procedures

This section briefly describes the user-written AutoLISP procedures of ILAN. Figure 5-2 shows the relationships between these procedures. Arrow lines represent calls from one procedure to another in the direction of the arrow. The flowcharts and further explanations of each of these procedures can be found in Appendix C.

At the beginning of each AutoCAD drawing session, AutoLISP functions in the ACAD.LSP file are loaded into the AutoCAD environment. As soon as the file is loaded, AutoCAD executes procedure S::STARTUP if that procedure exists in ACAD.LSP file. For ILAN, the start up procedure executes only two functions. The first invokes another function called LOCATI. The second unloads LOCATI after it had been executed.

For large problems, the size of the drawing database can be enormous. In order to avoid unnecessary delays at each command that searches the database for some specific object, LOCATI procedure compiles an index that contains lists of important objects from the database. LOCATI procedure reads every record in the database and

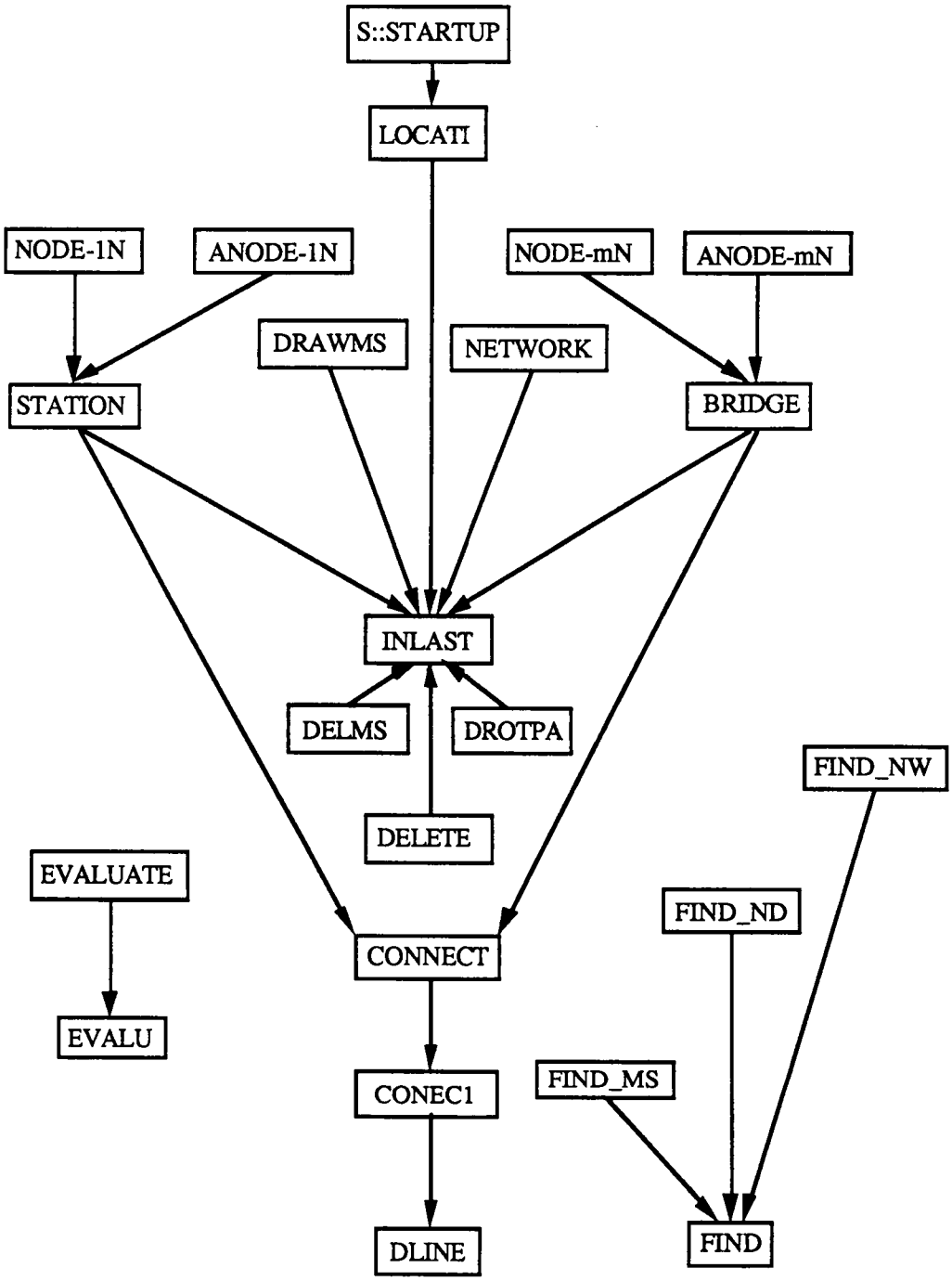


Figure 5-2. Relationships of ILAN Procedures

files locations of the object and its number in the object lists. Table 5-2 shows the names of object lists, the object types and the object numbers that are stored. For example, the list STLIST contains the names of all stations and bridges in the drawing, their corresponding station numbers and their record locations. Suppose that information about station number 2 is to be found by ILAN. First, the record location of station number 2 is found from STLIST. With the record location of station number 2, the information can be read from the database immediately. Searching through the object list is much faster than searching through the database because the object list is much smaller. Only the essential items necessary for locating a particular object are contained in an object list, while the database contains information about every object.

NODE-1N and ANODE-1N procedures can be selected from the menu or executed on the command line. NODE-1N procedure calls STATION procedure with a NIL parameter to insert a station block into the drawing. ANODE-1N procedure calls

Table 5-2. Object Lists

NAME OF OBJECT LIST	NAME OF OBJECT TYPE	OBJECT NUMBER
STLIST	Station and Bridge	Station Number
NWLIST	Network	Network Number
HELIST	Headend Remodulator	Network Number
PLLIST	Polyline	Insertion Point
OTLIST	Other Parameters	1

STATION procedure with an "ADD" to insert a station and to connect the station to a specific existing sub-network that had been defined on the drawing. Both procedures ask the user for information about the station and store the information as attributes in the station block before inserting it on the drawing. The station inserted using NODE-1N or ANODE-1N can be connected to only one sub-network. Each time a station is inserted in the drawing, INLAST procedure is called to insert the station number and its record location within the AutoCAD database in STLIST.

NODE-mN and ANODE-mN procedures can be selected from the menu or executed on the command line. NODE-mN procedure calls BRIDGE procedure with a NIL parameter to insert a bridge block to the drawing. ANODE-mN procedure calls BRIDGE procedure with an "ADD" to insert a bridge block and to connect it to specific existing sub-networks that had been defined on the drawing. The bridge block may represent either a station, bridge or filter that can be connected to multiple sub-networks. If a station is inserted, then the station can transmit messages to each sub-network that is connected to it. However, it does not automatically forward transmissions from one sub-network to the another sub-network. If a bridge is inserted, then the bridge acts as a forwarding station. Messages that need to be transmitted to another sub-network are automatically forwarded to that sub-network through the bridge. If a filter is inserted, then only the transmissions from the first sub-network to the second sub-network on the filter's sub-network list are permitted. Transmissions from the second sub-network to the first sub-network are not allowed. The user is asked for information about the station, bridge or filter and must store the information as attributes in the bridge block before inserting it on the drawing. The list

of station numbers and their record locations in the AutoCAD database (STLIST) is updated by calling the INLAST procedure.

NETWORK procedure can be executed by selecting it from the menu or by typing its name on the command line. This procedure inserts a network block in the drawing. The user is interrogated for the values of sub-network attributes. These attributes are part of the network block inserted in the drawing. After the network insertion, the INLAST procedure is called to update the list of sub-network numbers and their locations (NWLIST). The user is asked to a draw line segment or segments that represents the cable line of the sub-network. The type of line that is used to draw the sub-network cable is called polyline. INLAST procedure is called again to update the list of polylines and its record location in PLLIST. If this sub-network is broadband, the user is asked for the coordinate point at which the headend remodulator is to be inserted before it is inserted on the drawing. If the headend remodulator is inserted, INLAST is called to include the location of the new headend remodulator in HELIST.

CONNECT procedure is executed either by selecting it from the menu or by calling it from the command line. Its function is to connect every station to its specified sub-networks. Stations in this case include bridges and filters. It is assumed that these sub-networks have been defined and drawn on the drawing prior to executing the CONNECT procedure. For each station contained in the station list (STLIST), CONNECT procedure calls CONEC1 procedure. Different parameter values of the insertion point of the station and the sub-network numbers that are connected to the station are passed to the CONEC1 procedure at each call. For each sub-network

connected to the station that CONEC1 is processing, DLINE procedure is called. The insertion point of the station and the sub-network number that is to be linked to the station are passed to DLINE procedure when it is called. DLINE procedure draws a perpendicular line from the main sub-network cable line to the edge of the station.

The DRAWMS procedure can be invoked through the menu or the command line. Upon its execution, the procedure interrogates the user for the information to be transmitted. The message information is stored in attributes as part of the message block that is inserted in the drawing. As relationships between messages become known, information stored in the message blocks and drawn in the drawing are immediately updated. The immediate visual display feedback of the messages being defined aids the user in spotting errors quickly. The interactive graphical input interface helps the user define messages and correct defined message mistakes easily and quickly. Through a mouse, the user can immediately select the message node to be modified. Each time a new message is inserted, the INLAST procedure is called to update MELIST, which indexes messages in the AutoCAD database.

To delete a message, the user must call the DELMS procedure. It can be executed from the command line or from the menu. This procedure deletes the message that is selected and asks the user to select the message that spawned the deleted message. If such a message exists, the information about spawning the deleted message is erased. If the deleted message was not spawned from another message, then the user must hit the "CARRIAGE RETURN key" (CR) on the keyboard. In any case, the user is asked to select the arrow line that joined the deleted message and its parent message. If the arrow line exists and is selected, it is deleted from the drawing.

If there is no arrow line to be deleted, the user should hit the CR key. After the message is deleted, INLAST procedure is called to delete the information about the deleted message from the MELIST.

The DROTPA procedure asks the user for information about the rest of parameters that are not related to any specific station, bridge, network, or message. This information is inserted in the drawing as the other parameter block. The INLAST procedure is called to insert the record location of the other parameter block in OTLIST after the insertion.

To delete any stations, bridges, headend remodulators, other parameters, sub-network declarations or cable lines, the user must execute the DELETE procedure. The regular AutoCAD's ERASE command cannot be used because it does not update the database index lists. The DELETE procedure can be invoked either from the command line or from the menu. Upon its execution, the user is asked to select the items to be deleted. Each item selected is erased from the drawing, and INLAST procedure is called to delete it from its index list.

Generally, if any changes are made to the drawing, AutoCAD automatically updates these changes in the drawing's database. However, these changes are not updated to the database index lists automatically. Thus it is the responsibilities of the procedures that modify the drawing to also revise the database index lists. A procedure that handles the task of updating any of the index lists is needed. This procedure is to be called whenever changes are made to the drawing. Only a few lines of codes are needed for this procedure to be written in AutoLISP. The name of the procedure is INLAST. Figure 5-3 shows the listing of INLAST procedure.

```
(DEFUN INLAST (ACT OBJ NUM REC)
   (IF (EQUAL ACT "INSERT")
      (SET OBJ (CONS (LIST NUM REC) (EVAL OBJ)))
      (IF (EQUAL ACT "DELETE")
         (SET OBJ
            (APPEND
               (REVERSE (CDR (MEMBER (LIST NUM REC ) (REVERSE (EVAL OBJ)))))
               (CDR (MEMBER (LIST NUM REC) EVAL OBJ))
            )
         )
      )
   )
)
```

Figure 5-3. INLAST Listing

To invoke the INLAST procedure, four parameters must be passed to it. Table 5-3 shows the name and meaning of these parameters. The INLAST procedure tests to see if the ACT (action) is "INSERT" (see second line of Figure 5-3). If the action is "INSERT," then the list of the item number and the record location of the item inserted is concatenated to the appropriate database index list to make a new list. By evaluating the OBJ symbol, the name of the appropriate database index list is simultaneously determined and used. If the ACT is "DELETE," then the item number and the record location of the item deleted is erased from the appropriate database index list. In another conventional language such as FORTRAN, the same procedure would be

Table 5-3. Parameters for INLAST

PARAMETER NAME	MEANING
ACT	Action can be either "DELETE" or "INSERT"
OBJ	Object name of item deleted or inserted can be 'NWLIST, 'HELIST, 'STLIST, 'MELIST, 'OTLIST or PLLIST.
NUM	Item number
REC	Record location in the database that contain information about the object number.

considerably longer. Two IF statements and two statements that modify the database index list would be needed for each database index list (one for delete and the other for insert cases).

In large problems, where there are several nodes, sub-networks and messages on the drawing, it is almost impossible to locate any particular item manually. A procedure named FIND is written to aid the user in locating these items. The procedure is called from either the FIND_MS, FIND_ND or FIND_NW procedure to locate any given message, node or sub-network, respectively. The located item is then displayed on the screen.

Before simulating the defined MAP network problem, it is a good idea to call the EVALUATE procedure to check for any obvious error(s). To free up the AutoCAD environmental space, the procedure is kept in a separate file. When invoked, it loads a

procedure called EVALU from a file called EVALU.LSP. Since the checking procedure is usually executed once at the end of the editing session, it is not kept in the ACAD.LSP file. The procedure checks for duplications in sub-network numbers, station numbers and message numbers defined. The sum of percentages for launching new messages for each message is also verified to be equal to 100. The existence of sub-network definitions corresponding to sub-networks that are attached to the defined stations are checked. The existence of message definitions corresponding to messages that arrive externally at the stations, messages that are spawned from another message and messages that are transmitted first for message groups is verified. The existence of at least one external message arrival is investigated. Since the simulation program starts when a message arrives externally to the network, there must be at least one external arrival. The completeness of message definition for each message is verified. The existence of station definitions corresponding to the destination stations specified for the messages is determined. Finally, the existence of frame size definition corresponding to each frames size specified for each message is checked.

CHAPTER 6

ILLUSTRATION

6.1 Executing ILAN

In order to access the graphical interface of ILAN, the user must enter the directory where ILAN files reside and type:

ILAN filename

(where "filename" is the name of the drawing). If the drawing exists, then ILAN uses it. Otherwise, ILAN starts with a new drawing whose name is given by filename. The ACAD.LSP containing user-written AutoLISP procedures are automatically loaded into the AutoCAD environment. The ACAD.MNU containing ILAN customized menu is also loaded automatically. The graphical interface is divided into three categories, which are hardware, message and other parameters. The hardware section includes the definition of the stations, bridges, filters, headend remodulators and network cables. Information concerning the messages is defined in the message section. All other parameters are defined in the "other parameters" section.

6.2 Hardware Definition

To get into the hardware section the user must select HARDWARE from the menu. The bottom screen of Figure 6-1 shows the HARDWARE being selected from

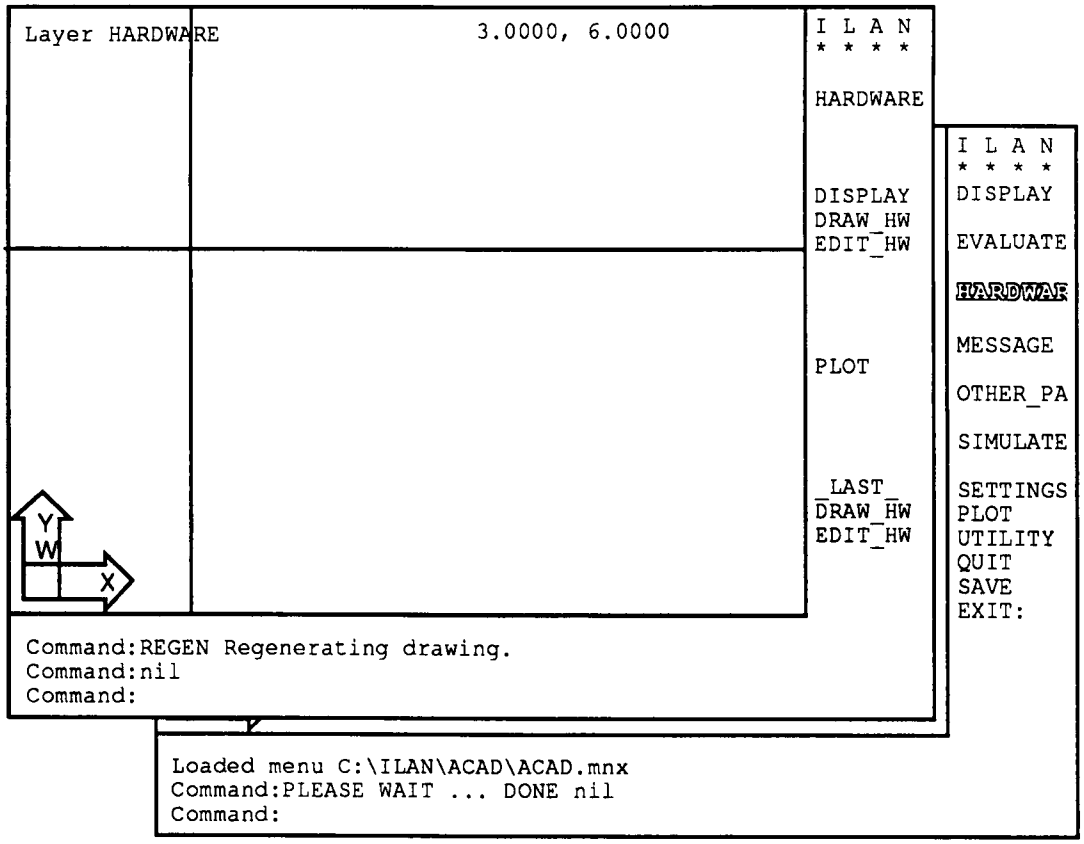


Figure 6-1. Initial Screen of Hardware

the menu. The top screen of Figure 6-1 shows the initial hardware screen. Notice that the layer name on the left side of the screen is changed to HARDWARE.

There are five options that the user could choose from the hardware menu. If the ILAN option on the first line of the menu is selected then ILAN's main menu or root menu is displayed. The DISPLAY option can be selected to switch the layer, magnify, move to a different part of the screen, find and display the item specified. To insert a new hardware item into the drawing, the DRAW_HW option is selected from the menu. The existing drawing can be edited by selecting EDIT_HW from the menu.

Figure 6-2 shows the "draw hardware" screen after the DRAW_HW has been selected. New options on the menu are NODE-1N, NODE-mN, HEADEND, NETWORK and CONNECT. These commands can also be executed via the command line. A station that is connected to a single sub-network is defined by selecting NODE-1N. NODE-mN option is used to define a station, bridge or filter that may be connected to multiple sub-networks. HEADEND is selected for drawing a headend remodulator. A sub-network is drawn by selecting NETWORK. The defined stations and their corresponding sub-networks are connected by selecting CONNECT from the menu.

When NODE-1N is invoked, the user is asked for the node number of the node being defined (Figure 6-3). If the station number is 1, then the user types "1" on the command line. The procedure accepts the answer and asks if the user would like to specify default values of stack delays associated with transmitting messages from this node (node number 1 in this case). If there are several messages transmitted to this node and all of them have the same stack delay, then the user must type "Y" (Figure 6-3).

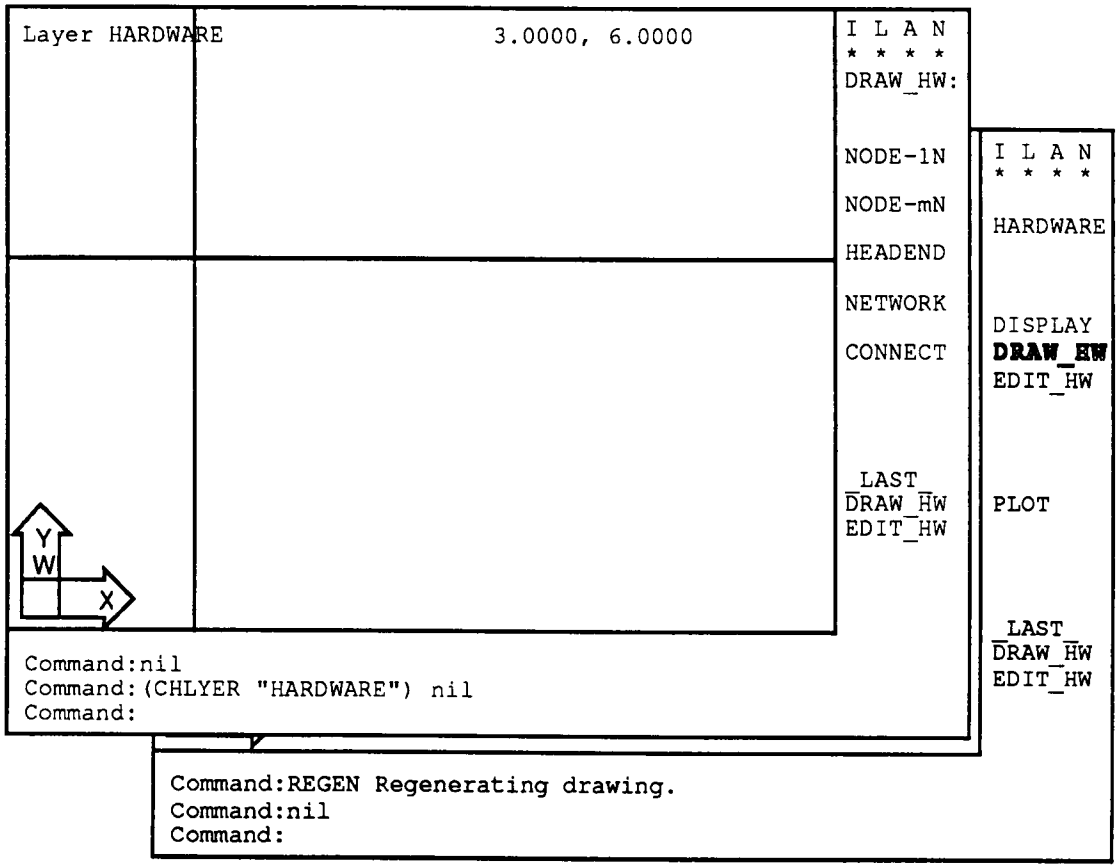


Figure 6-2. Draw Hardware Screen

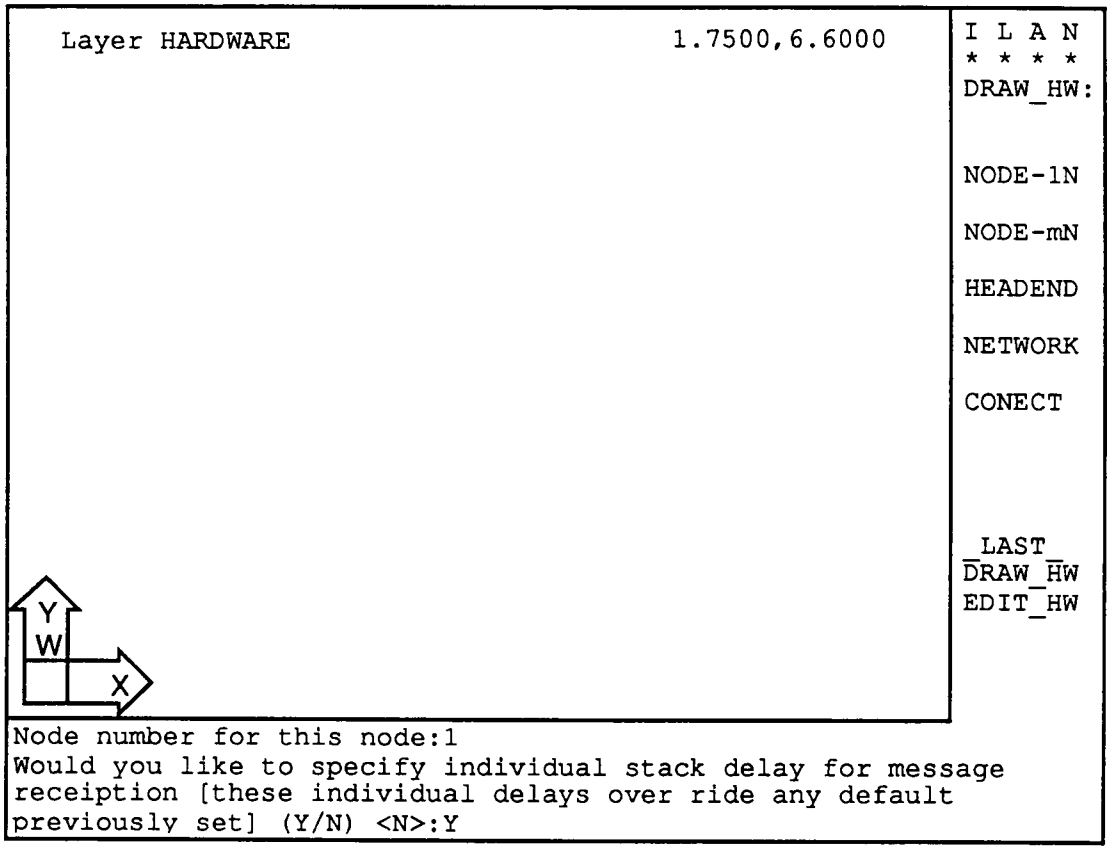


Figure 6-3. NODE-1N Screen

The user can switch to the text screen by hitting the F1 key. Figure 6-4 shows the rest of the questions and answers on the text screen. If an answer is too long to fit on the command line, the user may continue typing on the next line. Values enclosed in <> are the default values. A default value is used when no value is entered and the CR key is pressed. The user types in the answer to each question after the colon (:).

If the user wishes to specify a default value for stack delays associated with transmitting packets for any message, then the program asks for the last message number that uses this default delay for messages that are transmitted to this node. Suppose that the stack delays associated with receiving message numbers 1 through 20 are the same; then the last message number that uses this default delay is message number 20. In this example there are only 2 messages (messages number 1 and 2) that have the same delay of 36500 microseconds. Thus the user may type "2" in response to the question about the last message number that uses the default delay. The next question asks for the value of the default stack delay associated with receiving messages. Since this is different from the default value given for this question (<36500>), the user must enter "145000." If this were the same as the default value, then the user could hit the CR key to accept the default value. Note that some of the messages that are given the default stack delays associated with receiving may or may not be transmitted to this node. If they are not transmitted to this node, then their delays are not used. Thus the user may specify messages 1 through 20 to have the same default delays when only messages 12, 17 and 20 are transmitted to this node. In this example only message number 2 is transmitted to this node.

The user may also override the given default values of delays. The next question asks if the user wants to specify individual stack delays associated with

```
Enter last message no. that uses default reception stack
delays at this node [message 1 through this last message
no. uses this default delay]:2

Enter the default reception stack delay in microseconds
<36500>:

Would you like to specify individual stack delay for
message reception [these individual delays over ride any
default previously set] (Y/N) <N>:N

Would you like to specify default transmission stack delays
for messages that are transmitted from this node (Y/N)?
<N>:N

Would you like to specify individual stack delay for
message transmission [these individual delays over ride any
default previously set] (Y/N) <N>:Y

How many different individual message delays in
transmission stack are to be specified ?:2

Enter message number of a message that is transmitted from
this node:1

What is its transmission stack delay in microseconds?
<36500>: 145000
```

Figure 6-4. NODE-1N Input Text Screen

```
Enter message number of a message that is transmitted from
this node:2

What is its transmission stack delay in microseconds?
<36500>: 145000

Find distances from drawing, User input or assume 0 (D, U,
or 0):0

How many sub-networks are connected to this node :1

Enter sub-network number of a sub-network connected to this
node:1

Messages arrive externally to this station ? (Y,N):Y

Do all external messages arrive Once, Constantly or
Randomly (O/C/R) <C>:C

What is the constant time between message arrivals in mic.
sec.:1.0E7

How many different messages to be transmitted correspond to
external arrival?:1

Enter a message number that may be transmitted as a result
of an external arrival:1

Enter % of the time this message is transmitted when an
external message arrives:100
```

Figure 6-4. (Continued)

receiving. Suppose that messages numbers 2, 4, 12, 17 and 20 are transmitted to this node and that stack delays associated with receiving are the same for messages 12, 17 and 20 but are different for messages 2 and 4. Default delays can be specified for messages 1 through 20 and the individual values can be specified for messages 2 and 4. In this example, no individual values for stack delays associated with receiving need to be specified. Thus "N" should be entered.

The user is asked if the default stack delay associated with transmitting is to be specified for messages that are transmitted from this node. The user types "N" if no default delay is to be specified. The user is also asked if individual stack delays associated with transmitting are to be specified. The user types "Y" to specify these values individually. The next question asks the user for the number of message delays to be specified. The user enters "2" to specify 2 message delays. The program asks for the first message number. The user enters "1" to specify the delay for message number 1. The user is asked for the value of delay for message number 1. The user enters "145000" for the value of delay for message number 1. The program asks for the second message number. The user enters "2" to specify the delay for message number 2. The user is asked for the value of delay for message number 2. The user enters "145000" for the value of delay for message number 2.

The program asks if the distances between nodes are to be calculated from the drawing, input by the user or assumed to be zero. The user enters "0" to indicate that the distances between nodes are assumed to be zero. The user types "1" in response to the question about the sub-network number for this node to indicate that node number 1 is on sub-network number 1.

The user is asked if messages arrive externally to be transmitted from this node. If so, then the user types "Y." Messages that arrive externally at a given node are messages that first originate on that node. In contrast to the spawned messages that are generated from messages received at a given node, external message arrivals are generated at some specified interval. The user may specify the rates at which these messages are to be generated (arrive externally) on any given node. In this example, message number 1 arrives externally to this node with a constant time between arrivals of 10 seconds (1.0E7 microseconds). Therefore, the user should type "C" to select the constant arrival rate and "1.0E7" for the time between arrivals. If there is only one arrival during the entire simulation run, then "O" should be typed. If the arrival rates are random, then the user must be able to specify the statistical distribution. Notice that to specify the name of the statistical distribution, the user does not need to type the full name. Only the capitalized part of the name is required.

The next question asks for the number of messages that may arrive externally. The user types "1" to indicate that only one message may arrive externally. Then the user types "1" again for the first message number of the message that arrives externally. The subsequent question asks for the percentage of time that an external message arrival is message number 1. Since there is only one message that arrives externally to this node, the user should type "100." If there were more than one external message arrival, then the program would loop back and ask for the second message number and its percentage. The user is asked for the insertion point of the station on the drawing. The coordinates of the insertion point can be typed on the command line or selected by pointing to it and clicking the left mouse button. Figure 6-5 shows the drawing of station 1.

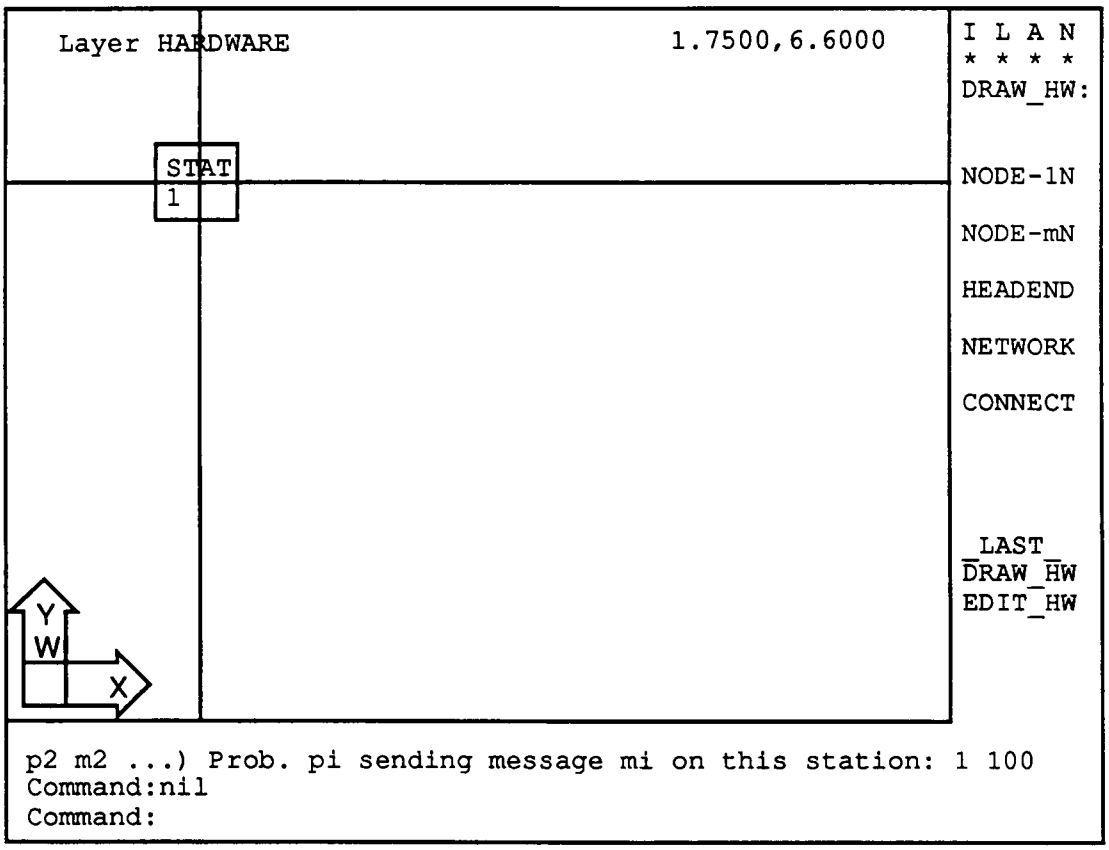


Figure 6-5. Drawing of Station 1

To unclutter the screen and to improve the speed of AutoCAD, the attributes in the drawing are not normally displayed. The user may make these attributes visible by using the AutoCAD's ATTDISP command. This command can be typed on the command line or selected from the menu. To select it from the menu, the user may have to go through several menu screens. If the current screen displays the DRAW_HW menu, then the user can go to the root menu by selecting ILAN from the current menu (bottom screen of Figure 6-6). When ILAN's main menu is on the screen, the ATTDISP can be selected. Through the ATTDISP command, the user can switch the attribute display mode to either on, off or normal. The top screen of Figure 6-6 shows the final screen after the attribute display mode was turned on.

The DISPLAY menu may be selected from the MESSAGE, HARDWARE as well as the root menu. The user may use AutoCAD's display commands such as the ZOOM or PAN to magnify the display or to move to the other parts of the drawing that are not shown on the screen. However, after using either ZOOM or PAN command, the attributes may not be displayed on the screen even if the ATTDISP command is used to make these attributes visible. In this case the user should execute AutoCAD's REGEN command from either the command line or the DISPLAY menu. In any case, the drawing is regenerated with the values of the attributes that should be visibly displayed on the screen. Note that if the attributes are normally invisible and the ATTDISP command has not been issued to change their status, then invoking the REGEN command does not show them on the screen.

The user may move to either the HARDWARE, MESSAGE or OTHER_PA layer by selecting it from the DISPLAY menu. Furthermore, any specific message, node or network can be located and displayed on the screen by selecting FIND_MS,

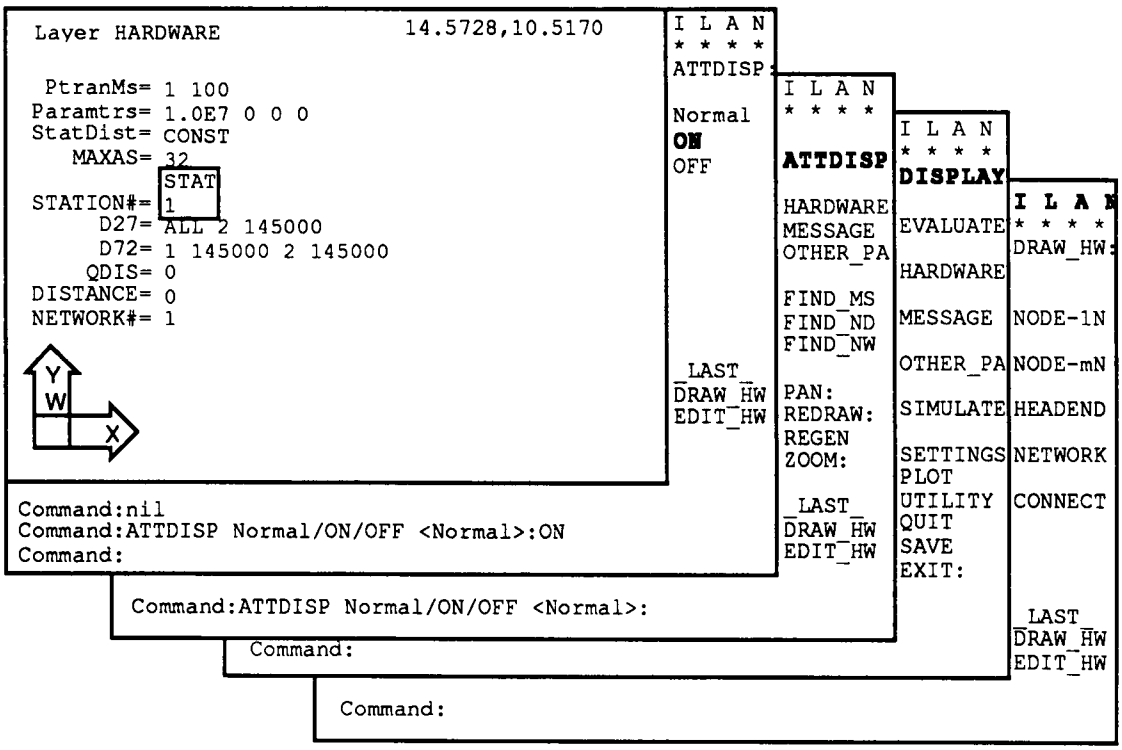


Figure 6-6. Attribute Display

FIND_ND or FIND_NW, respectively. These commands may also be called from the command line. For example, if the current screen displays the HARDWARE layer but the user wants to move to MESSAGE layer and display message number 2, then FIND_MS should be invoked. After the FIND_MS procedure obtains the message number of the message to be displayed from the user, it switches the layer to the MESSAGE layer and zooms in on the specified message.

From the ATTDISP menu screen, the user can go directly back to the DRAW_HW menu screen by selecting the DRAW_HW item from the menu. To define another node connected on another sub-network, the same NODE-1N can be selected again. Figure 6-7 shows station number 2 on sub-network number 2. The station has no external message arrivals. For each packet belonging to message number 1 that arrives at this station, a 145000-microsecond stack delay is experienced. For each packet belonging to message number 2 that is transmitted, a 145000-microsecond stack delay is imposed. Distances between stations are assumed to be zero.

A bridge can be drawn by selecting NODE-mN from the DRAW_HW menu or by typing NODE-mN on the command line. Figure 6-8 shows an example of the text screen for interactions between NODE-mN and the user. There is only one difference between NODE-1N and NODE-mN. The additional question in NODE-mN procedure asks if the type of item inserted is a bridge, a filter or a station. The picture of the inserted bridge with the attribute display turned on is shown in Figure 6-9.

If there is no more station, bridge or filter to be inserted, then it is time to draw the sub-networks. The NETWORK item is selected from the menu. After the information about the network is supplied, the user is asked for the insertion point where the information is drawn on the drawing. The user is also asked to draw the

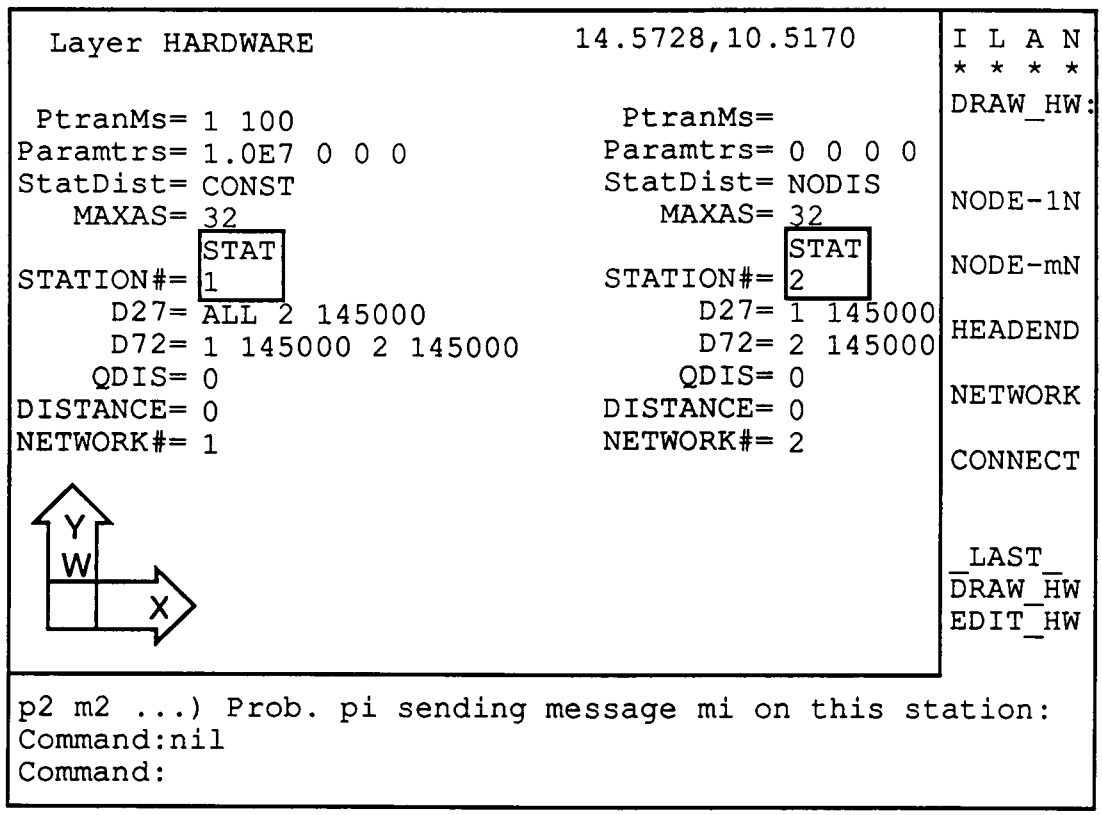


Figure 6-7. Station 2

```
Station number for this station :3

Is this a Bridge, Filter or Station ? (B,F or S):B

Would you like to specify default reception stack delays
for messages that are transmitted from this node (Y/N)?
<N>:Y

Enter last message no. that uses default reception stack
delays at this node [message 1 through this last message
no. uses this default delay]:2

Enter the default reception stack delay in microseconds
<36500>: 145000

Would you like to specify individual stack delay for
message reception [these individual delays over ride any
default previously set] (Y/N) <N>:N

Would you like to specify default transmission stack delays
for messages that are transmitted from this node (Y/N)?
<N>:Y

Enter last message no. that uses default transmission stack
delays at this node [message 1 through this last message
no. uses this default delay]:2

Enter the default transmission stack delay in microseconds
<36500>: 145000

Would you like to specify individual stack delay for
message transmission [these individual delays over ride any
default previously set] (Y/N) <N>:N

Find distances from drawing, User input or assume 0 (D, U,
or 0):0
How many sub-networks are connected to this node:2

Enter sub-network number of a sub-network connected to this
node:1

Enter sub-network number of a sub-network connected to this
node:2

Messages arrive externally to this station ? (Y,N):N
```

Figure 6-8. NODE-mN Input Text Screen

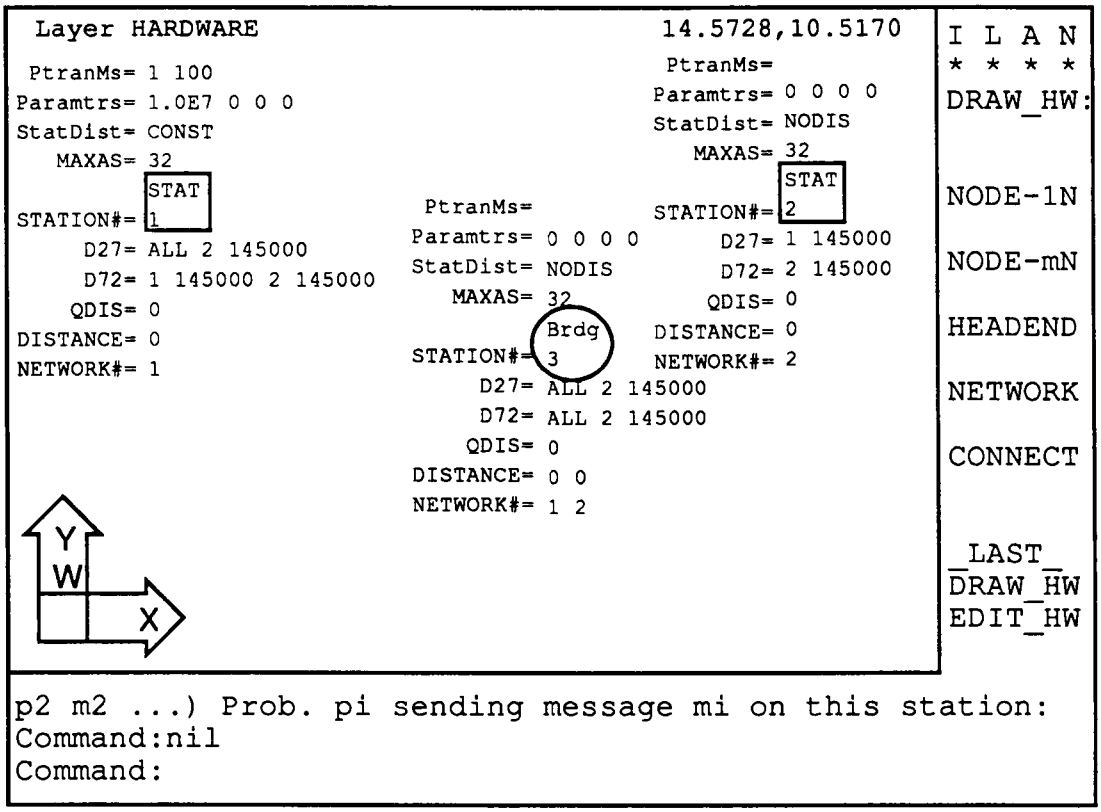


Figure 6-9. Bridge Insertion

cable line segments. These line segments must be drawn either vertically or horizontally. After drawing the cable line, the user should hit the CR key. Figure 6-10 shows the text screen of the input for a broadband sub-network that connects station 1 and bridge 3. The insertion point (X=3.7878, Y=1.3519 in Figure 6-10) for the network is used as the starting point for the cable line segment. The procedure asks for the end point of the line segment and writes "To:" on the command line. The user may either select the end point of the line segment on the drawing or type the coordinate on the command line following the "To:" prompt. In Figure 6-10, the coordinate (3.7878, 7.9508) is typed on the command line. The cable line segment is then drawn on the drawing. Coordinate (3.7878, 7.9508) becomes the beginning of the next line segment, and the program asks for the end point of the line segment by issuing another "To:" prompt. The user can continue drawing additional line segments in this manner. To end the process, the user needs to hit the CR key. For each broadband sub-network, the user is asked to select the point where the headend remodulator is to be inserted. The headend remodulator can be placed at either end of the cable. Figure 6-11 shows the text screen of the input for a carrierband sub-network that connects station 2 and bridge 3. The colons (:) at the beginning of any lines in Figures 6-10 and 6-11 indicate that these lines contain AutoCAD messages that were not reproduced. The drawing of sub-networks number 1 and 2 are shown in Figure 6-12.

To connect each defined (drawn) station to its corresponding defined sub-networks, the user must execute the CONNECT command. This command can be invoked from the command line or the DRAW_HW menu. Figure 6-13 shows these connections after the command was issued. Notice that the edge of each node is connected to its corresponding main cable sub-network.

```
Command:NETWORK
Carrier-band or Broadband (C or B) <B>:B
Data transmission rate (Mbits/sec.) <10>:10
Token holding time (microseconds):5000
Sub-network number for this sub-network:1
:
Insertion point:3.7878,1.3519
:
Command: To:3.7878,7.9508
:
Command: To:
:
HEADEND Insertion point:3.43,1.11
```

Figure 6-10. Input for Sub-network Number 1

```
Command:NETWORK
Carrier-band or Broadband (C or B) <B>:C
Data transmission rate (Mbits/sec.) <10>:5
Token holding time (microseconds):5000
Sub-network number for this sub-network:2
INSERT Block name (or ?):NETWORK Insertion
point:11.7019,3.1849
:
Command: To:11.7019,7.4620
:
Command: To:
```

Figure 6-11. Input for Sub-network Number 2

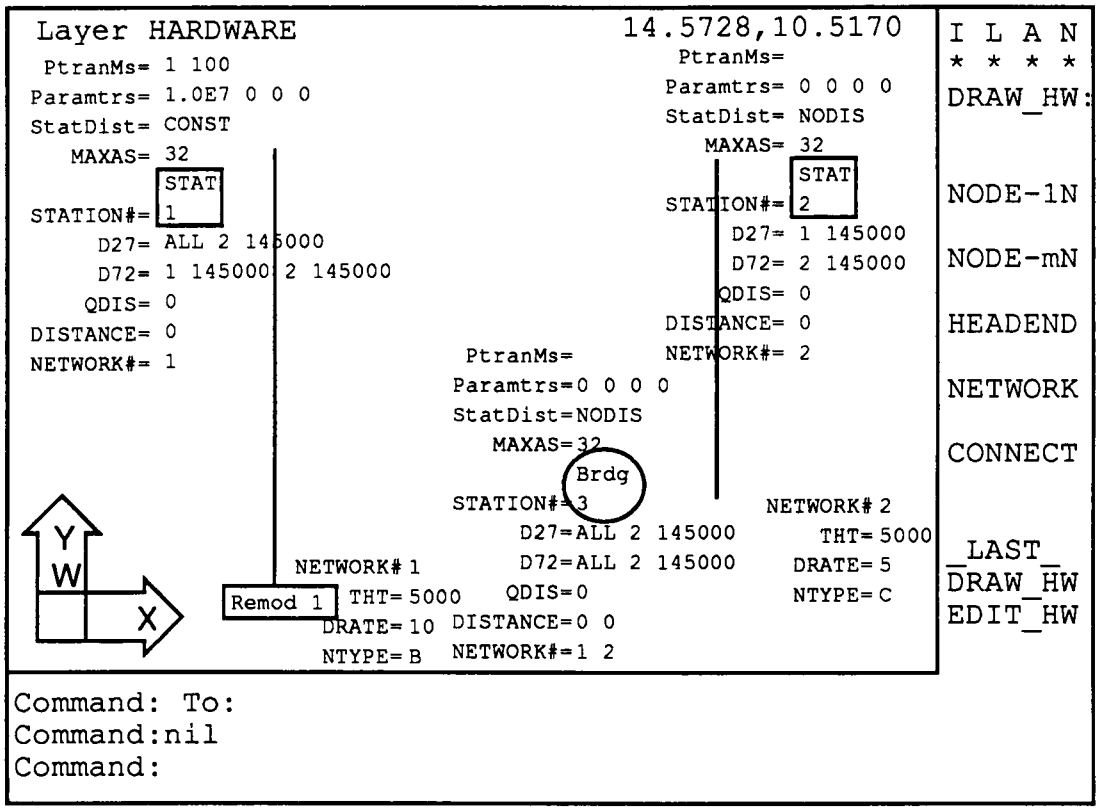


Figure 6-12. Drawing of Sub-networks 1 and 2

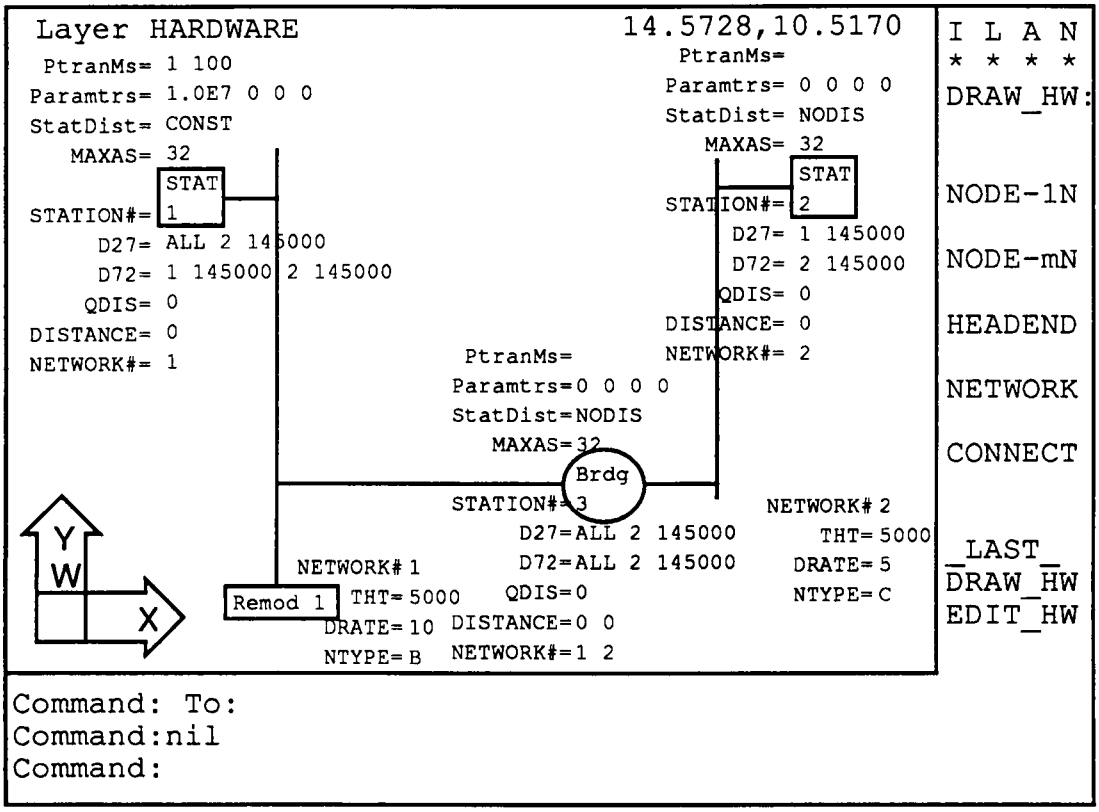


Figure 6-13. Connections of Sub-networks 1 and 2

To edit the existing hardware drawing the user should select EDIT_HW from the menu. From the menu, ATTEDIT, DELETE, ADD_DEV, MOVE and UNDO can be selected. The ATTEDIT, MOVE and UNDO are AutoCAD commands. ATTEDIT command is issued to edit attributes. For example, to change the value of maximum associations for station 2 from 16 to 32, the ATTEDIT command can be activated. The user needs to hit the CR key four times, select the number (16) that is to be changed on the screen, hit the CR key again, type V, R, and the new number (32) and hit the CR key once again. The MOVE command is called to move the items that had been initially inserted at one location to another. The UNDO command allows the user to undo changes that had been made to the drawing during the current editing session. While the DELETE and ADD_DEV are ILAN functions, the DELETE command can be executed through the menu or the command line. After the item (or items) to be erased is selected, the user must hit the CR key to end the DELETE command. For example, to delete station number 2 from the drawing, the DELETE command is executed. The user is asked to select the item to be deleted. The user can point at any visible part of station number 2 and hit CR key. Figure 6-14 shows the EDIT_HW menu screen and the result of deleting station number 2 from the drawing.

To add devices to the existing drawing, the user can select ADD_DEV from the menu. After the selection, the ANODE-1N and ANODE-mN are shown on the menu. Apart from the additional function of connecting the newly defined node to its corresponding sub-network, ANODE-1N is equivalent to NODE-1N and ANODE-mN is also equivalent to NODE-mN. For example, the user can re-insert station number 2 that had just been deleted by executing ANODE-1N. Recall that ANODE-1N or ANODE-mN is called to add a node to the drawing and connect the node to its

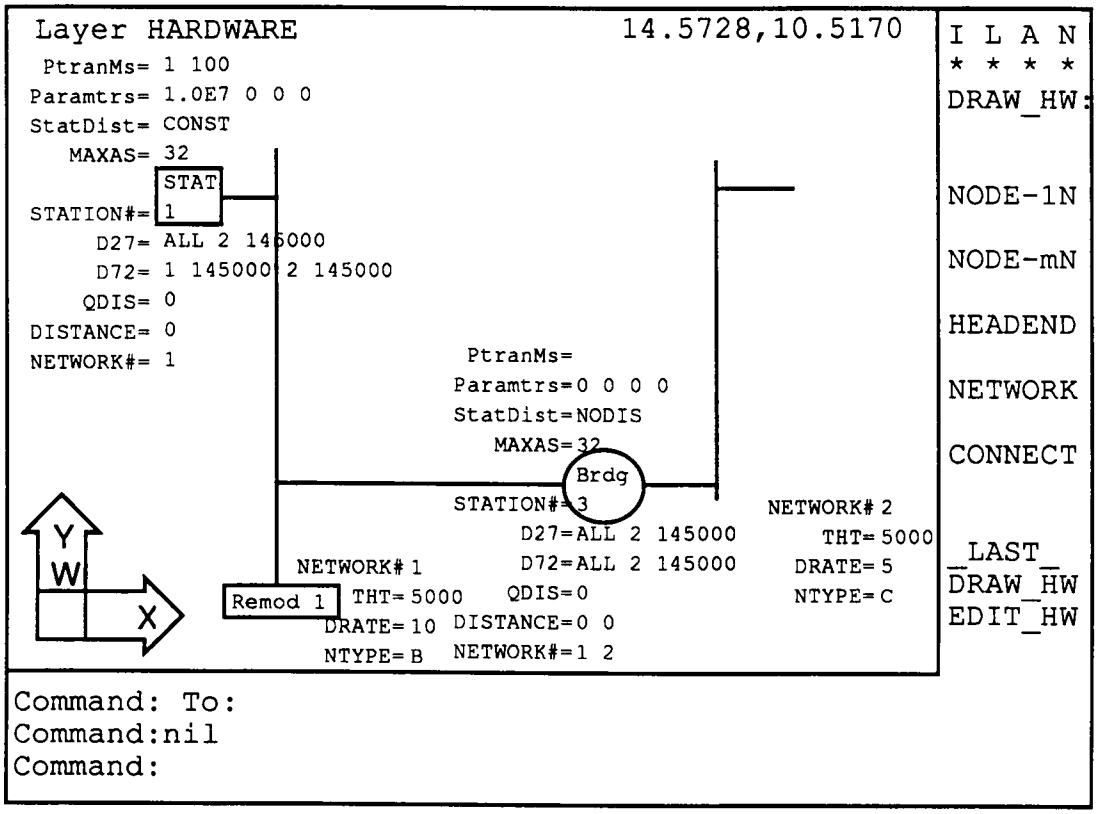


Figure 6-14. Deletion of Station 2

corresponding defined sub-network(s). However, the CONNECT command is issued to connect every defined stations to their corresponding defined sub-networks. Thus, if station number 3 that is on sub-network number 3 is to be inserted to the existing drawing of Figure 6-13, then sub-network number 3 must be inserted first before issuing the ANODE-1N command.

6.3 Message Definition

To start the message definition, the MESSAGE item can be selected from the ILAN's main menu followed by choosing DRAWMS item from the MESSAGE menu. DRAWMS can also be executed from any menu screen by typing DRAWMS on the command line. Figure 6-15 shows an example of the input needed for DRAWMS procedure on the text screen. After the DRAWMS command is issued, the user is asked several questions about the message.

The first question asks if the message to be defined had already been partly defined or if it is a new message that has never been defined. The user also has the option of ending the message definition command by typing "E" at this point. For the example in Figure 6-15, "N" would be entered to indicate that a new message is being defined. The program proceeds by asking for the number of possible destination nodes for message number 1. Since there is only one destination node for message number 1, "1" is typed. The procedure then asks for the message number followed by the percentage of times that this message is transmitted to each of its destination stations. For this example, the message number is 1, and the message is always transmitted to node number 2. Thus "2" and "100" should be entered.

```
Command:DRAWMS

Current message is New message, Defined message or End
(N,D,E):N

Message number:1

How many destination nodes can this message possibly
transmitted to ?:1

Enter a node number from the list of possible destination
nodes:2

What is the percent of time that this is the destination
node ?:100

Start message # (if current message is a group message)
<0>:

Association status for this message <0>:

Press 1=write request, 2=write response, 3=read request,
4=read response,or 5=file transfer <5>:3

How many elements to read or write ? <1>:10

How many possible spawn values for this message (0 spawn is
1 value)? <1>:

Enter a number to spawn from the list of possible spawn
values <0>:1

What is the % of time that this number is spawned from this
message?:100

How many different nodes can this message be transmitted
from ?:1

Enter a node number from the list of nodes that may
transmit this message:1

Enter application delay (mic s) for this message
transmitted from this node ?:0
INSERT Block name (or ?):MESSAGE Insertion point:
```

Figure 6-15. New Message Definition

Next, the user is asked for the start message number. Recall that a start message number is required for each group message. This value should be zero for any message that is not a group message. The value of the association status for this message is then requested. In this case, no association is made and a value of zero is given to the association status. The category of message for message number 1 is asked next. In this case, message number 1 is a read request. Thus, "3" should be typed. The next question asks for the number of data elements to read or write. For this problem, message 1 requests 10 data elements to be read. Hence, "10" should be entered.

The program proceeds by asking for the number of possible spawn values. There must be at least one possible spawn value. A spawn value of zero counts as one possible spawn value. The user is then asked to input the number of times this message may spawn new messages and their percentages. In this case, message number 1 always spawns 1 message after it has been received at layer 7 of its destination node. Therefore, "1" and "100" respectively should be entered in response to the next two questions.

The last information needed for message number 1 involves the application delays for transmitting this message from different stations. For this example, message number 1 is always transmitted from station number 1, and the application delay is assumed to be zero. The user is asked for the number of different nodes that this message can be transmitted from. "1" is entered to signify that this message can be transmitted form one node. The procedure then asks for the node number that this message can be transmitted from. "1" is typed to specify that the message is transmitted

from node 1. Finally, the value of the application delay for this message transmitting from node 1 is asked. "0" is typed to indicate that the application delay is zero.

After all of the above questions have been answered, the message is inserted in the drawing at the location specified by the user. Figure 6-16 shows the drawing of message number 1 with all of its attributes displayed. Normally, only the message number in the center of the circle is drawn to represent any particular message.

Notice that the values of the message numbers and the percentages for spawning them (P(NEXT_MESS) in Figure 6-16) have not been filled. However, the DRAWMS command has not ended. DRAWMS continues to ask if the message to be spawned from message number 1 is a new or a defined message. At this point the user has the option of ending the DRAWMS command by typing E. The user may end the DRAWMS command, save the drawing and continue the message definition in another drawing session. If the message does not spawn any other messages, then there is no need to continue with the message definition of this message.

If message number 2 is spawned from message number 1, then the user should type "N" on the command line in response to the last question of Figure 6-16. The procedure continues to ask for the message number of the new message to be spawned. This new message is message number 2. After typing 2, the user is asked to select an insertion point where information about the spawned message is to be placed on the drawing. When the insertion point is selected, the partly defined spawned message is displayed on the drawing. The percentage of times that this message is spawned from the parent message is requested. For this problem message number 1 spawns message number 2 and nothing else. Thus the user should type 100 in response to the question.

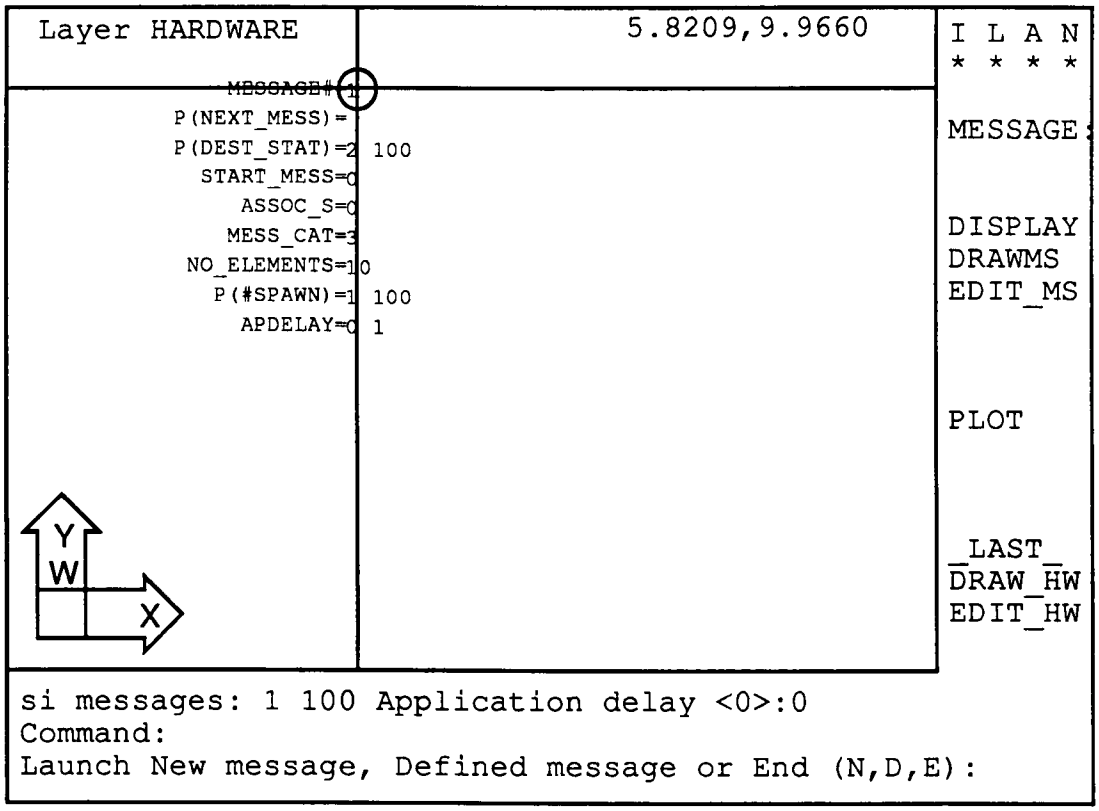


Figure 6-16. Drawing of Message Number 1

An arrow is drawn from the parent message to the spawned message. Figure 6-17 shows the drawing of message numbers 1 and 2.

The procedure asks if the message to be spawned from message number 1 is new or not (bottom of command line in Figure 6-17). At this point the user may continue to define another message that is spawned from message number 1. However, for this problem no other message is spawned from message number 1. Thus E is specified in response to the question. The procedure then asks if the user wants to start defining a new message or a message that had been inserted on the drawing. Here, a full description of message number 2 needs to be inserted into the drawing. Thus, D is the option that is selected. The user is asked to select the node of the message to be defined. After the node of message number 2 is selected, it is deleted from the drawing. The user is interrogated for the information about message number 2.

Figure 6-18 shows the input for the definition of message number 2 on the text screen. There is only one node that receives message number 2. Message number 2 is always transmitted to node number 1. It is not a group message and does not involve association connection or disconnection. The category for the message is a read response. The number of data elements for this read response is 10. This message does not spawn another message. The message can only be transmitted from node number 2. The application delay for this message is assumed to be zero when it is transmitted from node number 2. After the information about the message has been filled, the message is re-inserted at the same place.

Figure 6-19 shows the drawing of message numbers 1 and 2. At this point the user may specify messages that are spawned from message number 2. For this

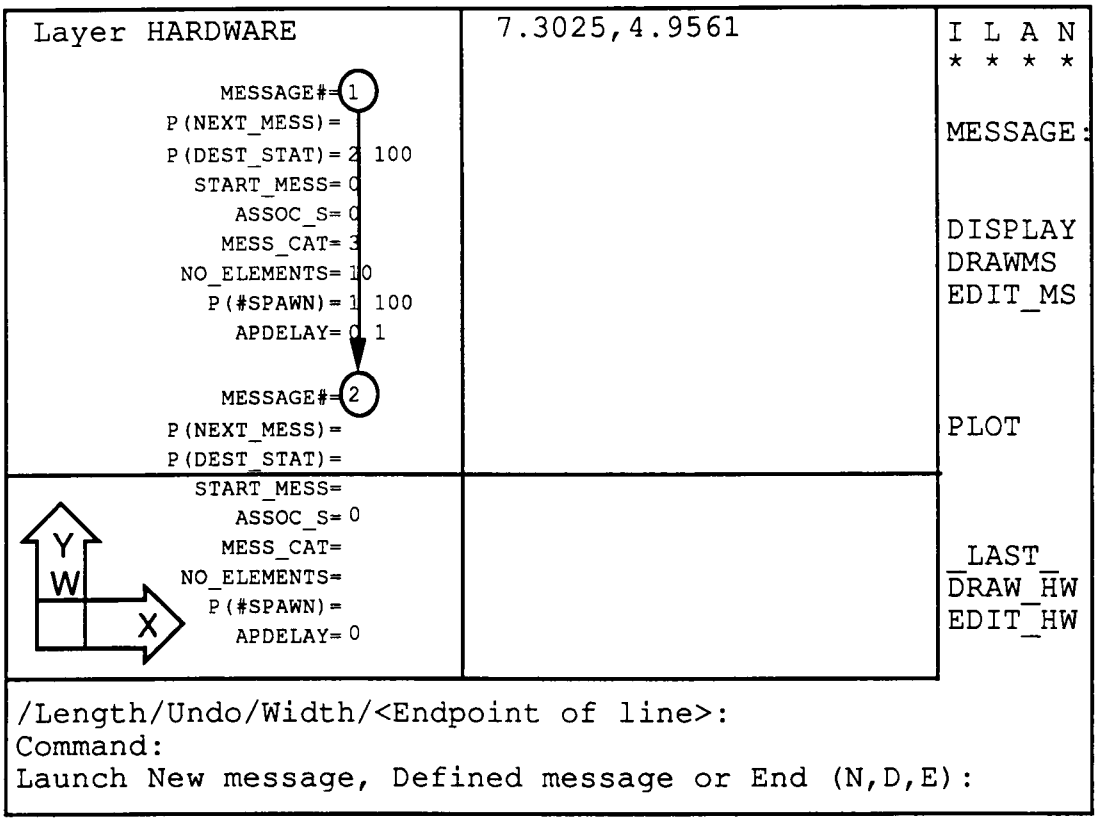


Figure 6-17. Message Numbers 1 and 2

```
Current message is New message, Defined message or End
(N,D,E):D

Select the defined message node:

How many destination nodes can this message possibly
transmit to ?:1

Enter a node number from the list of possible destination
nodes:1

What is the percent of time that this is the destination
node ?:100

Start message # (if current message is a group message)
<0>:

Association status for this message <0>:

Press 1=write request, 2=write response, 3=read request,
4=read response,or 5=file transfer <5>:4

How many elements to read or write ? <1>:10

How many possible spawn values for this message (0 spawn is
1 value)? <1>:

Enter a number to spawn from the list of possible spawn
values <0>:0

What is the % of time that this number is spawned from this
message?:100

How many different nodes can this message be transmitted
from ?:1

Enter a node number from the list of nodes that may
transmit this message:2

Enter application delay (mic s) for this message
transmitted from this node ?:0
INSERT Block name (or ?):MESSAGE Insertion point:s1 p1 s2
p2 ... transmit to station si with pi%:1 100
```

Figure 6-18. Definition of Message Number 2

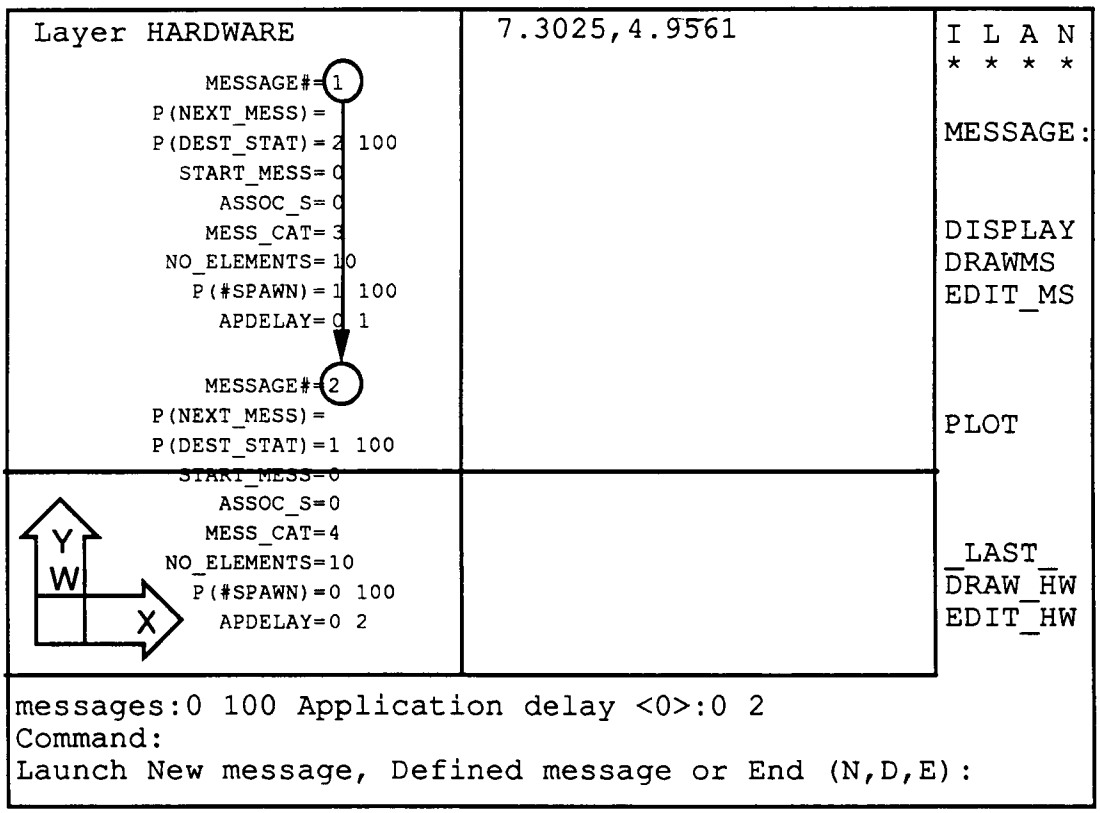


Figure 6-19. Drawing of Message Number 2

example, message number 2 does not spawn any message. Thus, E should be typed in response to the last question on the command line of Figure 6-19. After ending the message spawning definition from message number 2, the user is asked if any other messages are to be defined. For this example, there are only two messages in the system. Thus, another E is typed to end the message definition of the DRAWMS procedure.

The message attribute P(NEXT_MESS) for a particular parent message represents the values of its spawned message numbers and their spawned percentages. Notice that the value of P(NEXT_MESS) for message number 1 in Figure 6-19 is blank even though message number 2 is spawned from it. This is because the drawing was not automatically updated in anticipation of speeding up the message definition procedure. However, the user may update the drawing by using the AutoCAD's REGEN command. Thus, if REGEN is typed on the command line, P(NEXT_MESS) = 2 100 should be displayed on the screen indicating that message number 1 spawns message number 2 one hundred percent of the time.

From the MESSAGE menu, EDIT_MS may be selected to edit the messages displayed on the screen. Figure 6-20 shows the EDIT_MS menu. Several of AutoCAD's editing commands such as ATTEDIT, MOVE, STRETCH and UNDO are found on this menu. In addition to the AutoCAD's commands, an ILAN command called DELMS, which deletes messages, can be selected. The DELMS procedure can be executed from the menu or from the command line.

When the DELMS procedure is invoked, the user is asked to select the node message to be deleted. Through a pointing device such as a mouse the user can move the square icon over the message node that is to be deleted and select it. Figure 6-20

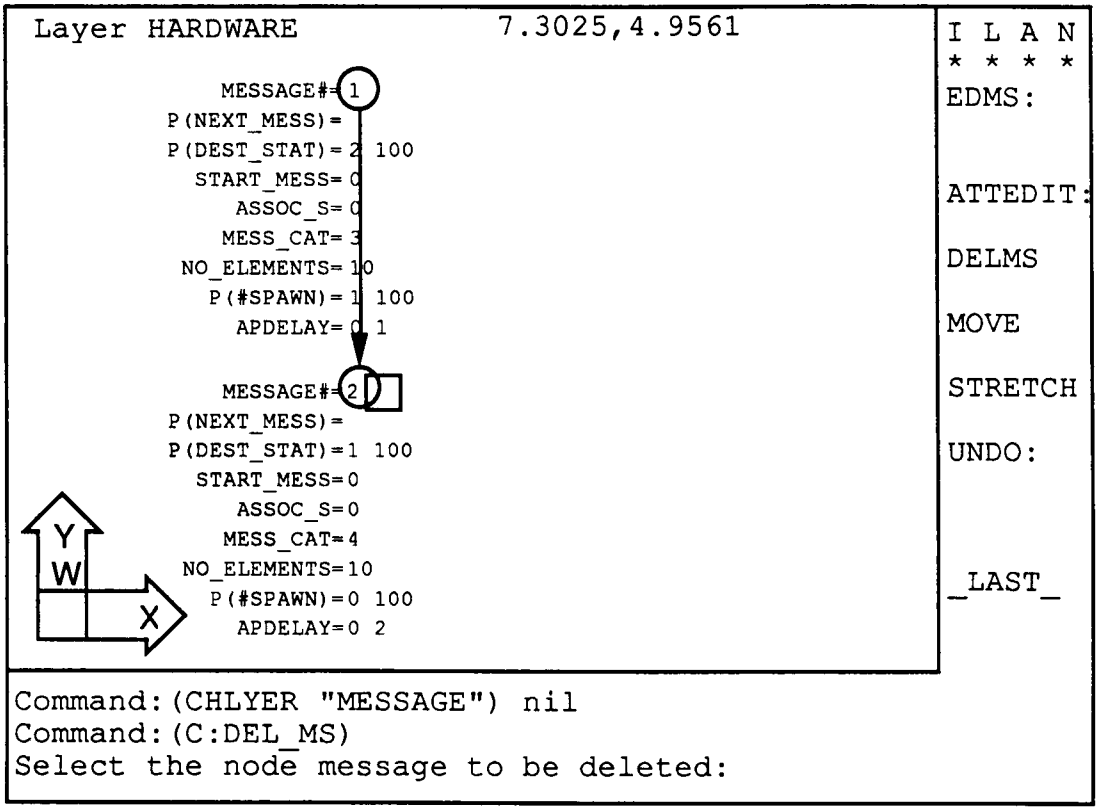


Figure 6-20. Edit Message

shows the square icon over message number 2 node. As soon as the message is selected, it is deleted from the screen and from the database. Figure 6-21 shows message number 2 deleted from the screen after it had been selected. The DELMS procedure asks the user to select the message that launched the message that had just been deleted. This information is needed in order to update the value of P(NEXT_MESS) for the parent message (message number 1). When message number 1 is selected, the internal update is performed. However, as before, the drawing is not updated. The procedure continues by asking the user to select the arrow line that connects the deleted message to the parent message (see Figure 6-22). The DELMS command ends when the selected arrow line is deleted from the drawing. If the deleted message has no parent message, then the user should hit the CR key twice to end the DELMS procedure. If the user accidentally presses CR key twice after deleting a message that has a parent message, then the value of P(NEXT_MESS) of the parent message must be edited using ATTEDIT command. The arrow line can be erased using the AutoCAD's ERASE command.

6.4 Other Parameter Definition

To complete the model definition of a MAP network in ILAN, additional parameter input is required. This is accomplished by executing the DROTPA procedure. To invoke the command from the menu, the user must select OTHER_PA from the main menu. From the OTHER_PA menu, the DROTPA menu item can be selected. Figure 6-23 shows the text input screen of OTHER_PA. In this case the maximum length of the simulation run is specified to be 1.0E8 microseconds. The simulation run is to be ended when the accuracy is within 10% of the mean with a

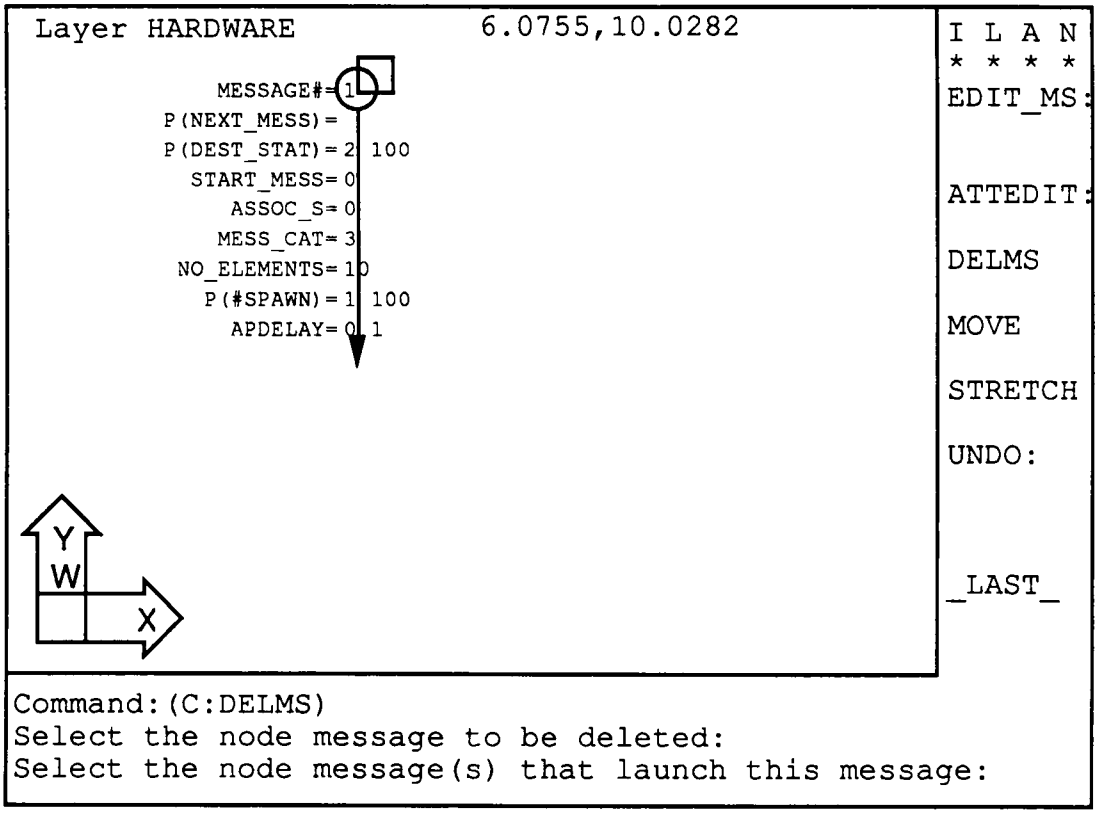


Figure 6-21. Message Number 2 Deleted

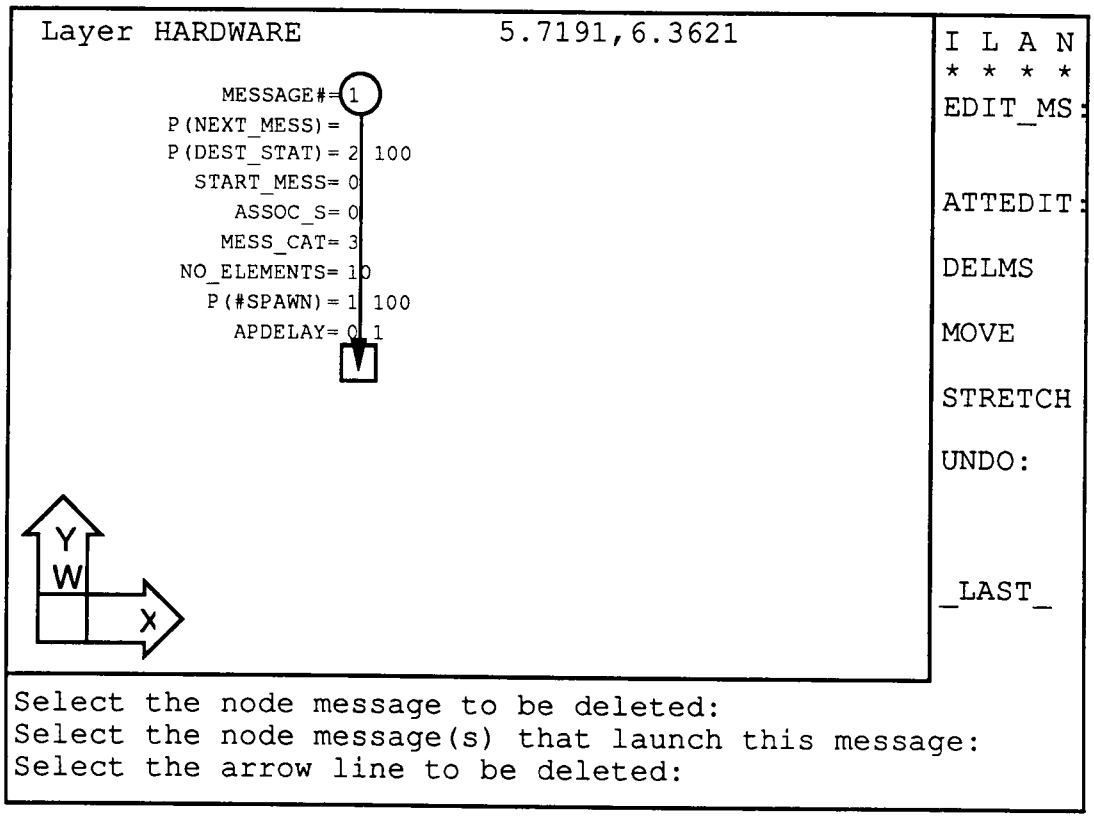


Figure 6-22. Deleted Arrow Line

```
What is the maximum length of the simulation run in
microseconds <1.0E8>:1.0E8

Acceptable range (+ or -) % from mean <10.0>:10

Percent confidence interval for end of simulation
<90.0>:90.0

Percent total transmission error <0.0>:0.0

Time to start debug (microseconds) <.2e20>:

Minimum number of observations for end of transient state:1

Distances measured in meters or feet (M or F) <M>:

Fiber Optic cables (Y/N) <N>:

Long output (Y/N) <Y>:Y

Fast simulation producing less accurate results] (Y/N)
<N>:Y

Calculate token rotation time [slows down the simulation in
light traffic] (Y/N) <Y>:
```

Figure 6-23. Other Parameter Input

confidence interval of 90 percent. The total transmission error for each frame transmitted is assumed to be zero. A large simulation time is specified for the time to start debugging to suppress outputting values of simulation variables during the execution. For this example, the end of the transient state is assumed to end after one observation is collected. The distances specified in the drawing are in meters. The type of cable is coaxial. At the end of the simulation run a long output is to be generated. Fast token simulation algorithm (described in Section 4.5) is specified for this simulation run. This algorithm skips a lot of steps and may produce results that are a little less accurate, but it significantly increases the speed of the simulation program. Finally, the token rotation time for each sub-network is calculated, and this may slow the speed of the simulation run appreciably in light traffic situations.

CHAPTER 7

VERIFICATION AND VALIDATION

7.1 Verification and Validation Overview

The purpose of verification is to establish the correctness of the simulation program. However, up to now, only a few performance data for MAP networks are available. There have not been any analytical or simulation studies that consider the performance of MAP networks interconnected by bridges with different stack delays associated with transmitting and receiving at each node for each message.

To provide enough evidence that the simulation program is correct, several cases of analytical and experimental results are compared to the simulation results. Although the whole MAP network cannot be represented analytically, solutions for two separate parts of a MAP network are available. For analytical representation, a MAP network can be split into token bus (layers 1 and 2) and higher protocol stack models. Several analytical studies of the token bus protocol do exist. Using queuing theory, simple analytical results can be derived for stack delays in cases where these delays are large compared to the delays in layers 1 and 2. The purpose of validation is to confirm the agreement between the behavior of the model and that of the real system. Statistical comparison of simulation and experimental results are presented at the end of this chapter.

7.2 Verification of Token Bus Protocol

The token bus protocol can be simulated using ILAN by assuming that the delays for all stacks associated with receiving and transmitting are zero. Non-exhaustive token passing strategy of token bus protocol (described in Section 2.5) is to be employed. There are 50 stations on a 10 Mb/s network. Message arrivals at each station are assumed to follow a Poisson process. Mean time between message arrivals at every station is assumed to be the same. Time to transmit data is 327.2 microseconds, and time to transmit the token is 80 microseconds. For a 10 Mb/s cable transmission rate, a message transmission time of 327.2 microseconds requires a layer 2 message length of 3272 bits, which corresponds to approximately 2472 bits or 309 bytes of message length at layer 3. Assuming that 26 bytes of overhead is added to the file between layer 7 and layer 2 and the message is transmitted as a file from layer 7, then the size of the file at layer 7 is approximately 309 bytes minus 26 bytes or 283 bytes. The formula used to calculate the token transmission time [80] is

Token Transmission Time = [(Length of Preamble + Source Address Length + Destination Address + 7) * 8 * 1.0E6 / Cable Transmission Rate] + Modulation Delay + Demodulation Delay

For ILAN, the preamble length is assumed to be 8 bytes, the source address length is 6 bytes and the destination address length is 6 bytes. If the modulation and remodulation delays are assumed to be 26.8 microseconds and 30 microseconds, then the token transmission time of 80 microseconds is achieved.

For the problem described above, ILAN simulation results of the mean token rotation time and standard deviation of token rotation time for each arrival rate are tabulated in Table 7-1. In order to minimize the computer execution times, only two

Table 7-1. Simulated Mean Token Rotation Times vs. Arrival Rates

λ (arrival /sec.)	T Run 1 (μ sec.)	S_1 Standard Deviation of t for Run 1	Run 1 # of Observations n_1	T Run 2 (μ sec.)	S_2 Standard Deviation of t for Run 2	Run 2 # of Observations n_2
20.0	6455.47	1824.03	335	4856.90	1259.43	732
25.0	6767.03	2641.70	419	6515.48	1751.74	314
30.0	7916.84	1514.35	260	7497.49	2288.27	341
35.0	9219.15	1451.63	265	7516.00	2543.67	350
40.0	12072.53	1669.87	274	12409.04	1687.18	240
41.4	12707.83	1678.42	243	12109.72	2933.82	284
44.0	14667.48	1926.48	259	14511.93	1612.84	260
46.0	15739.11	1573.16	253	16064.35	1284.50	249
47.5	17123.64	1402.15	257	17219.92	1533.88	253

simulation runs at each arrival rate were executed. For each simulation run, the simulation is stopped after .5E7 microseconds of simulation time.

At an arrival rate of 47.5 arrivals per second, 257 observations or token rotations were counted before the simulation ended. If the token made 257 rotations in .5E7 microseconds of simulation time when there were 50 stations on the network, then the average slot time per station is .5E7/(50*257) or 389 microseconds. For a non-exhaustive token passing strategy, the network is 100% utilized when each station transmits a message followed by a token. It takes 327.2 microseconds to transmit a message and 80 microseconds to transmit a token. Thus, if each station transmitted a message followed by a token every time it had the token, then the average slot time was 327.2+80 or 407.2 microseconds. Since an average slot time of 389 microseconds was close to but less than the maximum slot time of 407.2 microseconds for the arrival rate of 47.5 arrivals per second, it can be concluded that the network was under heavy traffic at this arrival rate.

Simulation run number 1 and run number 2 used the same set of input parameters. The only difference between run number 1 and run number 2 at each arrival rate was the random seed numbers specified in the experimental frame. For every simulation run a different set of random seed numbers was employed. A set of random seeds is comprised of four different random seed numbers. These simulation runs were executed on the 16 MHz Compaq Deskpro 386 with an 80387 math co-processor. The execution time for each run ranged from 3.5 minutes to 7.5 minutes.

An analytic model of the token bus protocol was developed by Sethi and Saydam (S&S) [87]. The model assumes that there are M stations on the network. Messages arrive at each station at an average rate of λ with independent Poisson

distributions. An infinite buffer is assumed at each station. The time to pass the token is constant and is denoted by r. The time to transmit data is also constant and is denoted by x. The token rotation time, denoted by t, is defined as the elapsed time from the instant a station receives the token to the next instant the same station receives the token after the token had traveled all the way around the logical ring. The mean token rotation time is denoted by T and the traffic intensity is denoted by ρ. Since the token rotation is the only value that may be practically measured in the real system being modeled, validation is based on this parameter. The equations for calculating the mean token rotation times are:

$$\rho = M \lambda r / (1 - M \lambda x) \qquad (7\text{–}1)$$

and

$$T = M r + M \rho x. \qquad (7\text{–}2)$$

A single estimate for the variance of mean token rotation time can be pooled from variances of the two simulation runs (S^2_1,S^2_2) for each message arrival rate. Assuming that the two values estimate the common variance, the combined estimate is

$$S^2_T = \left[\frac{(n_1 - 1)S^2_1 + (n_2 - 1)S^2_2}{n_1 + n_2 - 2}\right]\left(\frac{1}{n_1} + \frac{1}{n_2}\right). \qquad (7\text{–}3)$$

These equations are used to calculate the mean token rotation times and their variances for the same cases that were described and simulated with the ILAN simulation program. The results of the two methods are compared through the

"analysis of variance" (ANOVA) procedure. Table 7-2 shows the mean token rotation times and their variances using Equations (7–1), (7–2) and (7–3) for x=327.2, r=80 and M=50 at different message arrival rates. In order to use the ANOVA procedure, replications of mean token rotation times are needed at each arrival rate. A mean token rotation time is a random variable that is the sum of token rotation times divided by the number of observations. These token rotation times are random variables. Thus, using the Central Limit Theorem, it is concluded that the mean token rotation time is normally distributed. The sample token rotation times can be generated from the normal distributions using random normal deviates. If z is the random variable from the standard normal distribution, Equation (7–4) describes required transformation for the sample mean token rotation time:

$$z = \frac{(T - \mu)}{S_T} \quad \Rightarrow \quad T = zS_T + \mu \qquad (7-4)$$

where T = sample mean token rotation time

μ = mean token rotation time

S_T = standard deviation of mean token rotation time.

Table 7-3 shows two sample mean token rotation times for arrival rates ranging from 20 to 47.5 arrivals per second. Sample mean token rotation times are calculated as follows:

1. Pick a random normal deviate value, denoted by z, from a random normal deviate table.

Table 7-2. S&S Mean Token Rotation Times and Their Variances vs. Arrival Rates

λ (arrivals/seconds)	Mean Token Rotation Time (microseconds)	Variance of Mean Token Rotation Time (microseconds)
20.0	5945.303	568190
25.0	6768.189	752490
30.0	7855.459	964210
35.0	9358.915	1179080
40.0	11574.070	1330910
41.4	12395.570	1337320
44.0	14277.560	1250250
46.0	16165.540	1020560
47.5	17945.270	673481

Table 7-3. S&S Sample of Mean Token Rotation Times vs. Arrival Rates

λ (arrivals per sec.)	Random Normal Deviate 1	Random Normal Deviate 2	S^2_T	T Sample 1 (microsec.)	T Sample 2 (microsec.)
20.0	1.102	0.148	568190	6051.45	5959.56
25.0	2.372	-0.145	752490	7175.96	6743.26
30.0	0.104	1.419	964210	7872.51	8088.06
35.0	0.069	0.797	1179080	9370.95	9497.98
40.0	-0.393	-0.874	1330910	11515.77	11444.41
41.4	0.125	-1.170	1337320	12422.19	12146.42
44.0	-1.091	2.304	1250250	14107.43	14636.85
46.0	-0.961	-0.783	1020560	16042.24	16065.08
47.5	0.487	-0.299	673481	18008.63	17906.37

2. Calculate mean token rotation time using Equations (7–1) and (7–2). These values are tabulated on Table 7-2.
3. Calculate variance of mean token rotation time using Equation (7–3).
4. Calculate sample mean token rotation time using Equation (7–4).

The arrival rates of 20 to 47.5 arrivals per second in Table 7-2 correspond to the traffic intensities of 0.1 to 0.85. A traffic intensity for a given arrival rate can be calculated by Equation (7–1). For example, for an arrival rate of 20 arrivals per seconds, the traffic intensity is

$$\rho = M\lambda r/(1 - M\lambda x) = \frac{50(20/10^6)(80)}{(1 - 50(20/10^6)(327.2)} = 0.1 .$$

The Sethi and Saydam (S&S) analytical results as well as the ILAN runs 1 and 2 simulation results of the mean token rotation times for each arrival rate from Table 7-3 are plotted in Figure 7-1. From the plot, it can be seen that simulation runs 1 and 2 closely match the analytical results. Note that better simulation results may be obtained by increasing the duration of each simulation run. Short simulation runs were used to demonstrate that long simulation runs are not needed to obtain reasonable results in those cases when the variations in the model are small.

There are two controllable factors in this experiment: (1) the method of calculating the mean token rotation times and (2) the message arrival rates. Since both factors are specified, this experiment is termed a "two-way classification, fixed effects model." The observed values of mean token rotation times may be described by Equation (7–5):

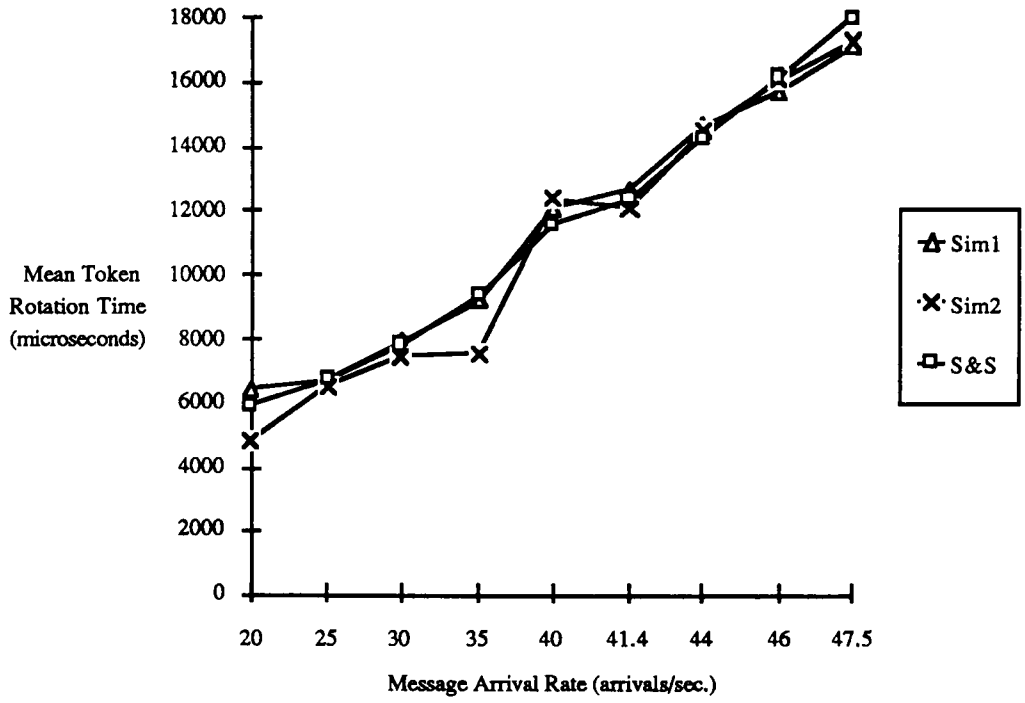


Figure 7-1. Simulated and S&S Mean Token Rotation Times vs. Arrival Rates

$$T_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk} \qquad (7-5)$$

where $i = 1, 2, ..., a$

$j = 1, 2, ..., b$

$k = 1, 2, ..., n.$

In Equation (7–5), μ is the overall mean effect, α_i is the effect of the i^{th} level of message arrival rate. Sample mean token rotation times were computed for nine different arrival rates. Therefore, there are nine levels of message arrival rates (a=9). The value of β_j is the effect of the j^{th} method of determining the mean token rotation time. There are two methods of determining the mean token rotation times (b=2). Method 1 is the ILAN simulation method (j=1), and method 2 is the analytical method by Sethi and Saydam (j=2). The value of $(\alpha\beta)_{ij}$ is the effect of the interaction between methods of determining the mean token rotation time and the message arrival rates, and ε_{ijk} is the random error. There are 2 samples for each method at each arrival (n=2). The hypotheses to be tested are

1. H'_0: $\alpha_1 = \alpha_2 = ... = \alpha_9 = 0$,

   H'_1: at least one of the α_i is not equal to zero.

2. H''_0: $\beta_1 = \beta_2 = 0$,

   H''_1: at least one of the β_i is not equal to zero.

3. H'''_0: $(\alpha\beta)_{11} = (\alpha\beta)_{12} = ... = (\alpha\beta)_{92} = 0$,

   H'''_1: at least one of the $(\alpha\beta)_{ij}$ is not equal to zero.

The following equations are used in the ANOVA calculations:

$$T_{i..} = \sum_{j=1}^{b}\sum_{k=1}^{n} T_{ijk} \qquad (7-6)$$

$$T_{.j} = \sum_{i=1}^{a}\sum_{k=1}^{n} T_{ijk} \qquad (7-7)$$

$$T_{ij.} = \sum_{k=1}^{n} T_{ijk} \qquad (7-8)$$

$$T_{...} = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} T_{ijk} \qquad (7-9)$$

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} T_{ijk}^2 - \frac{T_{...}^2}{abn} \qquad (7-10)$$

$$SS_A = \sum_{i=1}^{a} \frac{T_{i..}^2}{bn} - \frac{T_{...}^2}{abn} \qquad (7-11)$$

$$SS_B = \sum_{j=1}^{b} \frac{T_{.j.}^2}{an} - \frac{T_{...}^2}{abn} \qquad (7-12)$$

$$SS_{subtotals} = \sum_{i=1}^{a}\sum_{j=1}^{b} \frac{T_{ij}^2}{n} - \frac{T_{...}^2}{abn} \qquad (7-13)$$

$$SS_{AB} = SS_{subtotals} - SS_A - SS_B \qquad (7-14)$$

$$SS_E = SS_T - SS_{subtotals}. \qquad (7-15)$$

Table 7-4 shows the sample mean token rotation times to be tested and their totals. Table 7-5 shows the analysis of variance table for the mean token rotation times of token bus protocol.

Table 7-4. Sample of Mean Token Rotation Times and Their Totals

	Method b_1: Determine Mean Token Rotation Times Using ILAN Simulation Technique			Method b_2: Determine Mean Token Rotation Times Using S&S Analytical Technique			
λ (arrival /sec.)	T_{i11} Sample 1 (μ sec.)	T_{i12} Sample 2 (μ sec.)	$T_{i1.}$	T_{i21} Sample 1 (μ sec.)	T_{i22} Sample 2 (μ sec.)	$T_{i2.}$	$T_{i..}$
20.0	6455.47	4856.90	11312.37	6051.45	5959.56	12011.00	23323.3
25.0	6767.03	6515.48	13282.51	7175.96	6743.26	13919.22	27201.7
30.0	7916.84	7497.49	15414.33	7872.51	8088.06	15960.56	31374.9
35.0	9219.15	7516.00	16735.15	9370.95	9497.98	18868.93	35604.1
40.0	12072.53	12409.04	24481.57	11515.77	11444.41	22960.18	47441.8
41.4	12707.83	12109.72	24817.55	12422.19	12146.42	24568.61	49386.2
44.0	14667.48	14511.93	29179.41	14107.43	14636.85	28744.28	57923.7
46.0	15739.11	16064.35	31803.46	16042.24	16065.08	32107.33	63910.8
47.5	17123.64	17219.92	34343.56	18008.63	17906.37	35915.00	70258.6
$T_{.j.}$	201369.91			205055.11			406425.0

Table 7-5. ANOVA Table for Mean Token Rotation Times

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	Computed F
Message Arrival Rates	562047646.5	8	70255955.8	364.6415
Determination Methods	377241.5	1	377241.5	1.9580
Interaction	2340899.6	8	292612.5	1.5187
Error	3468084.1	18	192671.3	
Total	568233871.7			

At the 10% level of significance, $F_{(0.1,8,18)} = 2.04$ is less than computed the F statistic for message arrival rates of 364.6415 (Table 7-5). Therefore, H'_0 is rejected, and it can be concluded that message arrival rates affect the mean token rotation time. Since $F_{(0.1,1,18)} = 3.01$ is greater than the computed F statistic for the determination methods of 1.9580 (Table 7-5), H''_0 cannot be rejected. Thus, it can be concluded that there is no significant difference between the methods used in predicting the mean token rotation times. Furthermore, $F_{(0.1,8,18)} = 2.04$ is greater than the F statistic for the interactions between different arrival rates and determination methods of 1.5187 (Table 7-5). Hence, H'''_0 cannot be rejected, and it can be concluded that there is no interaction between different arrival rates and methods of determining the mean token rotation times.

The most important conclusion is that there is no significant difference between the methods used in predicting the mean token rotation times. In other words, the mean

token rotation times obtained from the ILAN simulation process are not significantly different from those computed from Sethi and Saydam's analytical model.

7.3 Verification of Higher Layers

Suppose that the stack delays associated with transmitting are very much greater than the token bus delays so that the token bus delays can be neglected. For a simple case where a message is transmitted form one origination station on one sub-network through a bridge to another destination station on another sub-network, if the stack delay associated with transmitting is greater than or equal to the stack delay associated with receiving, then the queues in the reception stack of the destination station can be neglected. This problem is similar to the example defined in Chapter 6. For exponential time between message arrivals and constant stack delays, the time waiting to be serviced by the transmission stack can be approximated by a simple M/D/1 queuing model. An M/D/1 queuing model is a model that has a Poisson arrival process, a constant service time and one server. The service time is the time to transmit data from layer 7 of the origination station to layer 7 of the destination station. The number of servers in this case represents the number of ways in which data can be transmitted from one station to the other. The mean transmission delay of a message is defined as the total elapse time from the instant that the message enters the origination station ready to be transmitted to another station to the time it is received at layer 7 of the destination station. Thus the mean transmission delay, denoted by T, can be approximated by the following equation:

$$T = O72Q + O72 + B27 + B72 + D27 \qquad (7\text{–}16)$$

where O72Q = mean waiting time in queue before entering transmission stack at the origination station;

O72 = mean time delay in transmission stack at the origination station;

B27 = mean time delay in reception stack at the bridge;

B72 = mean time delay in transmission stack at the bridge;

and D27 = mean time delay in reception stack at the destination station.

The average waiting time in the queue before entering the transmission stack at the origination station can be calculated by the following queuing equations:

$$\mu = \frac{1}{O72} \qquad (7\text{–}17)$$

$$\rho = \frac{\lambda}{\mu} \qquad (7\text{–}18)$$

$$O72Q = \frac{\rho^2}{2\lambda(1-\rho)}. \qquad (7\text{–}19)$$

The stack delays associated with transmitting and receiving were assumed to be 145000 microseconds, each, for the origination station as well as for the destination station. A large value was used to assure that the stacks delays were much greater than the token bus delay. Stacks delays for the bridge were assumed to be 48300 microseconds or one third of the stacks delays for other stations. Table 7-6 shows results of simulation run number 1 and run number 2 for message arrival rates ranging from 2.778 to 6.667 arrivals per second, corresponding to utilizations from 0.4 to 0.85, according to Equation (7–17). For each simulation run, different random seeds were used. The simulation results in Table 7-6 include the mean transmission delay,

Table 7-6. Simulated Mean Transmission Delays vs. Message Arrival Rates

λ (arrival /sec.)	T Run 1 (μ sec.)	S_1 Run 1	Run 1 # of Observations n_1	T Run 2 (μ sec.)	S_2 Run 2	Run 2 # of Observations n_2
2.778	432180.1	76300.0	1380	435973.2	93000.	1400
3.448	459876.0	122000.0	1700	456764.1	104000.	1700
4.167	494398.3	139000.0	2110	503514.4	144000.	2010
5.000	592945.8	287000.0	2540	583242.4	192000.	2480
5.556	700582.3	358000.0	2800	678994.1	325000.	2830
5.882	721496.3	337000.0	2870	792557.2	575000.	2890
6.250	946004.5	520000.0	3080	1125097.0	744000.	3090
6.452	1384720.6	948000.0	3160	1414379.7	1400000.	3230
6.667	2430363.8	1570000.0	3340	2474348.9	2380000.	3340

standard deviation of transmission delay (S_i) and the number of observations (n_i). Each simulation run was stopped after 5.E8 microseconds. The simulation time ranged from 1 minute and 4 seconds to 2 minutes and 49 seconds on a 16 Mhz 386 Compaq.

Table 7-7 shows the mean transmission delay, the pooled variance of the mean transmission delay calculated from equation (7–3), the two random normal deviates and the two message transmission delay samples for each message arrival rate. Since the transmission delay is a random variable, from the Central Limit Theorem, the mean transmission delay is a random variable that is normally distributed, and the mean waiting time at transmission stack is calculated using Equations (7–17), (7–18) and (7–19). The mean transmission delay and its variance are computed from Equations (7–16) and (7–3), respectively. Assuming that the real system behaves as predicted by the queuing model, two samples of the mean transmission delay can be drawn from the normal distribution using random normal deviates and Equation (7–4).

Figure 7-2 shows the mean transmission delays computed by the two simulation runs and the mean transmission delay predicted by queuing theory. The mean transmission delay from the two simulation runs matches the message transmission delay calculated analytically from queuing theory.

The two factors in this experiment are (1) the methods of calculating the mean transmission delay and (2) the message arrival rates. Both factors were specifically chosen before performing the experiment. Therefore this experiment is a two-way fixed effects classification model similar to the experiment in Section 7.2. The same hypothesis and analysis of variance as the previous section is applied to this experiment. Table 7-8 shows the sample message transmission delays to be tested and

Table 7-7. Queuing Sample Mean Transmission Delays vs. Arrival Rates

λ (arrivals per sec.)	Random Normal Deviate 1	Random Normal Deviate 2	S^2_T	Mean Time in the System	T Sample 1 (microsec.)	T Sample 2 (microsec.)
2.778	1.102	0.148	10425751	435495	439054	435973
3.448	2.372	-0.145	15117647	459100	468323	458536
4.168	0.104	1.419	19439918	497257	497716	503514
5.000	0.069	0.797	47726779	577736	578213	583242
5.556	-0.393	-0.874	83010832	686957	683377	678994
5.882	0.125	-1.170	154497671	807100	808654	792557
6.250	-1.091	2.304	267227619	1087433	1069599	1125097
6.452	-0.961	-0.783	898490836	1437850	1409044	1414380
6.667	0.487	-0.299	2433922155	2489100	2513126	2474349

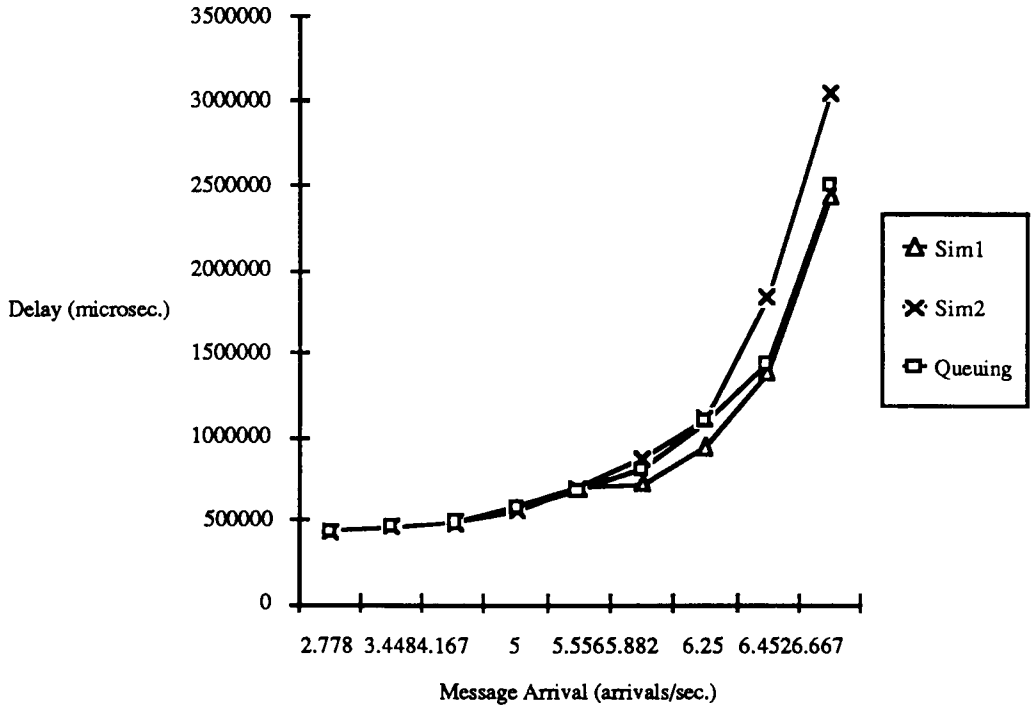


Figure 7-2. Mean Transmission Delays vs. Message Arrival Rates

Table 7-8. Sample of Mean Transmission Delays and Their Totals

	Method b_1: Determine Total Transmission delay Using ILAN Simulation Technique			Method b_2: Determine Total Transmission delay Using Queuing Theory Approach			
λ (arrival /sec.)	T_{i11} Sample 1 ($10^3\mu$. s.)	T_{i12} Sample 2 ($10^3\mu$ s.)	$T_{i1.}$	T_{i21} Sample 1 ($10^3\mu$ s.)	T_{i22} Sample 2 ($10^3\mu$ s.)	$T_{i2.}$	$T_{i..}$
2.778	432.18	440.17	872.35	439.05	435.97	875.03	1747.38
3.448	459.88	458.27	918.14	468.32	458.54	926.86	1845.00
4.168	494.40	487.26	981.65	497.72	503.51	1001.23	1982.89
5.000	592.95	555.17	1148.12	578.21	583.24	1161.46	2309.57
5.556	700.58	691.63	1392.21	683.38	678.99	1362.37	2754.58
5.882	721.50	874.74	1596.24	808.65	792.56	1601.21	3197.45
6.250	946.00	1101.56	2047.56	1069.60	1125.10	2194.70	4242.26
6.452	1384.72	1836.31	3221.03	1409.04	1414.38	2823.42	6044.46
6.667	2430.36	3051.36	5481.72	2513.13	2474.35	4987.47	10469.19
$T_{.j.}$	17659.03			16933.75			34592.78

their totals. The observations may be described by the model in Equation (7–5). Table 7-9 shows the ANOVA table for the mean transmission delay.

At the 10% level of significance, the value of $F_{(0.1,8,18)}$ is found to be 2.04. From Table 7-9 the value $F_{critical}$ for mean effect of the message arrival rates is 113.0521. Since $F_{(0.1,8,18)}$ is less than $F_{critical}$, H'_0 is rejected, and it can be concluded that message arrival rates affect the message transmission delay. At the same level of significance, $F_{(0.1,1,18)} = 3.01$ is greater than the critical value of F for the mean effect of determination methods of 0.8169. Thus H''_0 cannot be rejected and it can be concluded that there is no significant difference between the methods used in predicting the message transmission delay. Furthermore, $F_{(0.1,8,18)} = 2.04$ is greater than the F statistic for the interaction between message arrival rates and the different determination methods of 0.6414 (Table 7-9). Hence, H'''_0 is rejected, and it can be concluded that there is no interaction between different arrival rates and methods of determining the message transmission delay.

Table 7-9. ANOVA Table for Mean Transmission Delays

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	Computed F
Message Arrival Rates	1.6177E+13	8	2.0221E+12	113.0521
Determination Methods	1.4612E+10	1	1.4612E+10	0.8169
Interaction	9.1782E+10	8	1.1473E+10	0.6414
Error	3.2196E+11	18	1.7887E+10	
Total	1.6606E+13			

7.4 Validation of Simulation Procedure

Several experiments were conducted by Chung [14] in determining throughput between two RS232-C devices connected to seven-layer MAP multiplexers under no load without going through a MAP network (Figure 7-3) and with load on a MAP network (Figure 7-4). Equation (7–20) shows the relationship between the total time delay in transmitting a message, the throughput and the message size:

Total transmission delay = Message size / Throughput. (7–20)

The throughput for transmitting a message size of 10240 bits at 9600 baud (bits/sec.) from PCa to PCb (Figure 7-3) with no load on the network was 5200 bits/second. From Equation (7–20), the delay was 1.969231 seconds. When there was a load on the network, the throughput was 1% less than the no load condition, or 4680 bits/second, which corresponded to the delay of 2.188034 seconds. The load on the network was created by transmitting 10240 bits of message at 9600 baud from PC3, PC5, PC7 and PC9 to PC4, PC6, PC8 and PC10, respectively. As soon as these messages are received by the destination PCs, they are transmitted back to the origination PCs. For example, when PC5 received a message, it transmitted back the same message to PC6 and vice versa. During the experiments, two other nodes were also active in addition to the two multiplexers on the MAP network. A network analyzer node and a network manager node were connected on the network but did not transmit any messages.

Since ILAN represents devices connected to a multiplexer as a single node and allows only one message arrival distribution per node, the exact representation of the

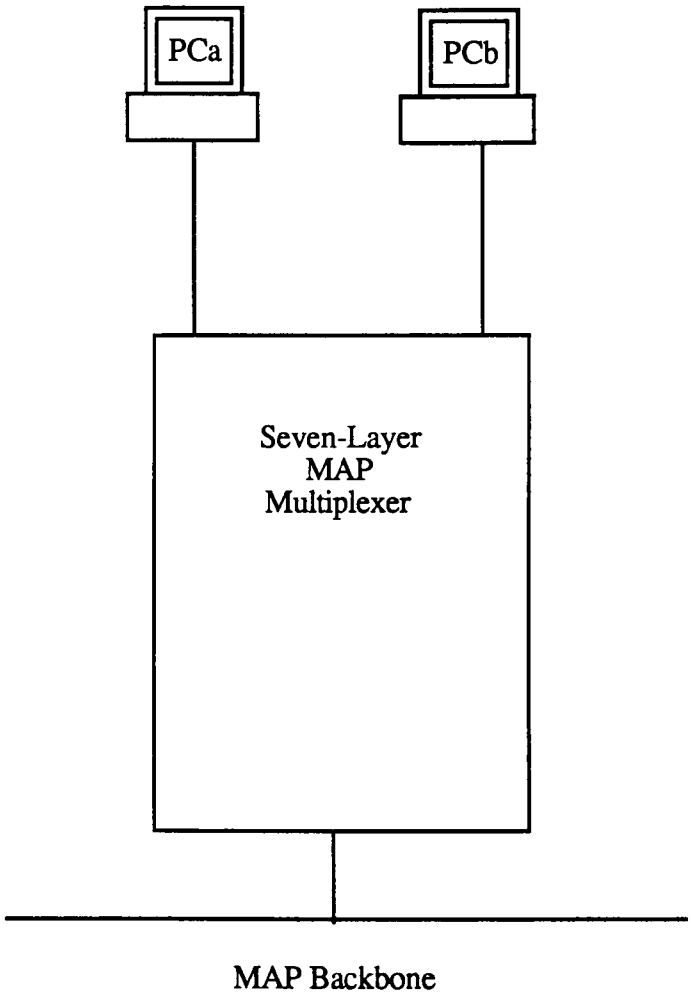


Figure 7-3. Experiment without Load

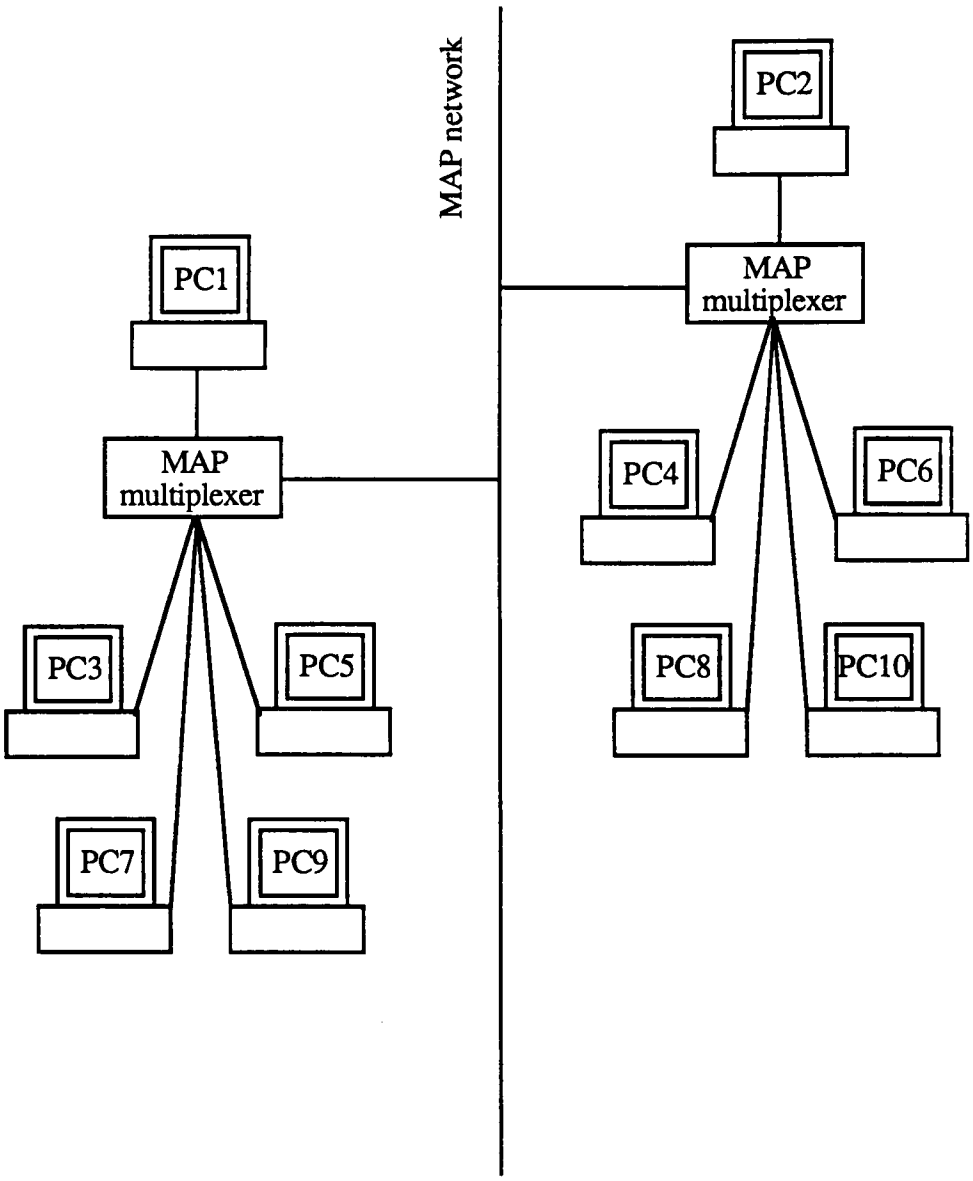


Figure 7-4. Experiment with Load

above experiment with a load is not possible. However, an ILAN simulation model that closely approximates the above experiment can be constructed. The number of nodes in the simulation model is four, which is the same as in the experiment. Instead of not transmitting any messages, the two nodes that are not multiplexers are used to initiate the messages that are transmitted to the multiplexers. Thus, a message of size 10240 bits is created by each of the four nodes on the network in the simulation model instead of the four PCs on one node. Once these messages arrive at the two multiplexer nodes, they are transmitted back and forth between the two multiplexer nodes and are not transmitted back to the two non-multiplexer nodes.

Figure 7-5 represents the four messages generated at the four nodes. Node numbers 1 and 2 are multiplexer nodes. Node numbers 3 and 4 are non-multiplexer nodes. At the beginning of the simulation run, message number "n" is initiated by node number "n." For example, message number 1 is initiated by node number 1. It is transmitted to node number 2 one hundred percent of the time. Once it arrives at node number 2, message number 2 is spawned. Message number 2 is transmitted back to node number 1 one hundred percent of the time. Upon reaching node number 1, message number 1 is spawned and so on. Messages number 3 and 4 are transmitted to node number 1 one hundred percent of the time. When it reaches node number 1, message number 1 is spawned and is transmitted to node number 2 and so on. Thus messages number 3 and 4 are transmitted only once during the entire simulation run to initiate the load that is transmitted back and forth between the two multiplexer stations. It is appropriate to model the message transmissions as file transfers because the lengths of these messages are known.

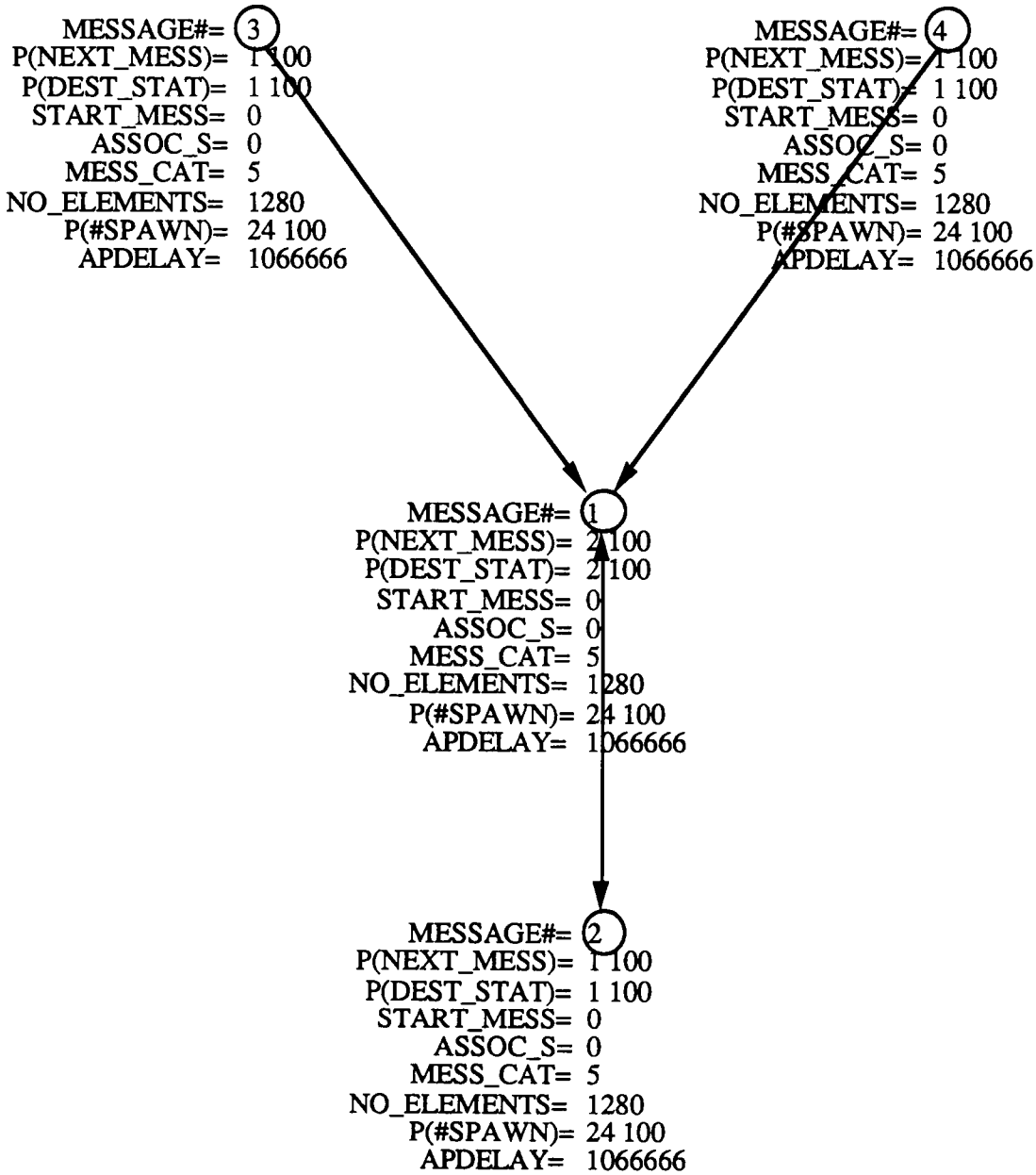


Figure 7-5. Simulation Message Sequences

The stack time delays for transmitting and receiving when there was no load on the network can be approximated from the experiment. In addition to the stack delay, there is a time delay in communicating between the RS232-C device and the MAP multiplexer existed. This RS232-C time delay is computed from Equation (7–21):

RS232-C time delay = Message size / Baud Rate. (7–21)

Thus the RS232-C time delay for a message size of 10240 bits was 1.0666667 seconds. This delay is simulated as an application delay. Messages are broken up into a maximum frame size of 512 bytes before being transmitted across a network. In order to transmit 10240 bits, or 1280 bytes, three transmissions are needed. By the time the last frame leaves the origination node, the first two frames will have been received by the destination node. Equation (7–22) shows the time elapsed before the third frame leaves the origination node:

T3 = RS232-C time delay + (3 * stack delays) (7–22)

where T3 = Elapsed time before the third frame left the origination node.

To complete the transmission, the last frame is received by the destination node. Therefore, the total transmission time is

T3 + stack delay.

Thus delay in the stack per frame is approximated by dividing the difference between the total time delay in transmitting a message under no load and the RS232-C time delay by 4 (Equation (7–23)):

Stack delay = (Total transmitting time - RS232-C delay) / 4.0 . (7–23)

Thus the stack delay for the simulation model is

(1.969231 - 1.066667) / 4 = 0.225641 seconds.

Using the application delay and the stack delay derived from the no load experiment, the simulation of the model with a load was constructed. The average time delay (T) for transmitting 10240 bits of 46 messages from node 1 and 48 messages from node 2 was 2104818 microseconds. The standard deviations for messages transmitted from nodes 1 and 2 were 333333 and 191000 microseconds^2, respectively. The pooled variance for mean message transmission times (S^2_T) calculated from Equation (7–3) was 3.17621E9 microseconds^2. Thus, the pooled standard deviation for mean message transmission times (S_T) was 56357.9 microseconds^2. The actual mean transmitting delay (μ) was assumed to be the value that was determined from the experiment, which was 2188034 microseconds. Using the Central Limit Theorem, it can be concluded that the average time delay for transmitting messages followed a normal distribution. The statistical hypotheses

$H_0: \mu = T$

$H_1: \mu \neq T$

could be tested by calculating the z transformation and comparing it to the critical z value ($z_{\alpha/2}$). The z transformation can be calculated as follows:

$z = (\mu - T) / S_T = 2188034 - 2104818) / 5637.9 = 1.48$.

For a two tails test at $\alpha = 0.1$ level of significance, $z_{\alpha/2} = 1.65$. Since the calculated z was less than $z_{\alpha/2}$, there was no reason to reject H_0. Therefore, it can be concluded that the experimental and the simulated mean message transmission times were the same.

It is impossible to validate any simulation model for every applicable scenario [58, 88]. The number of applications for a particular simulation model is usually enormous. Indeed, the main reason for building a simulation model is to use it to predict situations that had not been encountered before. The best that can be done is to validate the model in certain ranges and extrapolate the results of the validation to other ranges. "The success of validation establishes a basis for confidence in the results that the model generates under new conditions" [28]. However, simulation results that have not been validated should be carefully checked and analyzed to make sure that they behave in reasonable manners.

In this chapter, comparisons of simulation results to analytical results of S&S and M/D/1 as well as experimental results have been presented. It has been demonstrated that the ILAN simulation results are not significantly different from analytical results in computing the mean token rotation times. This section also shows that the message transmission delay obtained from the ILAN simulation model is not significantly different from that computed analytically from M/D/1 queuing theory. Finally, there is no significant difference between the mean transmission times

computed by the ILAN simulation program and those measured by the experiment. In conclusion, ILAN simulation results are well behaved within the various tested ranges. Sufficient evidence has been provided to illustrate that the ILAN simulation program was performing correctly.

CHAPTER 8

SIMULATION OF AN AUTOMOTIVE PLANT

8.1 System Description

To show the capability and versatility of ILAN software, analysis of an actual moderately sized MAP network is presented in this chapter. The network under investigation is the General Assembly Network at Saturn Corporation's Spring Hill, TN plant (Figure 1-1). Because this problem is too large to be modeled on a DOS version of ILAN, an OS/2 version of ILAN is used to analyze this network.

Although ILAN simulation software is capable of simulating large networks, the performance of the simulation program decreases as the size of the problem increases. It is desirable to divide a large network into smaller networks and analyze each of them separately. Depending on the inter-sub-network traffic, different sub-networks can be grouped together. If a sub-network has little or no communications with other sub-networks, then it can be analyzed separately. Several sub-networks that have heavy traffic between each other but little traffic elsewhere can be analyzed together. If heavy traffic between each sub-network exists, then it may be feasible to simplify the problem by reducing the number of nodes on the network. For example, nodes with little or no traffic may be eliminated. Several nodes on the same sub-network that do not communicate with each other may be combined into a single node.

The General Assembly Network consists of 59 PLCs (nodes number 1 through 59), 9 Microvaxes (nodes number 60 through 68), and 12 PC's (nodes number 68

through 80) on a 10 Mb/s broadband cable. Microvax node number 67 communicates exclusively with the 12 PCs. It receives a file of 1024 bytes every 270 seconds from each of 5 of the PCs (nodes number 69 through 73). It transmits a file of 1024 bytes to each of the remaining 7 PCs once each hour. Microvax node number 68 has no MAP traffic requirements at this time.

No unsolicited messages are generated, so all data are polled by the Microvaxes. Traffic for Microvax nodes number 60, 61 and 62 is tabulated in Figure 8-1. Figure 8-2 shows traffic for the remaining Microvaxes. Each area in these figures defines a number of PLCs which communicate with a single Microvax. In Figure 8-1, the first area ("Cockpit Area") has one Microvax (node number 60) and 3 PLCs (nodes number 1, 13 and 43), the second area ("Hardware Area") has one Microvax (node number 61) and 6 PLCs (nodes number 2, 6, 35, 40, 44, and 62), etc. The "Triggers" column represents the number of data elements being polled ("read requested") by the Microvax. The"Triggers Scan Time" column signifies the time between which two consecutive data requests are transmitted from the Microvax. The "Misc. Info. Regs." specifies the number of additional miscellaneous registers of data polled. The "Misc. Info. Scan Time" represents the time between two consecutive miscellaneous registers read requests that are issued.

The interpretation of the traffic flows for Figure 8-1 and Figure 8-2 is illustrated by explaining the traffic for the first PLC in the Cockpit Area of Figure 8-1. A read request for 44 data elements is transmitted from Microvax node number 60 to PLC node number 1 every 2 seconds. After the PLC node number 1 receives the read request, it transmits a "read response" for the 44 data elements back to the

Node Number	Triggers	Trigger Scan Time (sec.)	Misc. Info. Regs.	Misc. Info. Scan Time
Cockpit Area: Microvax Node Number 60				
1	44	2	12	6
13	17	2	8	6
43	8	2	8	6
Cockpit Subtotal =	69		28	
Hardware Area: Microvax Node Number 61				
2	72	3	12	6
6	40	2	12	6
35	20	2	8	6
40	15	2	8	6
44	17	2	8	6
62	20	2	15	6
Hardware Subtotal =	184		63	
Chassis Area: Microvax Node Number 62				
8	22	2	6	6
9	10	2	6	6
10	10	2	6	6
11	35	2	8	6
12	10	2	7	6
24	13	2	8	6
25	13	2	8	6
26	8	2	7	6
27	8	2	7	6
28	8	2	7	6
29	8	2	7	6
30	5	2	7	6
31	5	2	7	6
32	5	2	7	6
33	5	2	7	6
34	14	2	7	6
36	30	2	8	6
39	10	2	8	6
45	10	2	8	6
53	20	2	7	6
54	20	2	7	6
Chassis Area Subtotal =	269		150	

Figure 8-1. Nodes Number 60-62 Read Requests and Their Scan Times

Node Number	Triggers	Trigger Scan Time (sec.)	Misc. Info. Regs.	Misc. Info. Scan Time (sec.)
Trim Area: Microvax Node Number 63				
3	60	3	12	6
19	17	2	8	6
20	16	2	8	6
21	17	2	8	6
22	16	2	8	6
23	16	2	8	6
37	51	2	14	6
41	25	2	8	6
42	15	2	8	6
46	17	2	8	6
Trim Subtotal =	250		90	
Door Area: Microvax Node Number 64				
5	45	2	12	6
47	11	2	8	6
Door Subtotal =	56		20	
Panel Area: Microvax Node Number 65				
4	60	3	12	6
7	65	3	20	6
48	11	2	8	6
Panel Subtotal =	136		40	
Final Area: Microvax Node Number 66				
14	9	2	7	6
15	9	2	7	6
16	9	2	7	6
17	9	2	7	6
18	9	2	7	6
38	10	2	8	6
49	7	2	8	6
50	19	2	7	6
51	19	2	7	6
52	19	2	7	6
55	5	2	7	6
56	13	2	7	6
57	5	2	7	6
58	5	2	7	6
Final Area Subtotal =	147		100	

Figure 8-2. Nodes Number 63-66 Read Requests and Their Scan Times

Microvax that requested it. In addition, the Microvax also transmits a read request for 12 miscellaneous registers of data to the same PLC. Upon receiving the read request, the PLC transmits a read response for the 12 data elements back to the Microvax.

Several other input variables must be determined before executing ILAN. These input parameters include the time between any external message arrival at each node, the message number probabilities for a given external message arrival and the destination probabilities for a given message. A message in ILAN can be a file transfer, read request, read response, write request or write response. These values were computed for the General Assembly MAP network and are shown in Figures 8-3 and 8-4. The construction of the traffic flow data in Figures 8-3 and 8-4 is explained in the next section.

Node Number	Request Message Number	Response Message Number	Number of Data Elements	Single Send (sec.)	Unique Requests	Unique Responses	Number of Same Messages	Message Send (sec)	% Message	% Request Destination
Cockpit Area: Microvax node number 60 sends a read request every 0.5 sec.										
01	1	2	44	2	1	2	1	2.000	25.000	100.000
	3	4	12	6	3	4	1	6.000	8.333	100.000
13	5	6	17	2	5	6	1	2.000	25.000	100.000
	7	8	8	6	7	8	3	1.200	41.667	20.000
43	7	8	8	2						80.000
	7	8	8	6						
									100.000	
Hardware Area: Microvax node number 61 sends a read request every 0.261 sec.										
02	9	10	72	3	9	10	1	3.000	8.696	100.000
	11	12	12	6	11	12	2	3.000	8.696	50.000
06	13	14	40	2	12	14	1	2.000	13.043	100.000
	11	12	12	6						50.000
35	15	16	20	2	15	16	2	1.000	26.087	50.000
	17	18	8	6	17	18	3	2.000	13.043	33.333

Figure 8-3. Traffic Flows for Read Requests and Responses

Node Number	Request Message Number	Response Message Number	Number of Data Elements	Single Send (sec.)	Unique Requests	Unique Responses	Number of Same Messages	Message Send (sec)	% Message	% Request Destination
40	19	20	15	2	19	20	2	1.500	17.392	75.000
	17	18	8	6						33.333
44	21	22	17	2	21	22	1	2.000	13.043	100.000
	17	18	8	6						33.333
59	15	16	20	2						50.000
	19	20	15	6						
25.000									100.000	
Chassis Area: Microvax node number 62 sends a read request every 0.071 sec.										
08	23	24	22	2	23	24	1	2.000	3.571	100.000
	25	26	6	6	25	26	3	2.000	3.571	33.333
09	27	28	10	2	27	28	5	0.400	17.857	20.000
	25	26	6	6						33.333
10	27	28	10	2						20.000
	25	26	6	6						33.333
11	29	30	35	2	29	30	1	2.000	3.571	100.000
	31	32	8	6	31	32	10	0.333	21.430	5.556
12	27	28	10	2						20.000
	33	34	7	6	33	34	12	0.500	14.286	8.333
24	35	36	13	2	35	36	6	0.333	21.430	16.667
	31	32	8	6						5.556
25	35	36	13	2						16.667
	31	32	8	6						5.556
26	31	32	8	2						16.667
	33	34	7	6						8.333
27	31	32	8	2						16.667
	33	34	7	6						8.333
28	31	32	8	2						16.667
	33	34	7	6						8.333
29	31	32	8	2						16.667
	33	34	7	6						8.333
30	35	36	5	2						16.667
	33	34	7	6						8.333
31	35	36	5	2						16.667
	33	34	7	6						8.333
32	35	36	5	2						16.667
	33	34	7	6						8.333
33	35	36	5	2						16.667
	33	34	7	6						8.333
34	37	38	14	2	37	38	1	2.000	3.571	100.000
	33	34	7	6						8.333
36	39	40	30	2	39	40	1	2.000	3.571	100.000
	31	32	8	6						5.556

Figure 8-3. (Continued)

Node Number	Request Message Number	Response Mesage Number	Number of Data Elements	Single Send (sec.)	Unique Requests	Unique Responses	Number of Same Messages	Message Send (sec)	% Message	% Request Destination
39	27	28	10	2						20.000
	31	32	8	6						5.556
45	27	28	10	2						20.000
	31	32	8	6						5.556
53	41	42	20	2	41	42	2	1.000	7.142	50.000
	33	34	7	6						8.333
54	41	42	20	2						50.000
	33	34	7	6						8.333
									100.000	
Trim Area: Microvax node number 63 sends a read request every 0.154 sec.										
03	43	44	60	3	43	44	1	3.000	5.128	100.000
	45	46	12	6	45	46	1	6.000	2.564	100.000
19	47	48	17	2	47	48	3	0.667	23.077	33.333
	49	50	8	6	49	50	8	0.750	20.513	12.500
20	51	52	16	2	51	52	5	0.400	38.462	20.000
	49	50	8	6						12.500
21	47	48	17	2						33.333
	49	50	8	6						12.500
22	51	52	16	2						20.000
	49	50	8	6						12.500
23	51	52	16	2						20.000
	49	50	8	6						12.500
37	51	52	51	2						20.000
	53	54	14	6	53	54	1	6	2.564	100.000
41	55	56	25	2	55	56	1	2	7.692	100.000
	49	50	8	6						12.500
42	51	52	15	2						20.000
	49	50	8	6						12.500
46	47	48	17	2						33.333
	49	50	8	6						12.500
									100.000	
Door Area: Microvax node number 64 sends a read request every 0.75 sec.										
05	57	58	45	2	57	58	1	2.000	37.500	100.000
	59	60	12	6	59	60	1	6.000	12.500	100.000
47	61	62	11	2	61	62	1	2.000	37.500	100.000
	63	64	8	6	63	64	1	6.000	12.500	100.000
									100.000	

Figure 8-3. (Continued)

Node Number	Request Message Number	Response Mesage Number	Number of Data Elements	Single Send (sec.)	Unique Requests	Unique Responses	Number of Same Messages	Message Send (sec)	% Message	% Request Destination
Panel Area: Microvax node number 65 sends a read request every 0.6 sec.										
04	65	66	60	3	65	66	1	3.000	20.000	100.000
	67	68	12	6	67	68	1	6.000	10.000	100.000
07	69	70	65	3	69	70	1	3.000	20.000	100.000
	71	72	20	6	71	72	1	6.000	10.000	100.000
48	73	74	11	2	73	74	1	2.000	30.000	100.000
	75	76	8	6	75	76	1	6.000	10.000	100.000
									100.000	
Final Area: Microvax node number 66 sends a read request every 0.107 sec.										
14	77	78	9	2	77	78	5	0.400	26.786	20.000
	79	80	7	6	79	80	13	0.400	26.786	6.667
15	77	78	9	2						20.000
	79	80	7	6						6.667
16	77	78	9	2						20.000
	79	80	7	6						6.667
17	77	78	9	2						20.000
	79	80	7	6						6.667
18	77	78	9	2						20.000
	79	80	7	6						6.667
38	81	82	10	2	81	82	1	2	5.357	100.000
	83	84	8	6	83	84	2	3	3.572	50.000
49	79	80	7	2						20.000
	83	84	8	6						50.000
50	85	86	19	2	85	86	3	0.667	16.071	33.333
	79	80	7	6						6.667
51	85	86	19	2						33.333
	79	80	7	6						6.667
52	85	86	19	2						33.333
	79	80	7	6						6.666
55	87	88	5	2	87	88	3	0.667	16.071	33.333
	79	80	7	6						6.667
56	87	90	15	2	89	90	1	2	5.357	100.000
	79	80	7	6						6.667
57	87	88	5	2						33.333
	79	80	7	6						6.667
58	87	88	5	2						33.333
	79	80	7	6						6.667
									100.000	

Figure 8-3. (Continued)

Destination Node Number	File Message Number	Single Send (sec.)	Unique Message Number	Number of Same Messages	% Message	% Destination
Microvax node number 67 sends a file every 515 sec.						
74	91	36000	91	7	100.000	14.280
75	91	36000				14.280
76	91	36000				14.280
77	91	36000				14.280
78	91	36000				14.280
79	91	36000				14.280
80	91	36000				14.280
Microvax node number 68 has no MAP traffic						
PC node number 69 sends a file every 270 sec.						
67	92	270	92	1	100.000	100.000
PC node number 70 sends a file every 270 sec.						
67	92	270	92	1	100.000	100.000
PC node number 71 sends a file every 270 sec.						
67	92	270	92	1	100.000	100.000
PC node number 72 sends a file every 270 sec.						
67	92	270	92	1	100.000	100.000
PC node number 73 sends a file every 270 sec.						
67	92	270	92	1	100.000	100.000

Figure 8-4. File Transfer Traffic Flows

8.2 Construction of Traffic Flow Data

The time between any external message arrivals can be calculated by taking the reciprocal of the sum of the reciprocal of the time between each individual message arrival. For the Cockpit Area of Figure 8-1, there are six individual message arrivals. These message arrivals are read requests. Thus the time between any external message arrival is

$1/(1/2+1/2+1/2+1/6+1/6+1/6) = 0.5$ seconds.

In other words, Microvax node number 60 has a message to be transmitted (read request) every 0.5 seconds.

In Figure 8-3, each message (read requests, read responses and file transfers) is numbered. Messages are considered the same if they have the same length, and the same list of possible destination nodes. From Figure 8-1, eight data elements are requested for miscellaneous information registers from PLC nodes number 13 and 43. Eight data elements are also requested for the triggers from node number 43. The three read requests can be modeled using a single message type with node numbers 13 and 43 as the list of possible destination nodes for the message. Similarly, the three read responses for these read requests have the same length and the same list of destination nodes. In fact, these read responses are transmitted to a single node, Microvax node number 60. Consequently, these read requests can be represented by using a single message type. The Cockpit Area in Figure 8-3 shows the three read requests listed as message number 7 and their corresponding read responses listed as message number 8.

Since the message may be transmitted either to PLC node number 13 or 43, the percentages of time for transmitting to these nodes are needed. These percentages are calculated by dividing the arrival rate of the message that is transmitted to a given node by the total arrival rate to be transmitted to all of its destination nodes and multiplying by 100. For example, arrival rate for message number 7 to be transmitted to node number 13 is 1/6 arrivals per second. The arrival rate of message number 7 to be transmitted to node number 43 is 1/2 + 1/6, or 2/3 arrivals per second. The total arrival rate to be transmitted to all of its destination nodes is 1/6 + 2/3, or 5/6 arrivals per second. Thus the percent of time that message number 7 is transmitted to node 13 is (1/6)/(5/6)(100) or 20%. The percent of time that message number 7 is transmitted to node 43 is (2/3)/(5/6)(100) or 80%. These percentages are listed in Figure 8-3 under the last column, with the heading of "% Request Destination." Note that the read response destinations percentages are not listed in Figures 8-3 because they are transmitted to the area Microvax 100% of the time.

The "Single Send" column in Figures 8-3 signifies the time between transmitting each individual message, which is the combination of the columns "Trigger Scan Time" and "Misc. Info. Scan Time" from Figures 8-1 and 8-2. The "Unique Requests" and "Unique Responses" columns denote unique request and response message numbers. The "Number of Same Messages" column lists the number of times that these messages were listed under the "Request Message Number" column. For the Cockpit Area, only messages number 7 and 8 had multiple listings. They were listed three times. Thus 3 is shown under the "Number of Same Messages" column for these messages. The "Message Send" column represents the total time between transmitting these messages. For a message that has only one "Number of Same Messages," this

number is the same as the number in "Single Send" column. For a message that has multiple number of same messages, this number can be calculated by taking the reciprocal of the sum of the reciprocal of the time between transmitting these individual messages. For example, Microvax node number 60 transmits message number 7 (which is a read request) to PLC node number 13 once every 6 seconds, transmits message number 7 to PLC node number 43 once every 2 seconds and transmits another message number 7 to PLC node number 43 once every 6 seconds. Thus the total time between transmitting message number 7 is 1/(1/6+1/2+1/6) or 1.2 seconds. This value is listed under the "Message Send" column for message number 7.

The next parameter to be established is the percentage of time that each read request is to be transmitted. This number can be computed by dividing the total time between transmitting any messages from a given node by the total time between transmitting messages of a given type from the same node and multiplying it by 100. It was established that the total time between transmitting a read request from Microvax node number 60 is 0.5 seconds. Furthermore, the time between transmitting messages number 7 from the same Microvax was shown to be 1.2 seconds (from "Message Send" column of Figure 8-3). Thus the percentage of time that message number 7 is transmitted is 0.5(100)/1.2 or 41.667%. The column "% Message" in Figure 8-3 represents the percentage of time that a given message is transmitted from a particular Microvax.

Figure 8-4 shows the file transfer for the General Assembly network. The difference between Figure 8-3 and Figure 8-4 is that Figure 8-4 describes the file transfers instead of read requests and read responses. The construction of Figure 8-4 is similar to that of Figure 8-3. It should be noted that there are only two messages in

Figure 8-4. These are messages number 91 and 92. Although both messages have the same length of 1024 bytes, the difference between them is that they have different sets of destination nodes. Message number 91 transmits to PC nodes number 74 through 80, whereas message number 92 transmits to Microvax node number 67.

8.3 Input to ILAN

The hardware description of General Assembly in ILAN is straightforward. There are 80 nodes connected on a single 10 Mb/s broadband cable. Only nodes number 60 through 67 (Microvaxes transmitting read requests or file transfers) and 69 through 73 (PCs transmitting files) have external message arrivals. The stack delays associated with transmitting and receiving for each node are assumed to have the same value of 36500 microseconds. This value is approximated by the experiment conducted by Electronic Data Systems Corporation (EDS) [3]. A typical input for nodes with external message arrival is illustrated in Figure 8-5. Here Microvax node number 60 is constructed. The values of time between message arrivals, the unique message numbers and their percentages are found from Figure 8-3.

All of the messages in the General Assembly Network are read requests and read responses with the exception of message numbers 91 and 92 which are file transfers. A typical dialog for a read request message followed by a read response message definition in ILAN is found in Figure 8-6. Here, messages number 7 and 8 are defined. Message number 7 is a read request and message number 8 is a read response. The destination nodes and their percentages for message number 7 can be found in Figure 8-3 under the "% Dest." column. The number of data elements to read and write is listed in Figure 8-3 under the heading "Data Pts."

```
Command:NODE-1N
Station number for this station :60

Would you like to specify default reception stack delays
for messages that are transmitted from this node (Y/N)?
<N>:Y

Enter last message no. that uses default reception stack
delays at this node [message 1 through this last message
no. uses this default delay]:8

Enter the default reception stack delay in microseconds
<36500>: 36500

Would you like to specify individual stack delay for
message reception [these individual delays over ride any
default previously set] (Y/N) <N>:N

Would you like to specify default transmission stack delays
for messages that are transmitted from this node (Y/N)?
<N>:Y

Enter last message no. that uses default transmission stack
delays at this node [message 1 through this last message
no. uses this default delay]:8

Enter the default transmission stack delay in microseconds
<36500>: 36500

Would you like to specify individual stack delay for
message transmission [these individual delays over ride any
default previously set] (Y/N) <N>:N

Find distances from drawing, User input or assume 0 (D, U,
or 0):0
How many sub-networks are connected to this node:1

Enter sub-network number of a sub-network connected to this
node:1
```

Figure 8-5. Construction of Node Number 60

```
Messages arrive externally to this station ? (Y,N):Y

Do all external messages arrive Once, Constantly or
Randomly (O/C/R) <C>:C

What is the constant time between message arrivals in mic.
sec.:0.5E6

How many different messages to be transmitted correspond to
external arrival?:4

Enter a message number that may be transmitted as a result
of an external arrival:1

Enter % of the time this message is transmitted when an
external message arrives:25

Enter a message number that may be transmitted as a result
of an external arrival:3

Enter % of the time this message is transmitted when an
external message arrives:8.333

Enter a message number that may be transmitted as a result
of an external arrival:5

Enter % of the time this message is transmitted when an
external message arrives:25

Enter a message number that may be transmitted as a result
of an external arrival:7

Enter % of the time this message is transmitted when an
external message arrives:41.667

Command:DRAWMS

Current message is New message, Defined message or End
(N,D,E):N
```

Figure 8-5. (Continued)

```
Message number:7

How many destination nodes can this message possibly
transmit to ?:2

Enter a node number from the list of possible destination
nodes:13

What is the percent of time that this is the destination
node ?:20

Enter a node number from the list of possible destination
nodes:43

What is the percent of time that this is the destination
node ?:80

Start message # (if current message is a group message)
<0>:

Association status for this message <0>:

Press 1=write request, 2=write response, 3=read request,
4=read response,or 5=file transfer <5>:3

How many elements to read or write ? <1>:8

How many possible spawn values for this message (0 spawn is
1 value)? <1>:1

Enter a number to spawn from the list of possible spawn
values <0>:1

What is the % of time that this number is spawned from this
message?:100

How many different nodes can this message be transmitted
from ?:1
Enter a node number from the list of nodes that may
transmit this message:60
```

Figure 8-6. Definition of Message Numbers 7 and 8

```
Enter application delay (mic s) for this message
transmitted from this node ?:0
INSERT Block name (or ?):MESSAGE Insertion point:
:
Launch New message, Defined message or End (N,D,E):N

Message number:8

How many destination nodes can this message possibly
transmit to ?:1

Enter a node number from the list of possible destination
nodes:60

What is the percent of time that this is the destination
node ?:100

Start message # (if current message is a group message)
<0>:

Association status for this message <0>:

Press 1=write request, 2=write response, 3=read request,
4=read response,or 5=file transfer <5>:4

How many elements to read or write ? <1>:8

How many possible spawn values for this message (0 spawn is
1 value)? <1>:1

Enter a number to spawn from the list of possible spawn
values <0>:0

What is the % of time that this number is spawned from this
message?:100

How many different nodes can this message be transmitted
from ?:2

Enter a node number from the list of nodes that may
transmitted this message:13
```

Figure 8-6. (Continued)

```
Enter application delay (mic s) for this message
transmitted from this node ?:0

Enter a node number from the list of nodes that may
transmit this message:43

Enter application delay (mic s) for this message
transmitted from this node ?:0
INSERT Block name (or ?):MESSAGE Insertion point:
:
% of launching this message:100

Launch New message, Defined message or End (N,D,E):E
```

Figure 8-6. (Continued)

There are some minor differences between message descriptions for a file transfer and a read request. By pressing "5" to select a file transfer, the user is asked for the length of the file in bytes. For the General Assembly problem, all file transfers are 1024 bytes in length.

8.4 Simulation Results

The output of the ILAN simulation run for the General Assembly MAP network is listed in Appendix E. The simulation run was executed until + or - 0.1% of the mean transmission delay is within a 90% confidence interval of the mean transmission delay or until simulation time of .141E+10 microseconds is reached. The simulation run ended at time .141E+10 microseconds and took 1 hour to execute on a 25 MHz IBM PS/2 model 70 A121. A total of 115273 frames was transmitted on the network.

The traffic in the network is very light. The utilization of the network was found to be 0.013. This utilization is calculated as the total time that the data is transmitted divided by the total simulation time during the statistical collection period. The term "data" refers to frames that are not tokens. The average queue length for all nodes in the network waiting for the token is 0.0229. The mean queue lengths for frames waiting for the token at nodes that have data to transmit was found to range between .0001 at node number 67 to 0.4872 at node number 62. The results are as expected since node number 67 transmits messages most often while node number 62 transmits least often.

There is little or no queue for the transmission and reception stacks. The mean queue lengths for frames waiting to be serviced in the reception stack range from zero to 0.17. There is no queue detected for the transmission stack.

The transmission delay for messages transmitted from each node can be found in the simulation output. "Number of Observations" column lists the number of messages transmitted from these nodes. Note that only a few messages (files) were transmitted from nodes 69 through 73 and even fewer from node number 67. This is because the time between transmissions from these nodes was large compared with other nodes. No observation is recorded for node number 68 because it does not have any MAP traffic at this time. The transmission delay is defined as the time between the instant that the origination node has some data to transmit from layer 7 to the instant that the destination node received all of the data at layer 7. The data may be comprised of several frames. The transmission delay is increased for longer messages at the same network traffic intensity. From the simulation output, it can be seen that the times in the

system for file transfers (messages from nodes 67, 74 to 80) are longer than for other messages on the network.

In conclusion, the ILAN software adequately accommodates the General Assembly MAP network. The overall behavior of the network predicted by the simulation model can be explained logically above. This suggests that the simulation procedure is executing correctly. The results from the simulation indicate that the MAP network can handle much higher traffic loads.

CHAPTER 9

CONCLUSIONS

9.1 Graphical User Interface

The graphical user interface procedure of ILAN has been described. The construction of the graphical interface through AutoCAD has been demonstrated. Similar techniques can be applied in the construction of graphical interfaces for other simulation programs. Through the graphical user interface, the definition of the MAP network to be analyzed is easily constructed. The four stages of problem declaration, which consist of hardware, networks, messages and other parameters definitions, have been presented. Examples of these four phases of the problem-definitions have been implemented and illustrated in Chapters 5 and 6. The construction of the General Assembly MAP Network has been briefly demonstrated, showing that real systems can be effectively specified through the graphical user interface.

9.2 Simulation Procedure

The construction and implementation of ILAN simulation procedure have been described. The fast and efficient token passing codes have been explained. MAP networks described through the graphical user interface have been analyzed and validated. Validation results have established correct input to and output from the simulation procedure over various tested ranges. Furthermore, the fast token simulation methodology has also been validated.

The simulation of the General Assembly MAP network at GM's Saturn plant in Spring Hill, TN have been described. The results from the simulation run has been analyzed and shown to behave as expected. The problem description and simulation of the General Assembly problem establishes that real systems can be implemented using ILAN software.

9.3 Summary of Contributions

1. A powerful tool for analyzing any MAP network has been developed. The ILAN software system is capable of modeling multiple MAP sub-networks interconnected by bridges with different message arrival rates and arrival distributions at each node.
2. The implementation of stack delays associated with transmitting and receiving is a viable way of modeling delays in the upper layers of the MAP network. Delays for these layers in ILAN can be specified for particular nodes as well as between different message types.
3. A fast and efficient simulation routine for token bus protocol has been innovated. In medium and light traffic situations, much of the simulation time is spent in transmitting the token from one node to another on the logical ring(s). Depending on the amount of traffic on the network, the fast token bus simulation routine can significantly increase the speed of simulation. When long simulation runs are needed for MAP networks that have very light traffic, simulating the token passing from node to node without transmitting any data may not be feasible.

4. A methodology for describing messages to specify the sequences of message transmissions graphically has been developed. The read requests, read responses, write requests, write responses, and file transfers as well as the sequences needed to establish association connections can be defined.
5. Future improvements such as changes in sequences needed to establish associations, software that reduces application delays and hardware that results in shorter delays in the protocol stacks for each message at each node can be incorporated.
6. The procedure for constructing ILAN graphical user interface within the AutoCAD environment has been illustrated. Using AutoCAD's built-in features, functions and AutoLISP programming language, similar graphical user interface prototypes for other simulation programs can be constructed. AutoCAD environment simplifies the graphical user interface development. Furthermore, AutoLISP delivers the power of an AI language such as LISP that is needed in order to manipulate AutoCAD functions.
7. ILAN graphical user interface helps users check the input of the problem to be simulated. The ability to see the graphical representation of the problem as it is being described aids users in verifying the correctness of the input. For example, the spawning relationships between the messages can be seen as these messages are described.
8. One of the most important procedures that should be included with any graphical user interface is the FIND procedure. For a large problem, attempting to locate a particular node or message manually can be almost

impossible. In ILAN, FIND_MS and FIND_ND functions can be invoked to locate any specific message or node, respectively.

9.4 Further Research

1. For large problems the regenerations of drawings in AutoCAD may take a few minutes. If this is a problem, then another graphical environment is needed.
2. For more accurate prediction of the transmission delay, stack delays associated with transmitting and receiving as well as an application delay for each message at each station needs to be established experimentally.
3. The ILAN graphical user interface uses the maximum allowable size for AutoLISP programs within AutoCAD Release 10 running under the DOS operating system. No other functions can be added to the existing DOS version of ILAN graphical system. However, under the OS/2 operating system, the graphical interface can be expanded. The need of additional functions as well as the development of these functions are the subject of further research.
4. In OS/2, AutoLISP or C programming language may be used to analyze the results of the simulation. Unusual results may be detected and reported to the users. Furthermore, after analyzing the simulation output, the program can configure a better MAP network that meets specified goals and can then simulate this new problem to permit evaluation of proposed changes.
5. Further research is required to incorporate other parameters into the model. For example, a better way of including delays in higher layers of the MAP

protocol should be considered. Since different services may be required at different layers for any given circumstance, determining the best way to incorporate higher-layer delays into the simulation model without severely decreasing the speed of the simulation will require further study.

LIST OF REFERENCES

LIST OF REFERENCES

1. Aldelsberger, H. H., and Neumann, G., "Goal Oriented Simulation Modeling Using Prolog," *AI, Graphics and Simulation*, Society for Computer Simulation, La Jolla, GA., pp. 42-47, 1985.

2. Alemparte, M., Cheda, D., Seeley, D., and Walker, W., "Interacting with Discrete Simulation using on Line Graphic Animation," *Computer and Graphics*, Vol. 1, No. 4, pp. 309-318, December 1975.

3. Al-Hakim, D., Handy, J., Lanka, P and Payne, J., *MAP Stack Response Time for the University of Tennessee*, EDS., August 1, 1989.

4. Andrews, E. W., and Schultz, G. D., "A Token-Ring Architecture for Local Area Networks," *Submitted to IEEE Project 802 by IBM Corp. for Consideration of Local Network Standards*, March 8, 1982.

5. *AutoCAD Drafting Package Reference Manual*, Autodesk, Inc., 1989.

6. *AutoLISP Programmer's Reference*, Autodesk, Inc., 1989.

7. Backes, F., "Transparent Bridges for Interconnection of IEEE 802 LANs," *IEEE Network*, Vol. 2, No. 1, pp. 5-9, January 1988.

8. Ben-Arieh, D., "Manufacturing System Application of a Knowledge Based Simulation," *Proceedings 8th. Annual Conf. on Computers and Industrial Engineering*, Vol. 11, No. 1-4, pp 1986.

9. Blum, H., Paccione, F., "A GPSS Simulation to Determine Congestion in a Ethernet Type Local Area Network," *Winter Simulation Conference Proceedings*, pp.639-644, 1984.

10. Cargill, C. and Soha, M., "Standards and Their Influence on MAP Bridges," *IEEE Network*, Vol. 2, No. 1, pp. 87089, January 1988.

11. Chauhan, V. and Sethi, A. S., "Performance Studies of Token Based Local Area Networks," *10th. Conference on Local Computer Networks*, Minneapolis, Minnesota, pp. 100-107, October 7-9, 1985.

12. Cherukuri, R., Li, L., and Louis, L., "Evaluation of Token Passing Schemes in Local Area Networks," *Computer Networking Symposium Proceedings*, pp. 57-68, December 10, 1982.

13. Cheung, S., Dimitriadis, S., Karplus, W. J., *Introduction to Simulation using NETWORKII.5*, CACI Products Company, 1987.

14. Chung, V.,I., *Performance Analysis of the RS232-C Switch on the MAP Network*, Master Thesis, The University of Tennessee, Knoxville, May 1990.

15. Cleary, J., Goh, K. and Unger, B., "Discrete Event Simulation in Prolog," *AI, Graphics and Simulation*, pp. 8-13, 1985.

16. Conway, R., Maxwell, W. L., "XCELL: A Cellular, Graphical Factory Modelling System," *Winter Simulation Conference Proceedings*, pp. 160-163, 1986.

17. Cooper, E. B., "Broadband Network Design: Issues and Answers," *Computer Design*, pp. 209-216, March 1983.

18. Cox, W. W., "GPSS/PC Graphics and Animation," *Winter Simulation Conference Proceedings*, pp. 129-135, 1988.

19. Dahl, O., Myhrhaug, B., Nygaard, K., "SIMULA Common Base Language," *Publication S-22*, Norwegian Computing Center, Oslo, 1970.

20. Davies, N. R., "A Modular Interactive System for Discrete Event Simulation Modelling," *Proceedings of the Ninth Hawaii International Conference in System Sciences*, Western Periodical, pp. 296-299, January 1976.

21. DEC, Intel, and Xerox Corp., "The Ethernet: A Local Area Network, Data Link Layer and Physical Layer Specifications," *Version 2.0 Digital Equipment Corporation*, November 1982.

22. De Swaan Arons, D. H., "Expert Systems in the Simulation Domain," *Mathematics and Computers in Simulation XXV*, North-Holland Publishing Company, pp. 10-16, 1983.

23. Doukidis, G. I. and Paul, R. J., "Research into Expert Systems to Aid Simulation Model Formulation," *Journal of Operational Research Society*, Vol. 36, No. 4, pp. 319-325, 1985.

24. Duersch, R. R. and Laymon, M. A., "A Graphic Workflow Simulator," *Summer Computer Simulator Conference Proceedings*, pp. 141-144, 1983.

25. Elmaghraby, A. S. and Jayunnathan, V., "An Expert System for Simulationists," *AI, Graphics and Simulation*, pp. 106-109, 1985.

26. Faught, W. S., Klahr, P., and Martins, G. R., "An Artificial Intelligence Approach to Large-Scale Simulation," *Summer Computer Simulation Conference Proceedings*, pp. 231-235, 1980.

27. Faught, W. S., "Application of AI in Engineering Computer," *Computer*, 19:7, pp. 17-27, 1986.

28. Fishman, G. S., *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley and Sons, New York, 1977.

29. Futo, I., "Combined Discrete/Continuous Modeling and Problem Solving," *AI., Graphics and Simulation*, pp. 23-28, 1985.

30. Gabriel, Richard P., "Lisp Expert Systems Are More Useful," *Electronics*, pp. 65, August 7, 1986.

31. Gould Inc., Modicon Division, "Modway Public Industrial Local-Area Network: Conceptual Description," *Product Bulletin*, 1981.

32. Grant, M.R. and Darrell, W.S., "A Tutorial on TESS: The Simulation Support System," *Winter Simulation Conference Proceedings*, pp. 136-140, 1988.

33. Helman, D. H., and Bahuguna, A., "Explanation Systems for Computer Simulations," *Winter Simulation Conference Proceedings*, Wilson, J., Henrikson, J., Roberts, S. (eds.), pp. 453-459, 1986.

34. Hill, T. R., and Roberts, S. D., "A Prototype Knowledge-Based Simulation Support System," *Simulation*, Vol. 48, No. 4, pp. 152-161, April 1987.

35. Hills, P. R., *HOCUS manual*, P. E. Consulting Group, Egham, England, 1969.

36. Hurrion, R. D., "An Investigation of Visual Interactive Simulation Methods Using the Job-shop Scheduling Problem," *JORS (U.K.)*, 29, pp 1085-1093, 1978.

37. Hurrion, R. D., "An Interactive Visual Simulation System for Industrial Management," *EJOR (Netherlands)*, 5:2, pp. 86-93, Aug.1980.

38. Hurrion, R. D., "Visual Interactive Simulation using a Micro-computer," *Computers & Operations Research*, Vol. 8, No. 4, pp. 267-273, 1981.

39. Imhoff, C. H., and Cannon, L. W., "Interactive Simulation Graphics with INSIGHT," *Winter Simulation Conference Proceedings*, pp. 317-321, 1984.

40. Jackman, J. K., and Medeiros, D. J., "Modeling and Analysis of Ethernet Networks," *Winter Simulation Conference Proceedings*, pp. 595-602, 1984.

41. Jacobson, A. D., "LISP Is Not Needed for Expert Systems," *Electronics*, pp. 64-65, August 7, 1986.

42. James, F. H. and Martin L., *The AutoCAD Database Book: Accessing and Managing CAD Drawing Information*, Ventana Press, Chapel Hill, North Carolina, 1987.

43. Jasany, L. C., "MAP: A Struggle in the Making," Automation, pp. 10-14, November 1987.

44. Kaminski, M. A., JR., "Protocols for communicating in the factory," *IEEE Spectrum*, April 1986, pp. 56-62.

45. Keramati, B., Kelly-Sacks, C. M., and Tonkay, G. L., "Simulation and Animation of an Assembly System," *Winter Simulation Conference Proceedings*, pp. 559-661, 1983.

46. Kerckchoffs, E. J. H. and Vansteenkiste, G. C., "The impact of advanced information processing on simulation - An illustrative review," *Simulation*, Vol. 46, no. 1, pp. 17-46, January 1986.

47. Kornell, J., "Reflections on Using Knowledge Based Systems for Military Simulation," *Simulation*, Vol. 48, No. 4, pp. 144-148.

48. Kuehn, P. J., "Multiqueue Systems with Nonexhaustive Cyclic Service," *B.S.T.J.*,58, No. 3, pp. 671-698, March 1979.

49. Linge, N., Ball, E., Tasker, R. and Kummer, P., "A Bridge Protocol for Creating a Spanning Tree Topology within an IEEE 802 Extended LAN Environment," *Computer Networks and ISDN Systems*, 13, Elsevier Science Publishers, North-Holland, pp. 323-332, 1987.

50. *Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/DC)*, Standard for (Draft International Standard) (ANSI/IEEE Std. 802.3), 1985.

51. *Local Area Networks: Token Passing Bus*, Standard for (Draft International Standard) (ANSI/IEEE Std. 802.4), 1985.

52. *Local Area Networks: Token Passing Ring*, Standard for (Draft International Standard) (ANSI/IEEE Std. 802.5) 1985.

53. Luker, P. A., and Stephenson, J., "Simulation without Programs," *Proc. UKSC Conference on Computer Simulation*, Chester, England, pp. 181-191, 1978.

54. Manuel, T., "What's Holding Back Expert System?," *Electronics*, pp. 59-63, August 7, 1986.

55. *MAP Specifications*, Version 2.1 General Motors Corp., Warren, MI 48090, March 1985.

56. *MAP Specification*, Version 2.2 General Motors Corp., Warren, MI 48090, 1986.

57. *MAP Specification,* Version 3.0 General Motors Corp., Warren, MI 48090, 1987.

58. Maisel, H., Gnugnoli, G., *Simulation of Discrete Stochastic Systems*, Science Research Associates, Inc., 1972.

59. Marathe, M. V. and Smith, R. A., "Performance Analysis of MAP Network Adapter," *Net/One Technical Journal*, Ungermann-Bass, Inc,1989.

60. Mathewson, S. C., "Simulation Program Generators," *Simulation*, Vol. 23, No. 6, pp. 181-189, December 1974.

61. Mathewson, S. C., and Beasley, J. E., "DRAFT/ SIMULA," *Proceedings of the 4th. SIMULA User's Conference*, Amsterdam, 1976.

62. Mathewson, S. C., and Allen, J. A., "DRAFT / GASP - A Program Generator for GASP," *Proc. of the 10th. Annual Simulation Symposium*, Tampa, FL., pp. 221-227, 1977.

63. McCoy, E. E., *LANAGO: A Local Area Network Design Algorithm*, PhD. Dissertation, The University of Connecticut, 1985.

64. Melman, M., Livny, M., "Simulation Models of the ETHERNET Protocol," *Summer Computer Simulation Conference Proceedings*, pp. 116-121, 1983.

65. Melamed and Moris R.J.T., "Visual Simulation: The Performance Analysis Workstation," *Computer*, 18, pp. 87-94, 1985.

66. Metcalfe, R. M., and Boggs, D. R., "Ethernet: Distributed Packet Switching for Local Computer Networks," *CACM*, Vol. 19, No. 7, pp. 395-404, July 1976.

67. Miles, T., Sadwski, R.P. and Werner, B.M., "Animation with CINEMA," *Winter Simulation Conference Proceedings*, pp. 180-187, 1988.

68. Miner, R. J., Sabuda, J. D., and Wortman, D. B., "Graphics and Simulation: Tools and Applications," *Winter Simulation Conference Proceedings*, pp. 161-178, 1982.

69. Mok, D. S. T., *Performance Analysis and Comparison of Local Area Networks (LANs)*, PhD. Dissertation, Industrial Engineering, Iowa State University, 1985.

70. *NETWORK II.5 User's Manual*, Version 3.1, CACI Products Company, 1988.

71. Nielsen, N. R., "Expert Systems and Simulation," *Computer Networks and Simulation III*, S. Shoemaker (Editor), pp. 41-53, 1986.

72. O'Keefe, R., "Simulation and Expert Systems - A Taxonomy and Some Examples," *Simulation*, 46:1, pp. 10-16, January 1986.

73. O'Keefe, R., "What is Visual Interactive Simulation ? (And is there a Methodology for doing it right ?)," *Winter Simulation Conference Proceedings*, pp. 461-464, 1988.

74. Oren, T. I., "Computer-Aided Modelling Systems," *Progress in Modelling and Simulation (Cellier, ed.)*, Academic Press, London, U.K., pp. 189-203, 1982.

75. Overstreet, C. M. and Nance, R. E., "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM*, Vol. 28, No. 2, pp. 190-201, February 1985.

76. Palme, J., "Moving Pictures Show Simulation to User," *Simulation*, 29(6), pp. 202-209, 1977.

77. Paul, R. J., Doukidis, G. I., "Further Developments in the Use of Artificial Intelligence Techniques which Formulate Simulation Problems," *Journal of Operational Research Society*, Vol. 37, No. 8, pp. 787-810. 1986.

78. Peterson, J. L., "Petri Nets," *ACM Computing Surveys*, Vol. 9, No. 3, pp. 223-252, September 1977.

79. Pegden, C. D., *Introduction to SIMAN with Version 3.0 Enhancements*, System Modeling Corporation, 1985.

80. Pimentel, J.R., "Performance Simulation of the IEEE Token Bus Protocol using SIMAN," *Computer Science and Technology*, NBS Special Publication 500-127, pp. 5-34, 1986.

81. Pritsker, A. A. B., *The GASP IV Simulation Language*, John Wiley, 1979.

82. Raker, D. and Rice, H., *INSIDE AutoCAD: The Complete AutoCAD Guide*, New Riders Publishing, Thousand Oaks, California, Fifth Edition, 1989.

83. Roberts, S. D. and Flanigan, M. A., "Simulation Modeling and Analysis with INSIGHT: A Tutorial," *Winter Simulation Conference Proceedings*, pp. 102-111, 1988.

84. Ruiz-Mier, S., Talavage, J., and Ben-Arieh, D., "Towards a Knowledge-Based Network Simulation Environment," *Winter Simulation Conference Proceedings*, pp. 232-236, 1985.

85. Sachs, S. R., Kan, K., and Silvester, J. A., "Performance Analysis of a Token-Bus Protocol and Comparison with Other LAN Protocols," *10th Conference on Local Computer Networks*, Minneapolis, Minnesota, pp. 46-51, Oct. 7-9, 1985.

86. Schruben, L., "Simulation Modeling with Event Graph," *Communications of the ACM*, Vol. 26, No. 11, pp. 957-963, November 1983.

87. Sethi, A. S., and Saydam, T., "Performance Analysis of Token Ring Local Area Networks," *Proc. IEEE 9th. Conference on Local Computer Networks*, Minneapolis, Mn., pp. 26-31, October 1984.

88. Shannon, R. E., *Systems Simulation: The Art and Science*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

89. Sloman, A., "Why We Need So Man Knowledge Representation Formalisms," *Research and Developments in Expert Systems (Bramer, ed.)*, Cambridge University Press, Cambridge, U. K., pp. 1630184, 1984.

90. Smith, J. and Gesner, R. *CUSTOMIZING AuotCAD*, Second Edition, New Riders Publishing, Thousand Oaks, California, 1989.

91. Smith, J and Gesner, R. *INSIDE AutoLISP: Using AutoLISP to Customize AutoCAD*, New Riders Publishing, Thousand Oaks, California, 1989.

92. Stallings, W., "The IEEE 802 Local Network Standards," *Telecommunications*, pp. 40-48, March 1986.

93. Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F., and Sacerdoti, E., *Building Expert Systems, Hayes-Roth, F. (ed.)*, Addison-Wesley, 1983.

94. Suppan-Borowka, J., "Analyzing the Performance of the MAP Backbone," *Datacom Verlag*, Pulheim, Germany, 1988.

95. Taha, A. H., "Introduction to SIMNET v.2.0," *Winter Simulation Conference Proceedings*, pp. 93-101, 1988.

96. Thomet, M. A., "The Aircraft Movement Simulation Model," *Winter Simulation Conference Proceedings*, pp. 641-646, 1983.

97. Torn, A. A., "Simulation Nets, a Simulation Modeling and Validation Tool," *Simulation*, Vol. 45, No. 2, pp. 71-75, 1985.

98. Turing, A. M., "Computing Machinery and Intelligence," *Mind*, Vol. LIX, pp. 433-360, 1950.

99. Ulug, M. E., "Comparison of Token Holding Time Strategies for a Static Token Passing Bus," *Computer Networking Symposium Proceedings*, pp. 37-44, December 11, 1984.

100. Vaucher, H., "Views of Modeling: Comparing the Simulation and AI Approaches," *AI, Graphics and Simulation*, pp. 3-7, 1985.

101. Watson, W. B., "Simulation Study of the Traffic Dependent Performance of a Prioritized, CSMA Broadcast Network," *Computer Networks*, pp. 427-434, December 1978.

102. Winston, P. H., *Artificial Intelligence*, 2nd. Ed., Addison-Wesley, 1984.

103. Willis, R. and Austell W. P., "GMSS: Graphics Modelling and Simulation System," *Proc. of the 16th. Annual Simulation Symposium*, Tampa, Fl., pp. 137-160, 1983.

104. Yamamoto, Y. and Lenngren, M., "Graphic Model Building System," *Proc. of the 16th. Annual Simulation Symposium*, Tampa, Fl.a, pp.161-176, 1983.

105. Zeigler, B. P.,*Theory of Modelling and Simulation*, Wiley, 1976.

106. Zhang, L., "Comparison of Two Bridge Routing Approaches," *IEEE Network*, Vol.2, No. 1, pp. 44-48, January 1988.

APPENDICES

APPENDIX A

COMMON VARIABLES

BUS(I) = Total time that network I is busy.

DAPPL(I,K) = Application delay for message I on station K.

D27(I,K) = Reception stack delay of station K receiving packets belonging to message I on the MAP network.

D27(I,K) = Transmission stack delay of station K transmitting packets belonging to message I on the MAP network.

FER = Fraction of transmission error for each frame.

HED = Headend delay for broadband network.

IB27(I) = .TRUE. If reception stack of station I is busy; Otherwise = .FALSE.

IB72(I) = .TRUE. IF transmission stack of station I is busy; Otherwise = .FALSE.

IDI(I,K) = Five characters name of statistical distribution type for station I. Equivalent to DSTBR(I).

INETY(I,K) = Five-character name of network type for network I, which can be either CARRI for carrierband or BROAD for broadband. Equivalent to NTYPE(I).

KDESQ = New current destination global station number.

KNET = Current network number.

KNEWQ(I) = New current global station on network I.

MINF = Number of frames transmitted after which the statistics collection is started.

MULT = Multiple runs if equal to 1.

NAPG = Maximum number of message groups. Message numbers 1 through NAPG are message group numbers. Normal message numbers are from NAPG + 1 to NAPPL.

NAPPL = Total number of messages, which includes message groups as well as normal messages. Message group numbers are from 1 to NAPG. Normal message numbers are from NAPG + 1 to NAPPL.

NAS(I) = 1 if message I requests an association with the destination station;
= 4 if message I requests an requests a association disconnection;
= 5 if message I finished the association;
= 6 if message I finished the association disconnection.

NASS(I,K) = Total number of associations for station K, if I = 1. Equal to current number of associations for station K, if I = 2.

NAST(I) = Start message number for message group.

NBDG(I) = List of global station number of I^{th} bridges in the system.

NTST(I,K) = Twenty destination station numbers (K = 1 through 20 only) where message type I can transmit to.

NET(I) = Number of stations in network I.

NETNO(I,K) = I^{th} network number for station K: A station may be on several networks. For example, if the second network of station 10 is on network number 3, then NETNO(2,10) = 3.

NETOL = Total number of sub-networks.

NFAME = Total number of different frame sizes.

NFSIZ = Current frame size type number.

NQGLO(I,K) = Global station number for local station K in sub-network I.

NQLOC(I,K) = Local station number for global station K in sub-network I.

NSENT(I) = Number of frames transmitted from network I (double precision).

NSETO = Number of frames transmitted from all networks (double precision).

NSTA = Total number of stations on the MAP network.

NTOBR(I,K) = Global station number of the bridge that network I must transmit to in order to reach network K.

OVH(I) = Overhead (time to transmit token) on network I.

P(I,K) = Parameter number I for statistical distribution specified for station K.

PAPPL(I,K) = Cumulative probability that station K is transmitting message type I.

PDTST(I,K) = Cumulative probability of transmitting K^{th} station in destination station set I.

PERAC = + or - percent accuracy of the mean time in the system. System ended when this accuracy at PERCO is met.

PERCO = Percent confidence interval of the mean time in the system. Simulation run ended when this level of confidence and PERAC is met.

PERRO = Percent total transmission errors.

PFRMN(I,K) = Cumulative probabilities for transmitting I frames (I = 1 to 20).

PFSZ(I,K) = Cumulative probability for frame size I of message K.

PLAUN(I,K) = Cumulative probability distribution of launching message K after finishing message I to arrive at the station that received the last transmitted frame from message I.

PRPD(I,K) = Propagational delay for station K on sub-network I.

PSPAW(I,K = Probability of producing I - 1 spawns from message K.

NQB(I) = Number of frames in station I when statistic collection is started.

SEND(N,I) = Total Frame length for one token rotation at station I on sub-network N.

SENDL(N) = Last frame length transmitted from sub-network N.

THT(I) = Token rotation time for sub-network I.

TRFRM(I,K) = Transmission time for a frame of size K in sub-network I.

TSTART = Time at which statistics are collected.

APPENDIX B

EVENT SUBROUTINES

Subroutine L7ARR represents event 1, which is the arrival of message or frame at the highest layer in the protocol stack. Figure B-1 shows the flowchart of subroutine L7ARR. If attribute number 2 of the entity in event 2 is less than 2000000, then this is a frame.[1] All attributes for this frame have previously been calculated.[2] The values for source station numbers and message types from its attributes are decoded. The frame entities are disposed by using SIMAN's DISPOS subroutine. Otherwise, the entity is either an externally created (according to a specified statistical distribution) message or an internally created (spawned) message.

The values for message types, the number of times left to spawn new messages and message creation types for the message are determined from its attributes. If the message is externally scheduled, then the next message is scheduled for the same station. Otherwise, the message entity is removed from the system by calling the DISPOS subroutine. If the entity is a group message type and the association status for the message group is not an association disconnection, then the group application is set

---

[1] From table 2, recall that only message entity can have attribute number 2 grater than 2,000,000.

[2] By definition all attributes for frames are calculated before forming a frame entity. New message entities, on the other hand, may have some attribute values undermined.

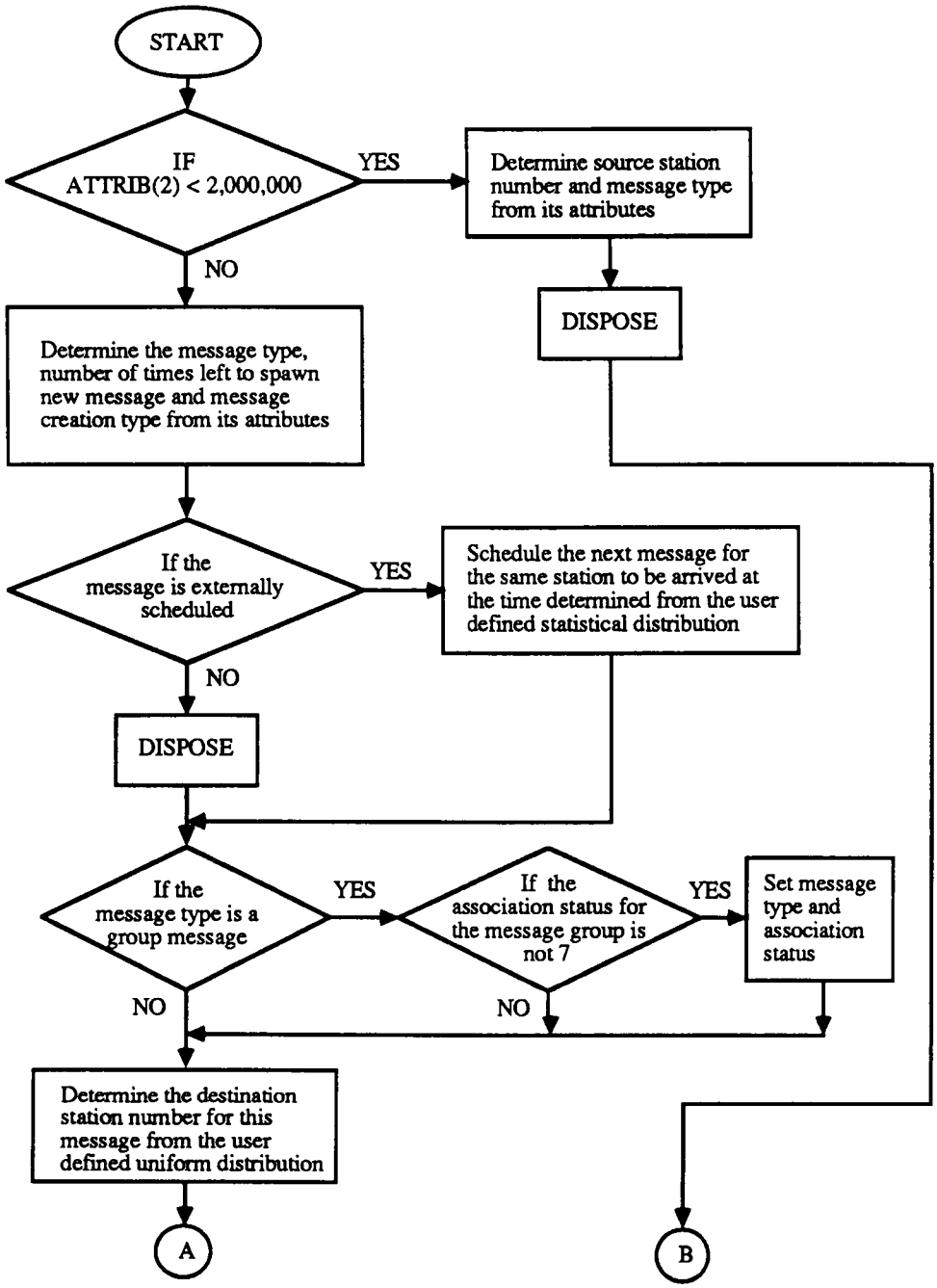


Figure B-1. Flowchart of Subroutine L7ARR

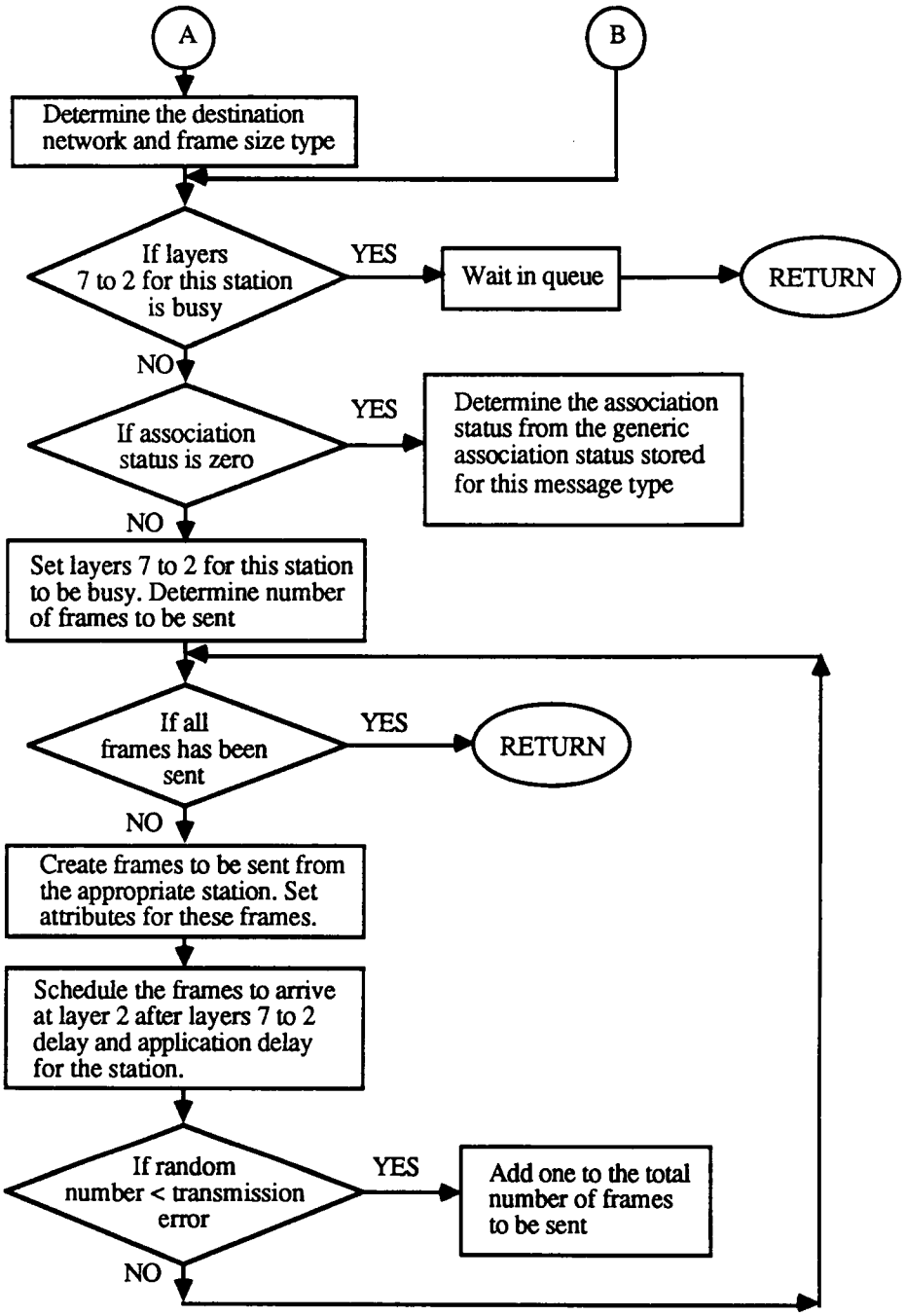


Figure B-1. (continued)

to NAST (group application number).[3] The destination station number for this message is determined from the user-defined uniform distribution. The destination network and frame size type are determined from user-defined cumulative probability distribution. For both frame and message entities, if the transmission stack for this station is busy, then the message waits in the transmission stack queue and return. Otherwise, if the association status is zero, then the association status is determined from NAST(I)[3]. At this point, the transmission stack is not busy. The transmission stack is set to busy to process the current message. No other messages will be processed by the transmission stack at this station until it has finished processing the current message. The number of frames to be transmitted is determined, and the attributes are set for these frames. The frames must be scheduled to arrive at layer 2 according to delays given by the transmission stack delay and application delay for the message on this station. A random number is generated for each frame transmitted. If a random number is less than or equal to the transmission error, then one more frame is transmitted. When all frames have been transmitted, the program returns to the caller.

Event 2, called L2ARR, is the frame arrival at layer 2 of the transmitting station. Figure B-2 shows the flowchart of subroutine L2ARR. The attributes of the frame that arrived in this event are decoded for the value of the current source station number. The frame is then inserted into the waiting queue of the source station awaiting the transmission across the physical media. The status of transmission stack of the source

---

[3] Variable NAST(I) is a user input parameter for generic association status of message type I.

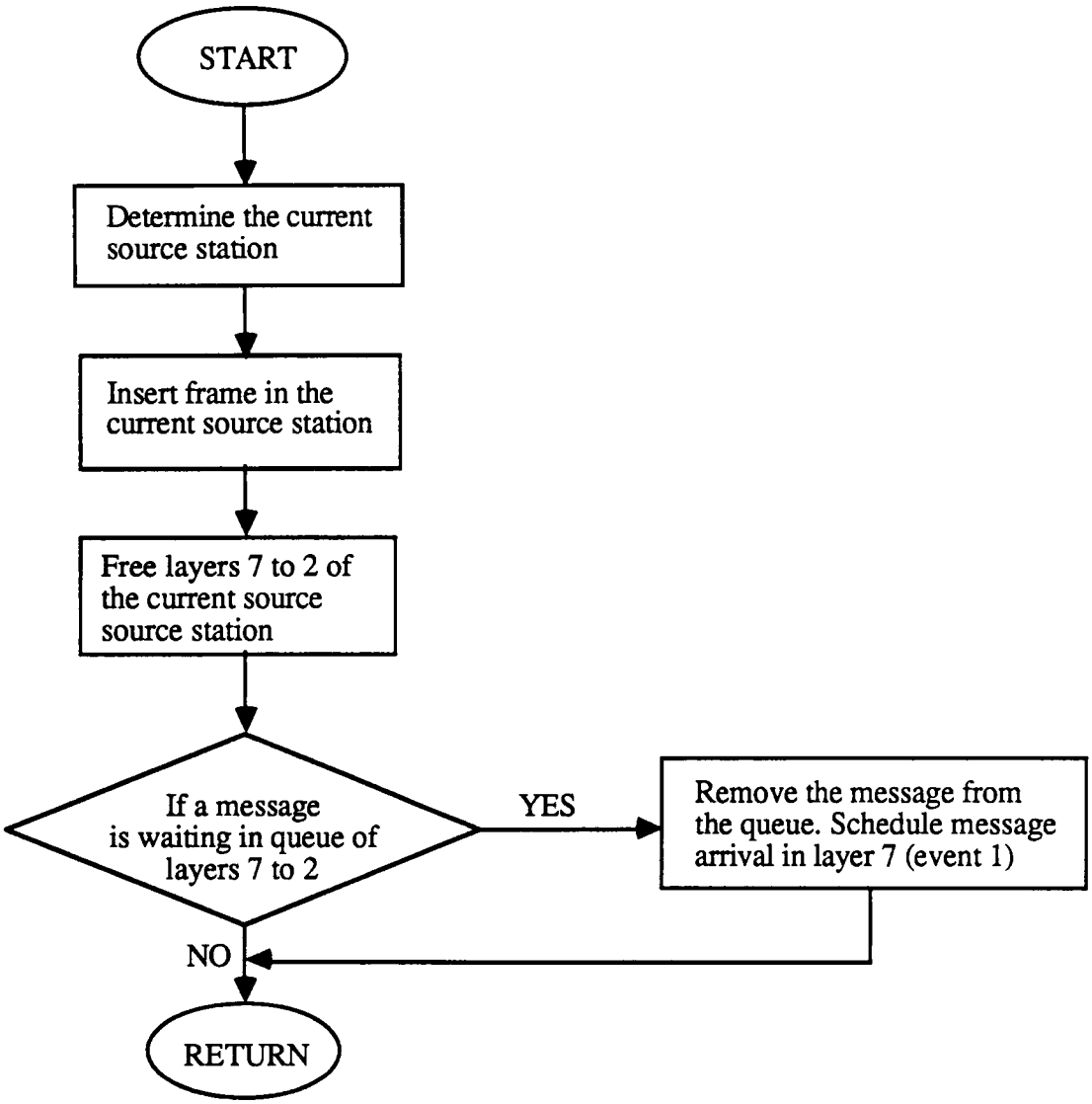


Figure B-2. Flowchart of Subroutine L2ARR

station is then set to not busy. If there is a message waiting in the queue of the transmission stack, then the message is removed form the queue and scheduled to arrive in layer 7 (event 1) before returning.

Event 3 of ILAN is called CHNGQ, representing the movement of the tokens as they move from one station to another. The logic in CHNGQ subroutine allows the token to move efficiently instead of simply moving to the next station without determining whether that station has anything to transmit or not. Figure B-3 shows the flowchart of subroutine CHNGQ. At the beginning of the subroutine, the sub-network number of the token entity is determined. The first and the last record locations of the event file are determined. From the first record location in the event file, the time of the next event is then determined. If the next event is either event 1, 2, or 6, it must be determined whether the entity scheduled for the next event is a frame or a message, and the attributes of the message entity or frame entity must be decoded.

In any case, if the next event is event 6, and the frame is not being transmitted to a bridge, or if the frame is not being transmitted to and from stations on the same sub-network that this token is on, then look at another event. If the next event is not the last event scheduled, then the next event is set to be the event after the next event and to loop back to test for events 1, 2, or 6. If the last scheduled event is reached, then no event scheduled represents arrivals to this sub-network. The time of the next event is set to a large number, say .2E22. If the destination and source stations are on the same sub-network as the current token or the last event scheduled is reached, then KNXQL is set to be the current station on this sub-network. TNXTQ is set to be the time the token will arrive at the next station on this sub-network. KNXQL is moved to the next station

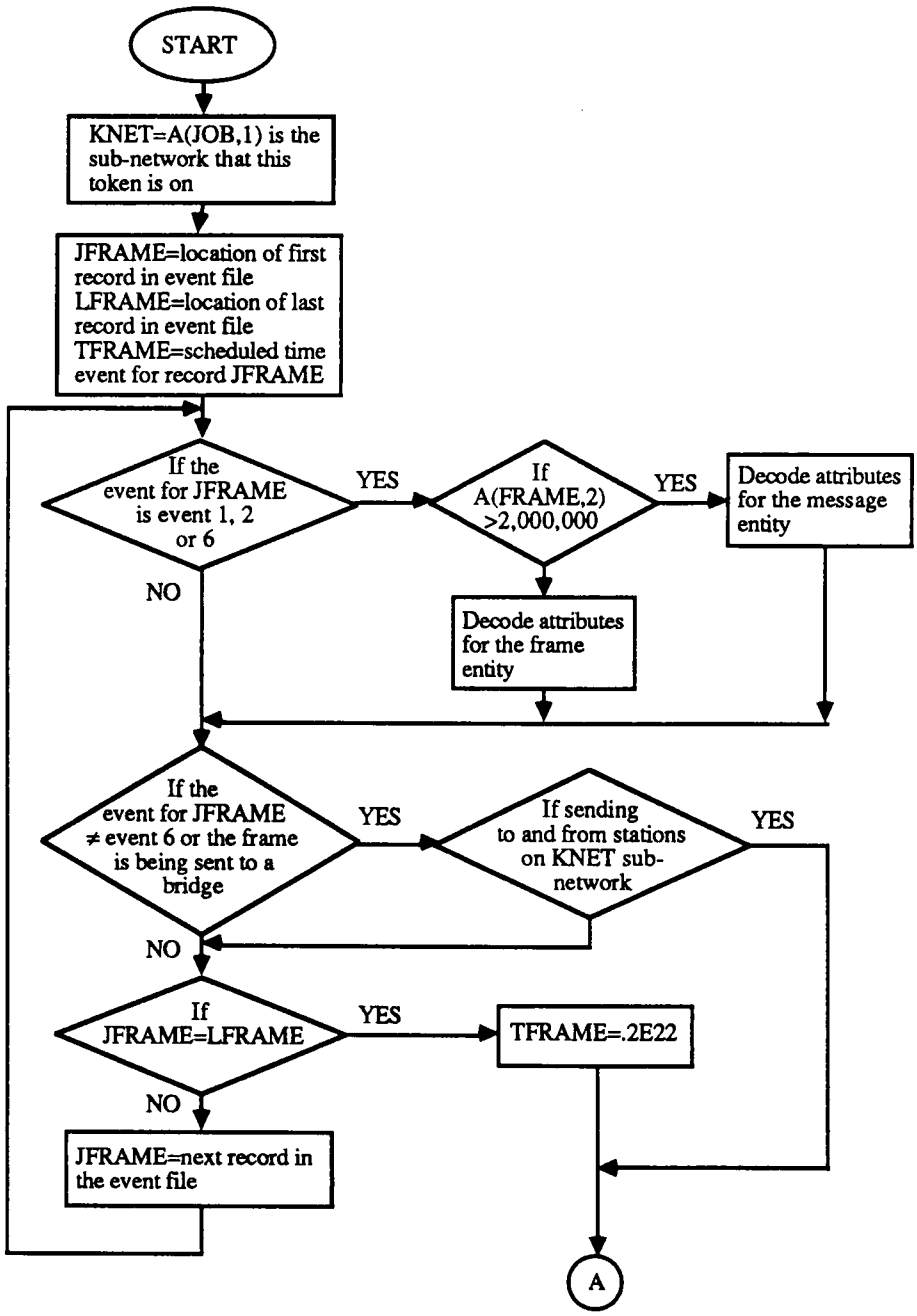


Figure B-3. Flowchart of Subroutine CHNGQ

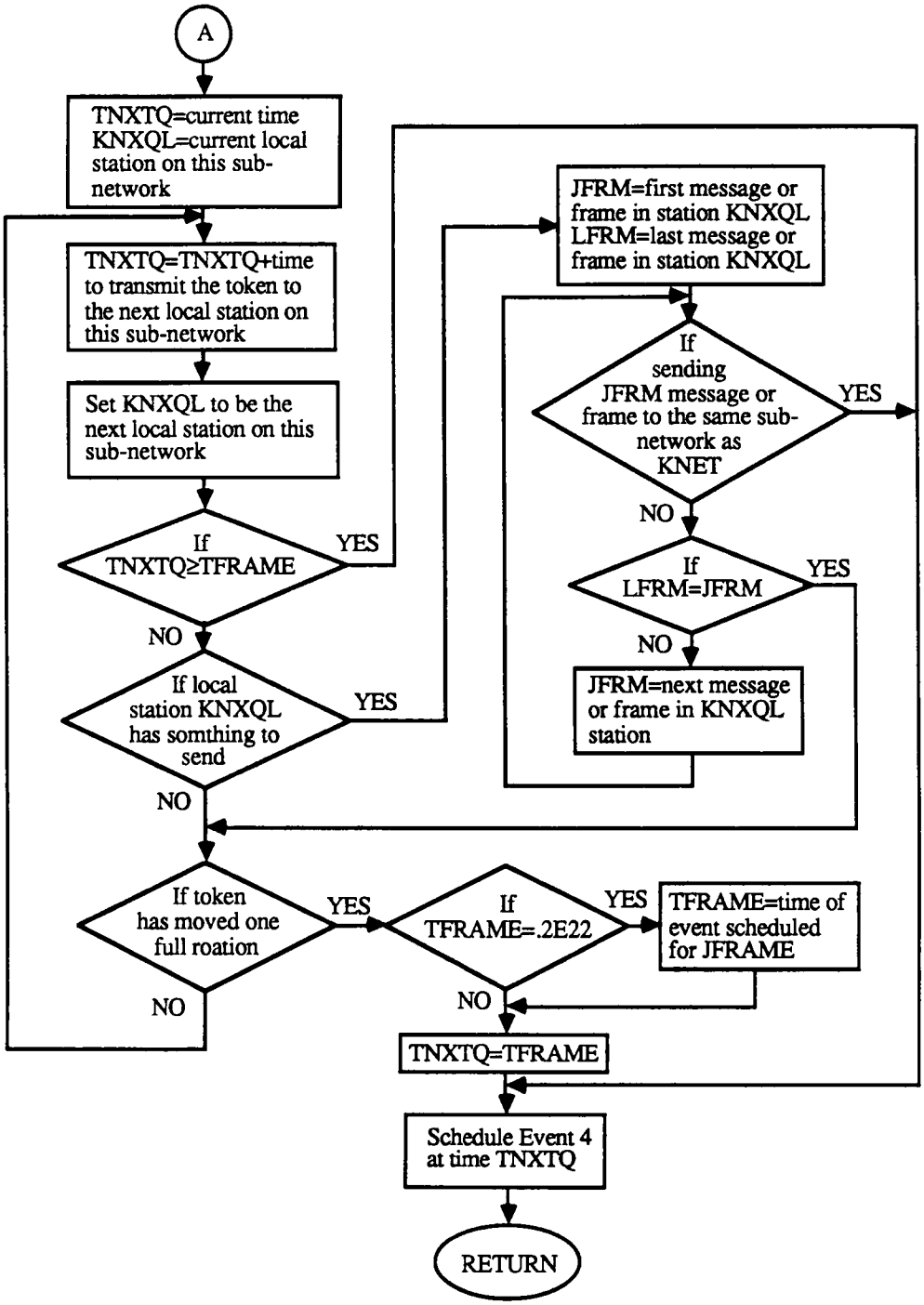


Figure B-3. (continued)

on this sub-network. If the time required for the token to arrive at the next station is greater than the time of the next event, or if the current station has a frame to be transmitted on the same sub-network that the current token is on, then event 4 is scheduled. Since the next event may be an arrival of a frame to be transmitted from the current station, the token should not be moved to the next station. Thus event 4 is scheduled to try to transmit a frame from the current station. Otherwise, if the token has moved less than one full rotation, then it loops back to set TNXTQ to be the time the token will arrive at the next station. Otherwise, if the time of the next event is .2E22, then event 4 is scheduled at the time that the token reaches the next station and the execution is returned to SIMAN. Recall that the next event time is set to .2E22 when no event on the event list represents transmitting a frame on this sub-network. Thus event 4 is scheduled to try to transmit from the next station. The token cannot be scheduled at later time because another event may be scheduled before that time.

TRANSM is the name of event 4. This event represents the transmission from the source station where there is something waiting to be transmitted. The flowchart of subroutine TRANSM is shown on Figure B-4. If there is nothing to transmit in the current station or time spent in this station has expired, then event 3 is scheduled to move the token to another station on the same sub-network at this time. All the frames that could not be transmitted but were removed from the station are restored and the execution returns to SIMAN. Otherwise, JOBQUE is set to the record location pointed to the first frame waiting to be transmitted from this station. The JOBQUE frame is removed from the queue of frames waiting to be transmitted in this station, and the

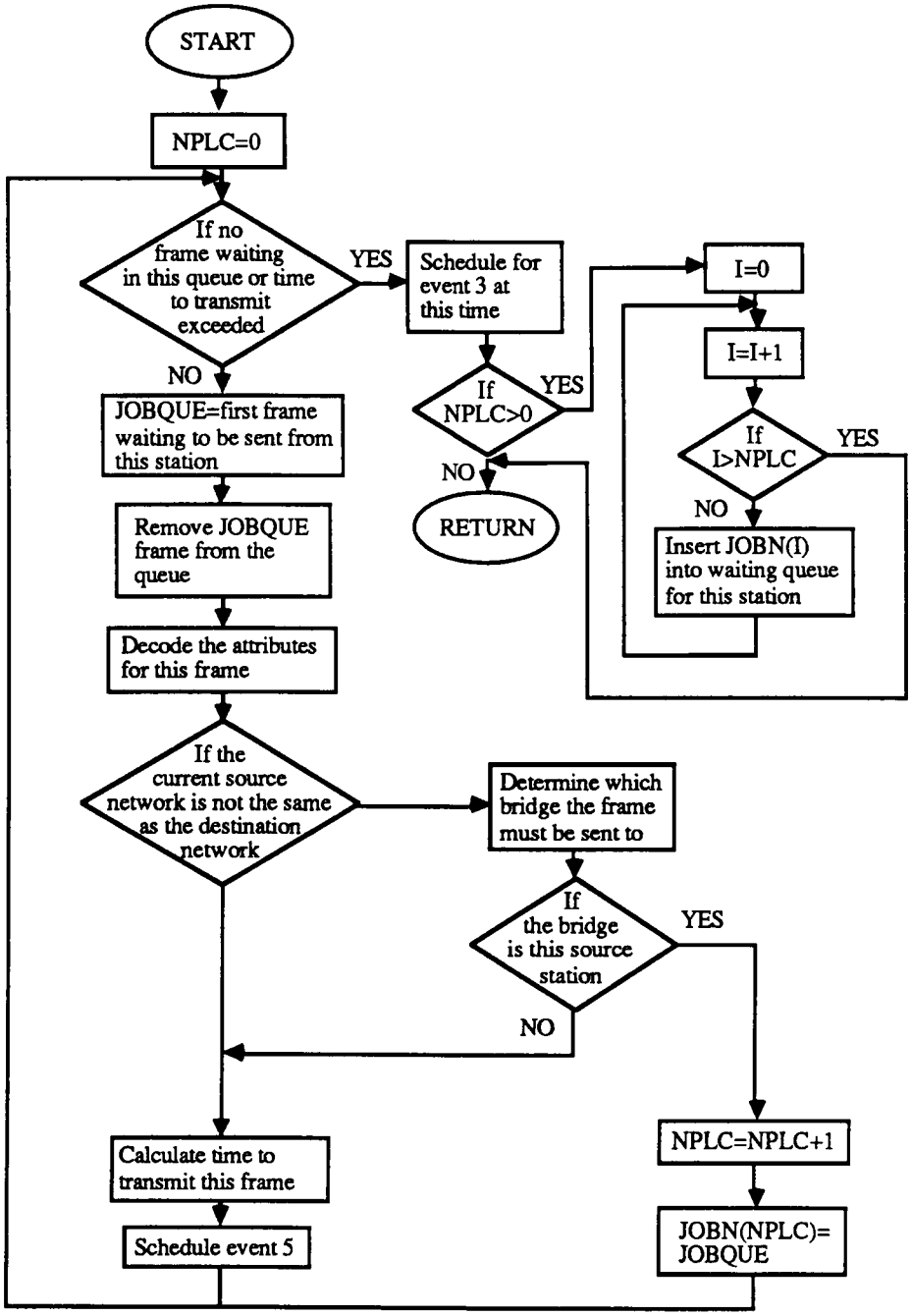


Figure B-4. Flowchart of Subroutine TRANSM

attributes for this frame entity are decoded. If the current source sub-network is not the same as the destination sub-network, then the station number of the bridge which will be the intermediate destination station from which the frame is to be transmitted is determined. However, if this bridge turns out to be the same station as the source station, then the JOBQUE frame is stored in the JOBN(NPLC) array and the NPLC counter is increased by 1. In this case, the frame is not transmitted because the current station is a bridge which can transmit to at least two different sub-networks. The sub-network that the token is currently on is not the same sub-network to which the frame is to be transmitted. Thus the frame must wait for the token on the correct sub-network to arrive. If the frame is to be transmitted, then the time to transmit this frame is calculated, and event 5 is scheduled for the reception of the frame at layer 2 of the destination station. Whether the frame is transmitted or not, the execution of the program loops back to determine if there is anything else to be transmitted from this station or if the time to transmit frames in this station has exceeded the limit. If not, the first frame waiting to be transmitted is stored in variable JOBQUE. It should have a different value from the previous execution of the same statement since the previous first frame waiting to be transmitted will have been removed from the waiting queue. The execution of this subroutine continues until there is nothing else to be transmitted or the time allowed for transmitting has expired.

Event 5 is called L2REC. This event represents the reception of the frame at layer 2 of the destination station. Figure B-5 shows the flowchart of subroutine L2REC. First, the attributes for this frame are decoded. If the reception stack of this station is busy, then the frame is inserted into the reception stack waiting queue for this

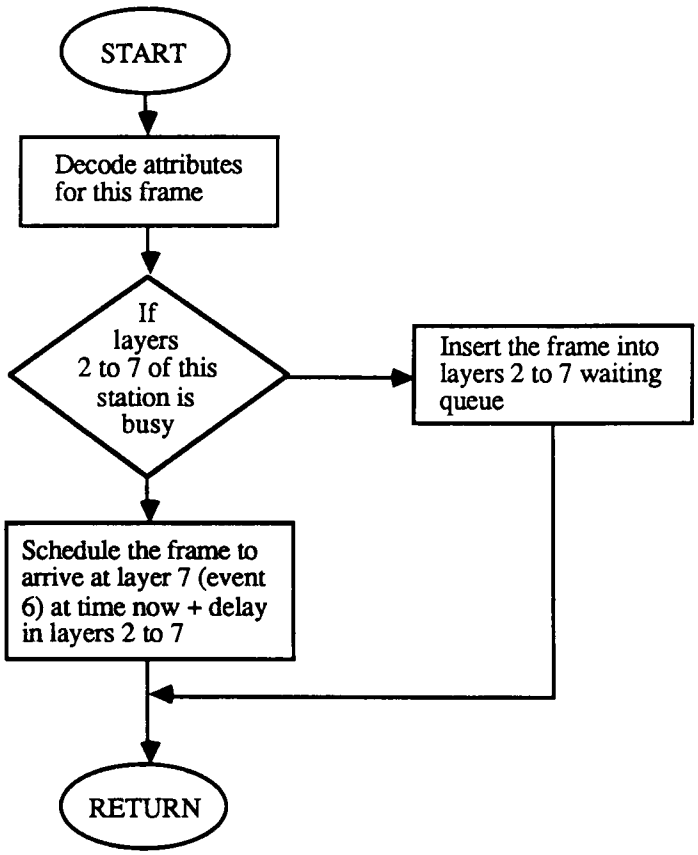


Figure B-5. Flowchart of Subroutine L2REC

station and the execution is returned to SIMAN. Otherwise, event 6, which is the arrival of the frame at layer 7, is scheduled, allowing for the processing time delay in the reception stack.

L7REC is the name for event 6, which represents the reception of frames at layer 7. Figure B-6 shows the flowchart of subroutine L7REC. The values of attributes for the frame entity is decoded. Since the frame has arrived at layer 7, the frame has passed through the reception stack. The status of the reception stack is set to not busy so that other frames can be processed. If the current station is not the final destination station but is an intermediate destination station, then event 1 is scheduled for the frame arrival at layer 7 of this station to continue to be transmitted to the final destination station. Otherwise, the frame is disposed of. In any case, if this frame is the last frame of the message, then set LAPPL variable (determined from the user-defined discrete probability distribution, called PLAUN) is set to the next message to be spawned. If the destination station is reached and either the number left to spawn is greater than zero or the frame belongs to a message group in the middle of transmitting frames within that group or a message group that transmits association disconnection after transmitting data then the program determines whether the association status of the frame is the association disconnection and the number left to spawn is not zero. If this is the case, LAPPL is set to the next message that is pointed to by NAST(I), which is the start message number for the message group I. NDS (the destination station of the new message) is set to be the current station. Otherwise, if the association status is 1, which is an association request, and if the maximum number of associations for this station is not reached, then the association status for the new message is set to 2 and the

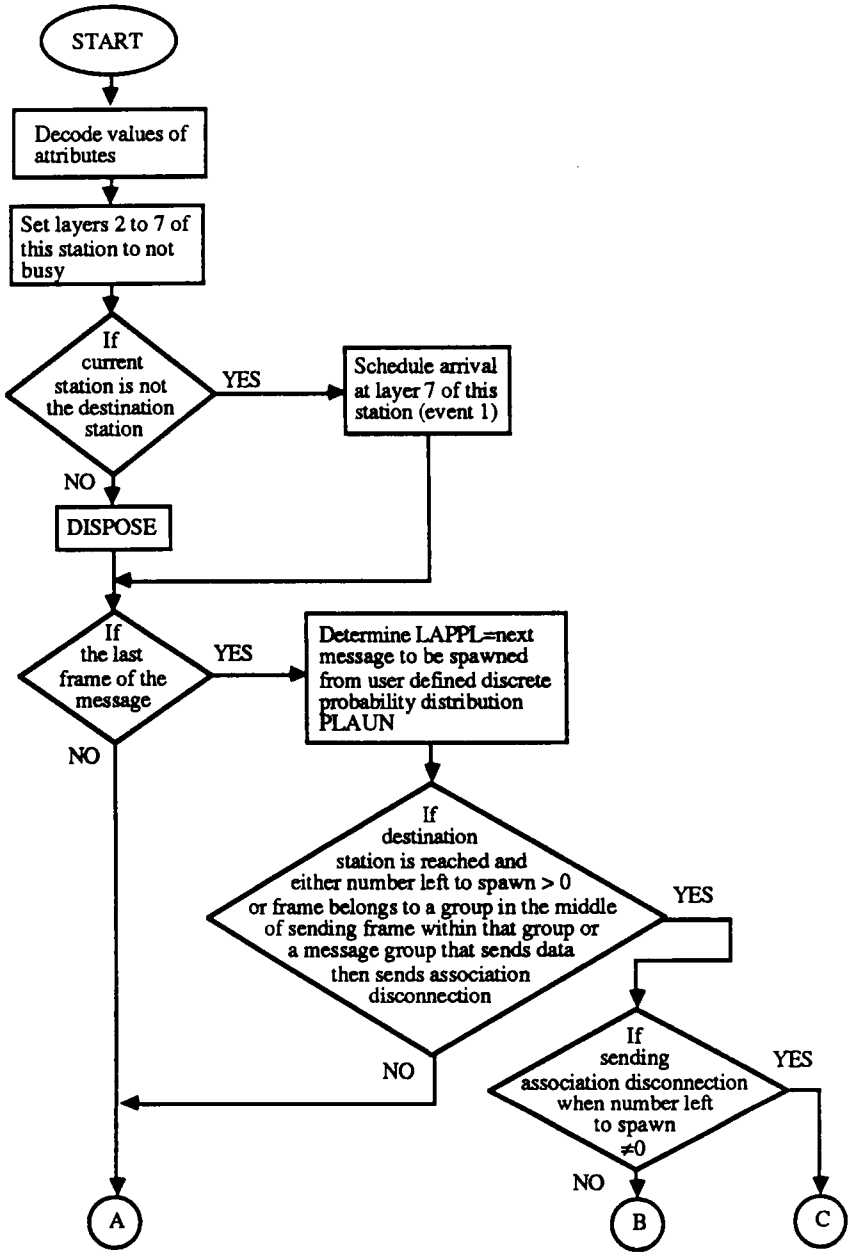


Figure B-6. Flowchart of Subroutine L7REC

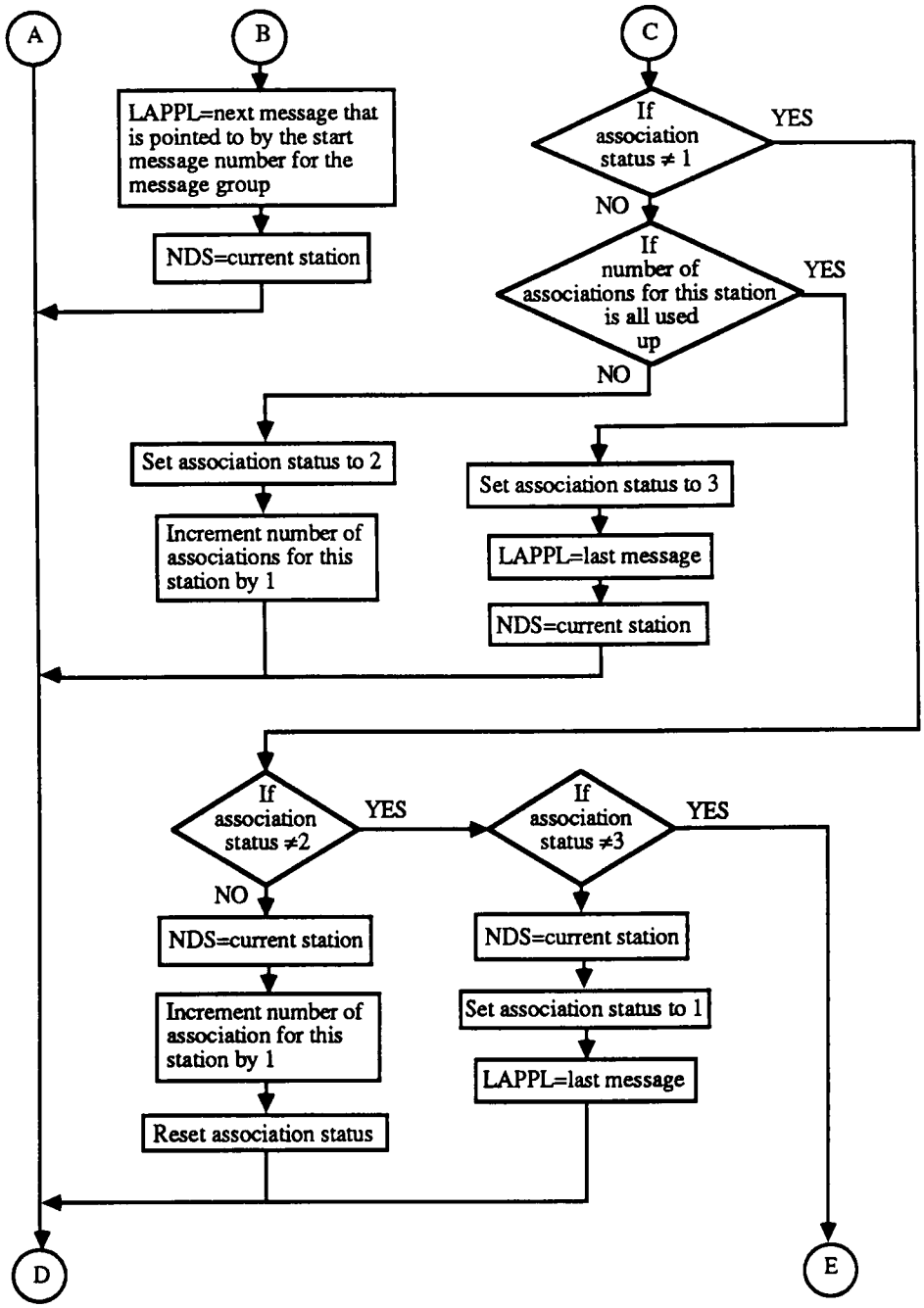


Figure B-6. (continued)

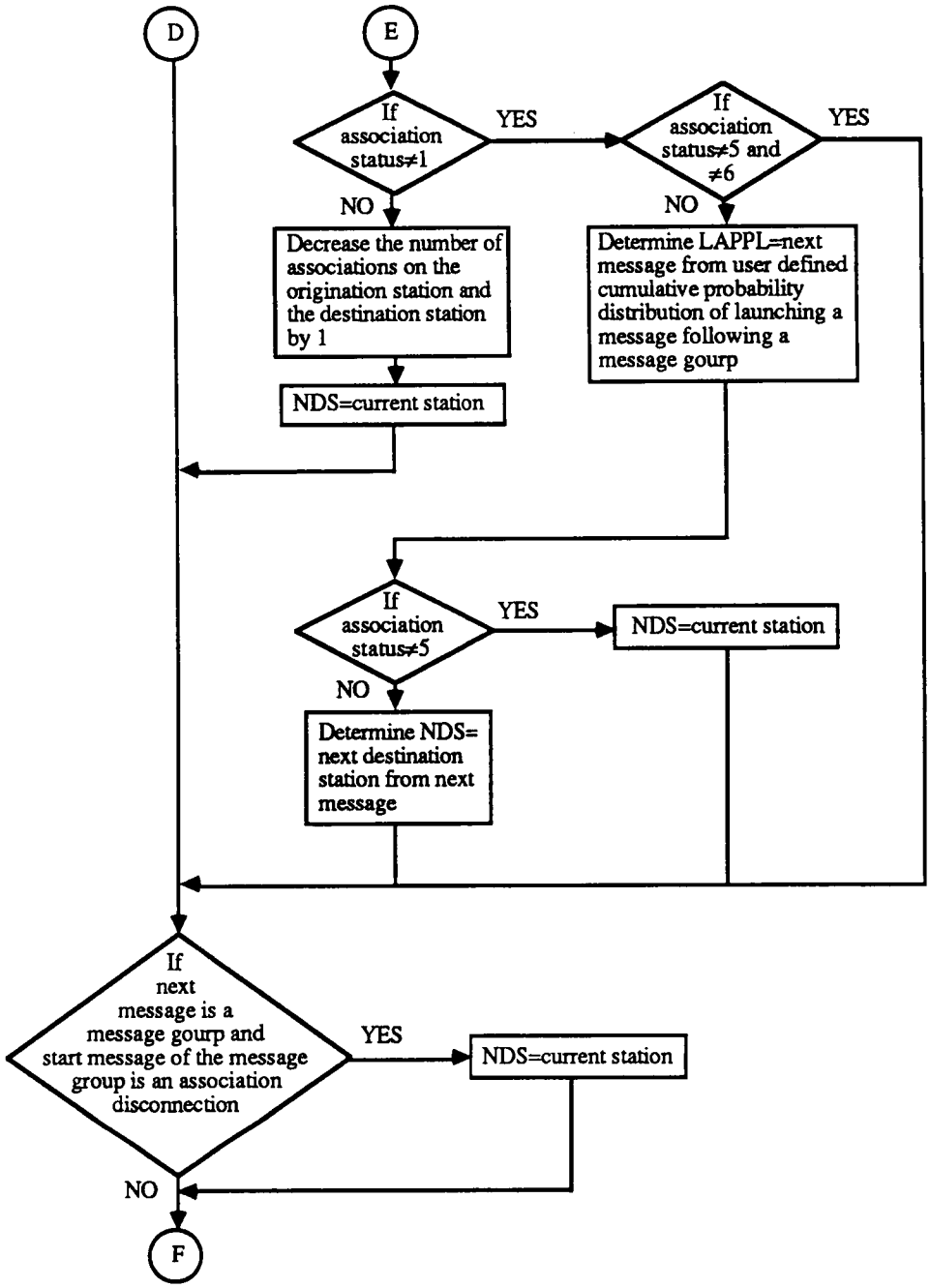


Figure B-6. (continued)

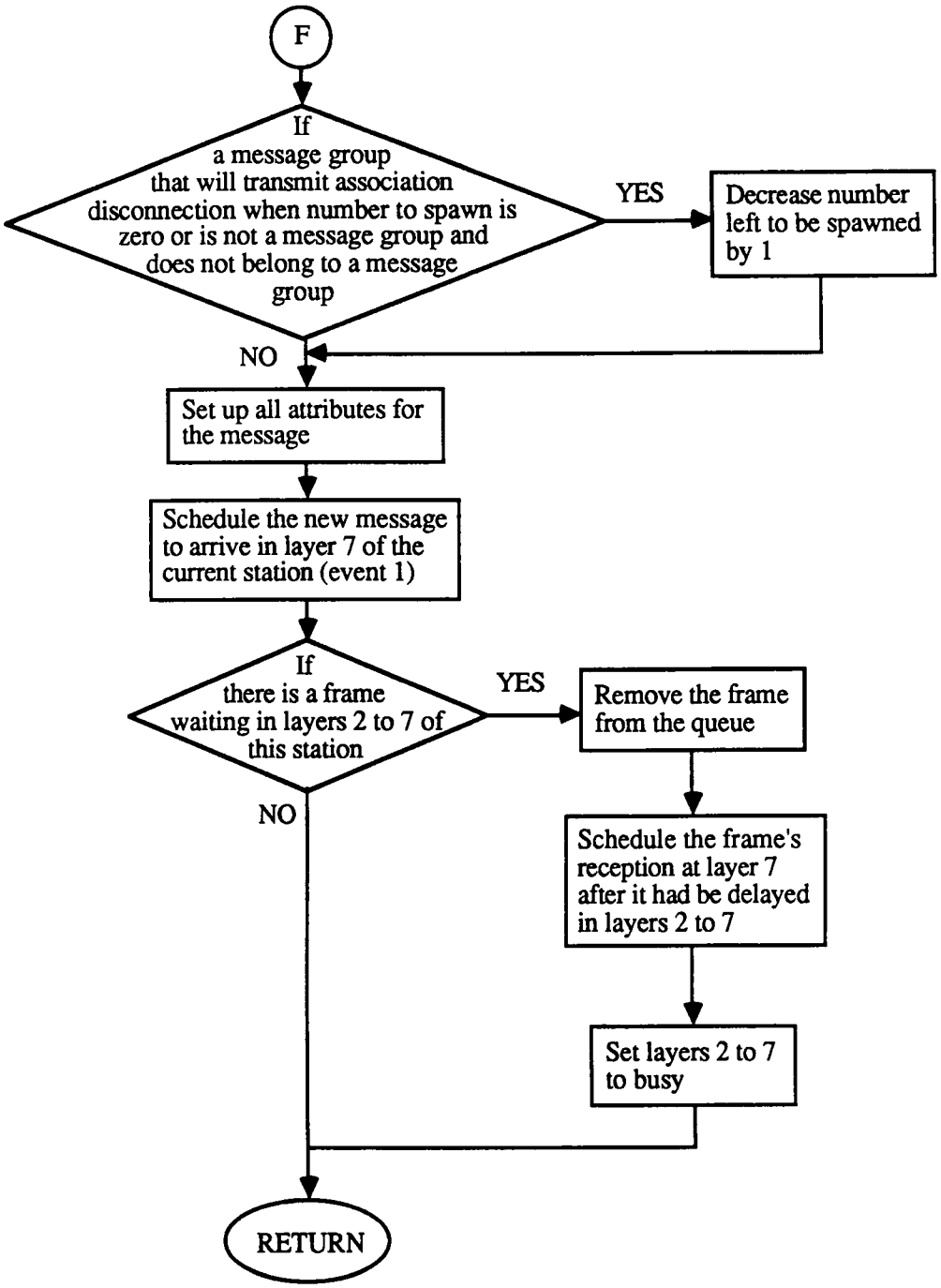


Figure B-6. (continued)

association status of the new message is set to 3 to indicate an association denial. LAPPL is set to be the same as the last message and NDS is set to be the current station.

If association status is not 1 but 2, an association has been granted, then NDS is set to the current station. The number of associations for this station (the station that requested the association) is increased by 1 and the association status is reset. An association status of 3 indicates that an association denial has been transmitted back to the current station (the station that requested the association). Thus, the current station must try to make association again by transmitting another association request. If the association status is 3, NDS is set to the current station, the association status is set to 1 and LAPPL is set to be the last message.

If the association request is 4, an association disconnection is requested. Since for simulation purposes, the request for disconnection is always granted, the number of associations is reduced by 1 from the origination and destination stations. NDS is set to be the current station. An association status of either 5 or 6 indicates that either the end of the association connection establishment is reached or the end of the association disconnection is reached. If so, the next message is determined from PLAUN, which is the user-defined cumulative probability distribution of launching a message following a message group.

If the association status is 5, then NDS is set to the current station. Since the association has just been established, the station that requested the association can now transmit data to the current station which is the destination station. If the association

status is 6, then NDS is set to be the next destination station from NPDST, which is the user-defined cumulative probability distribution of the destination station. Since the end of disconnection is reached, the message may be transmitted to another destination station according to the user-defined probability distribution, NPDST.

After establishing parameters for the next message according to the association status, it is determined whether the next message is a message group and whether the start of the message group is an association disconnection. If so, NDS is set to the current station so that the current association established for the last frame transmitted can be disconnected. In any case, if the entity is a message group which will transmit disassociation when the number left to be spawned is greater than zero but is transmitting a normal message now, or if the entity is not a message group, and if the entity is a frame that does not belong to a message group, then the number of messages left to be spawned is decreased by 1.

Note that it is possible to launch a disconnection even if the number left to be spawned is zero. In any case, all the attributes for the new message entity are set up and event 1 is scheduled for the new message to arrive in layer 7 of the current station. If there is a frame waiting in the reception stack of this station, then the frame is removed from the reception stack queue. The reception of the frame at layer 7 is scheduled after it has been delayed in the reception stack. The reception stack is set to busy. In any case, the execution is returned to SIMAN.

APPENDIX C

AUTOLISP PROCEDURES

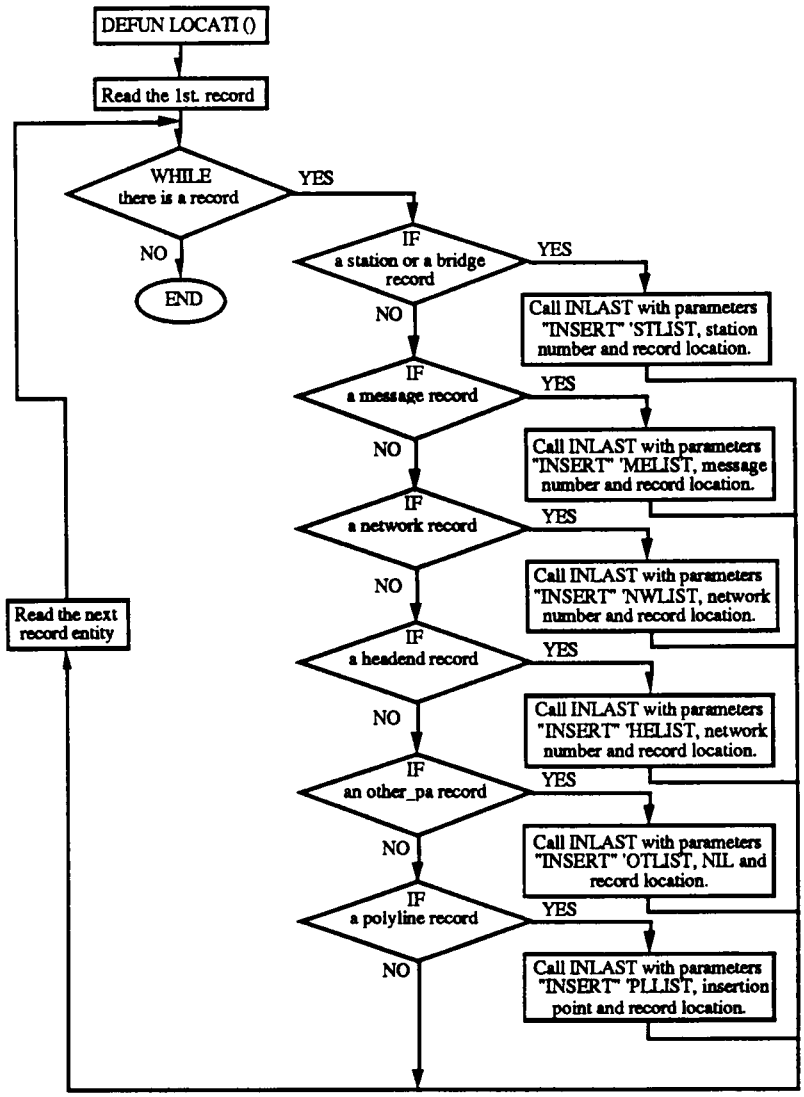


Figure C-1. LOCATI Procedure

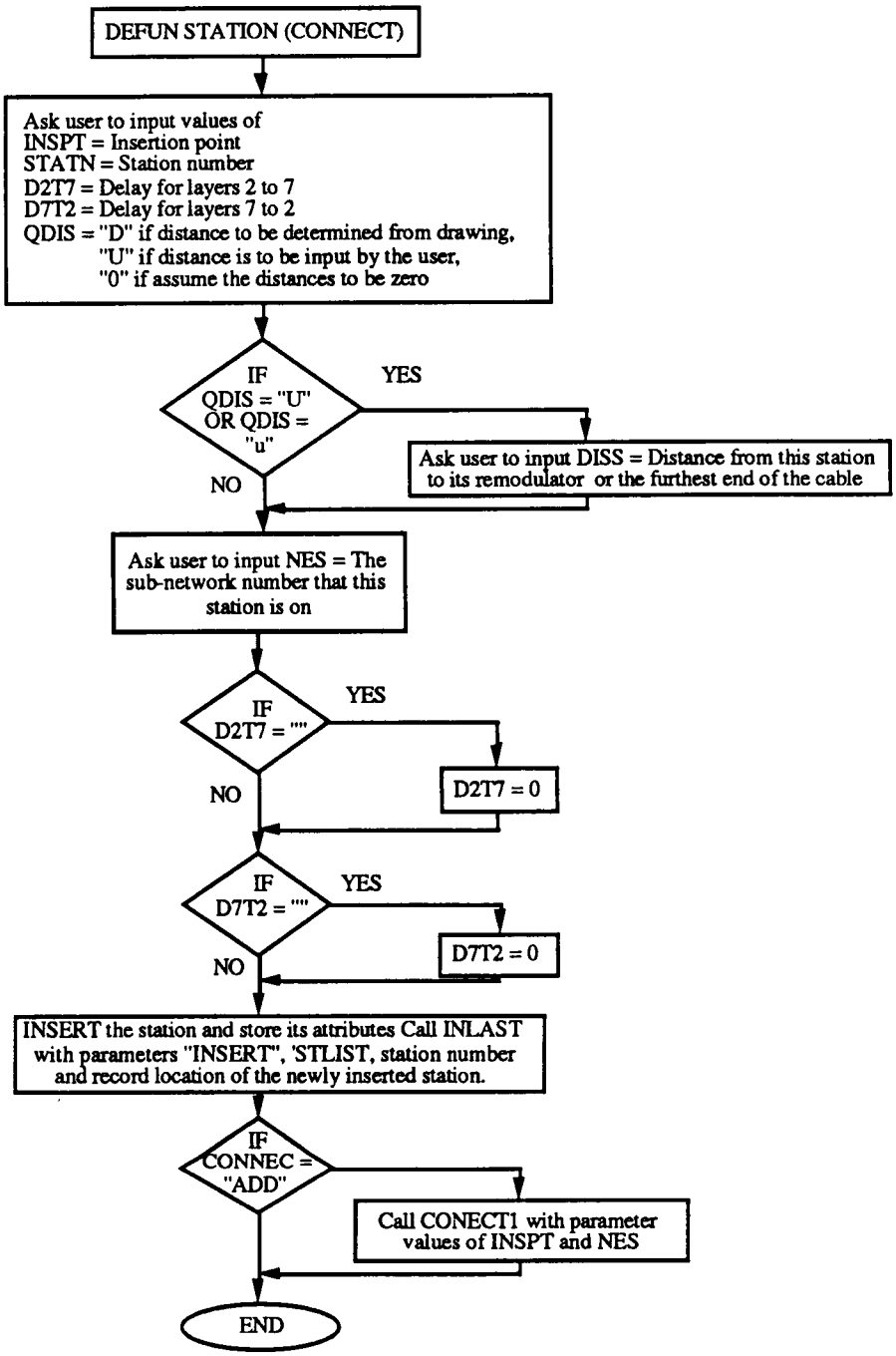


Figure C-2. STATION Procedure

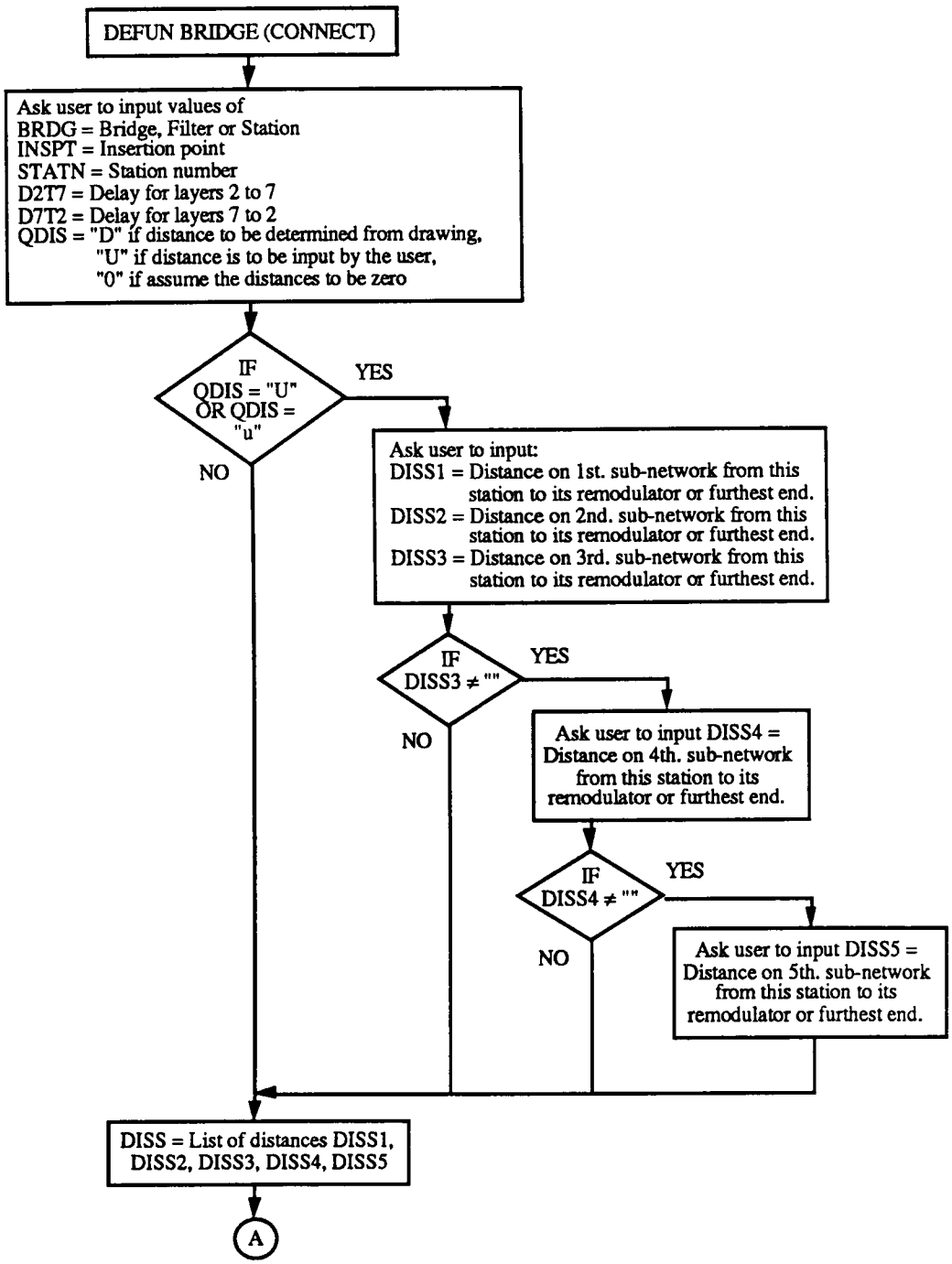


Figure C-3. BRIDGE Procedure

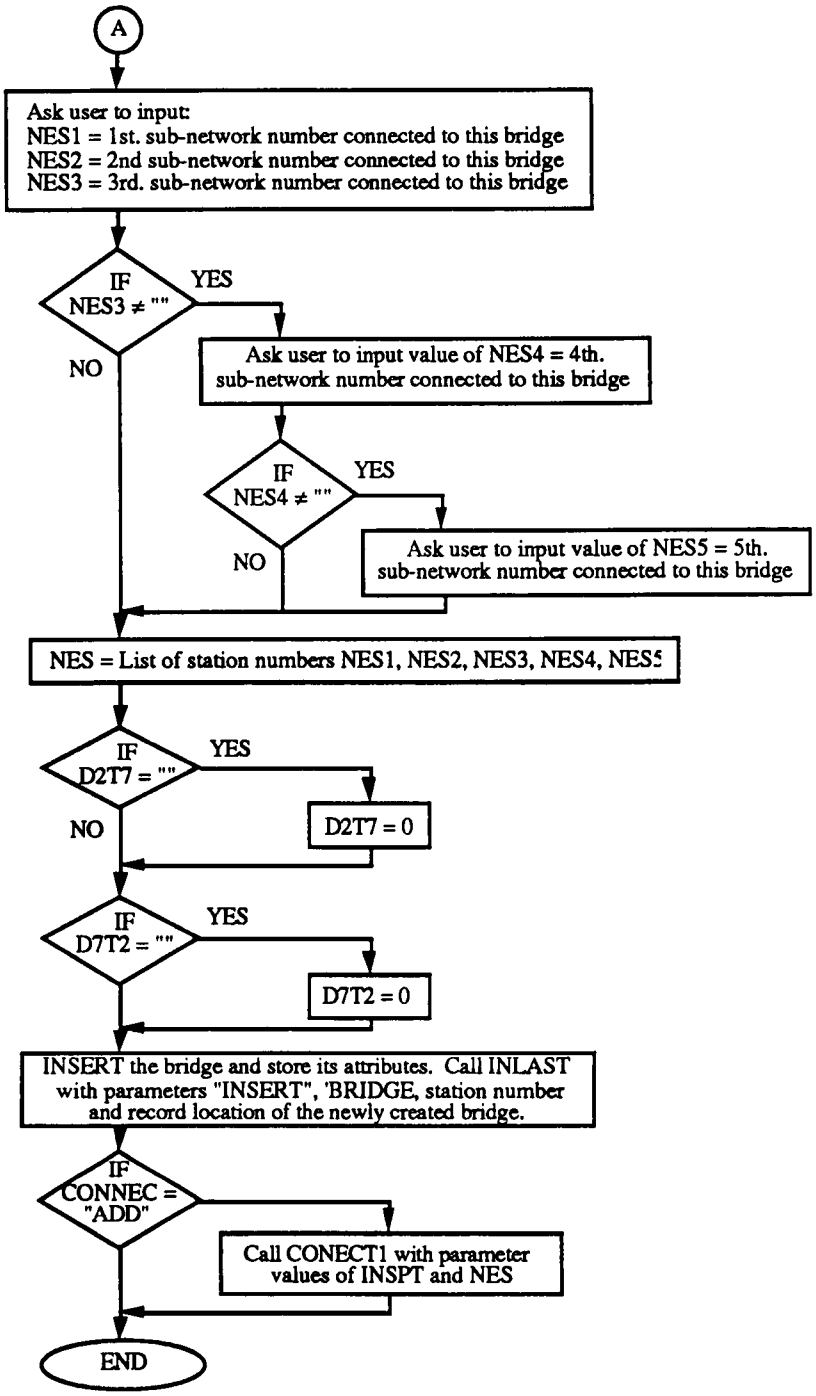


Figure C-3. (Continued)

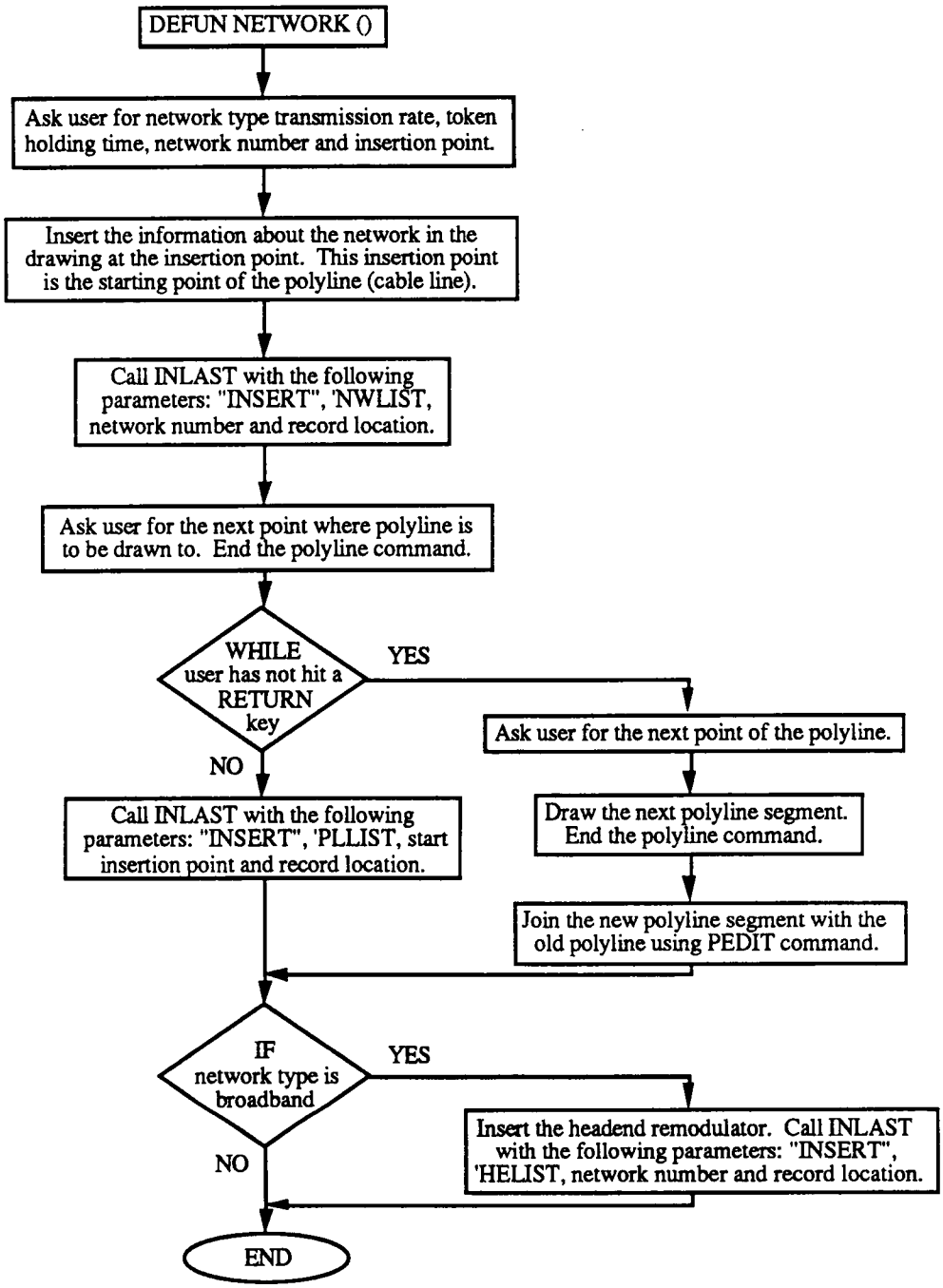


Figure C-4. NETWORK Procedure

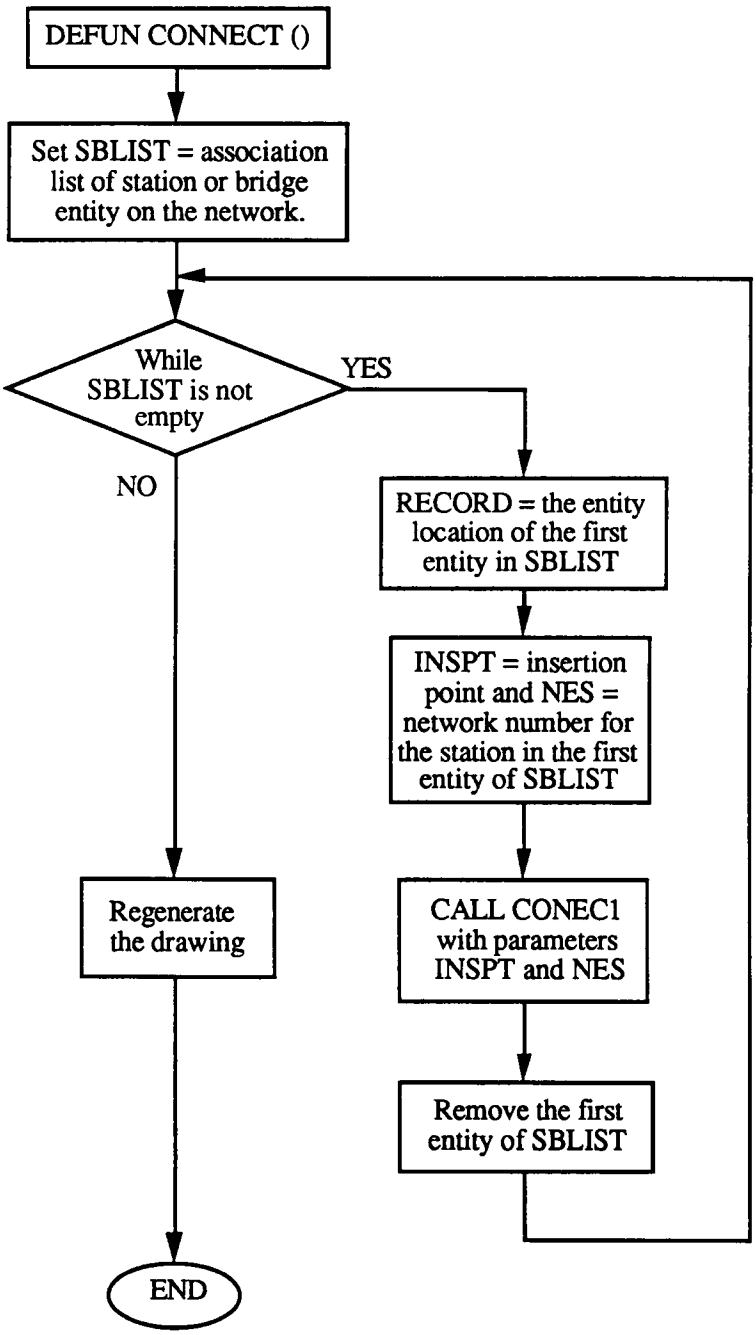


Figure C-5. CONNECT Procedure

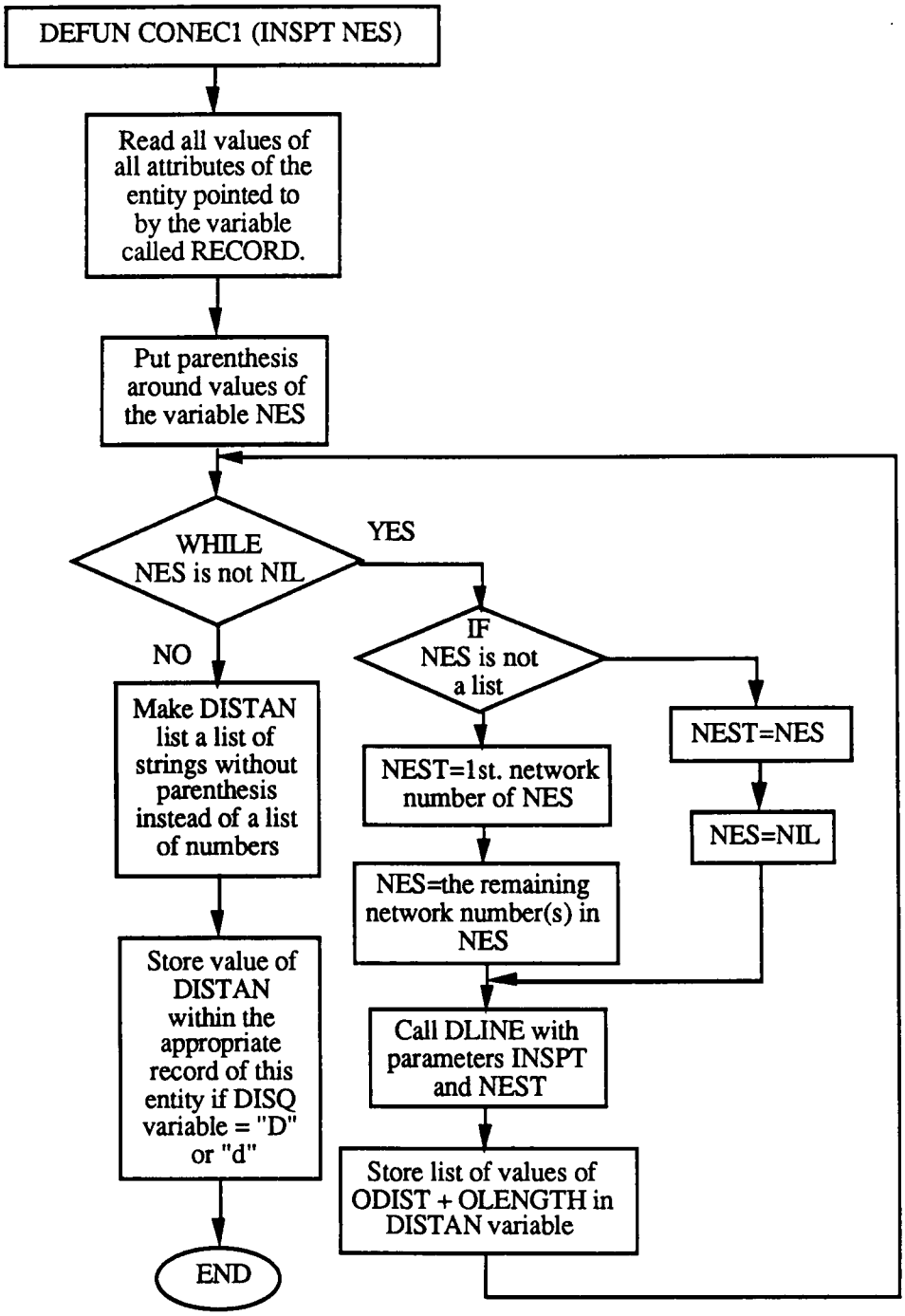


Figure C-6. CONEC1 Procedure

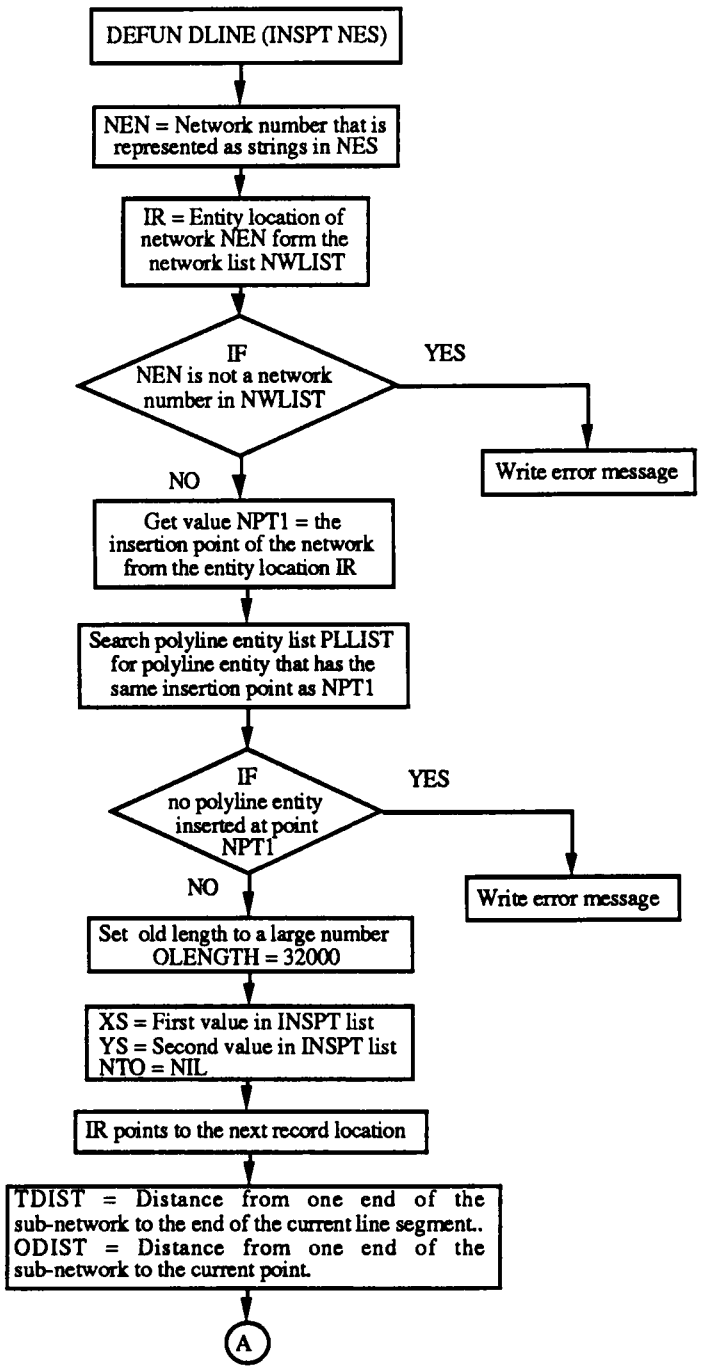


Figure C-7. DLINE Procedure

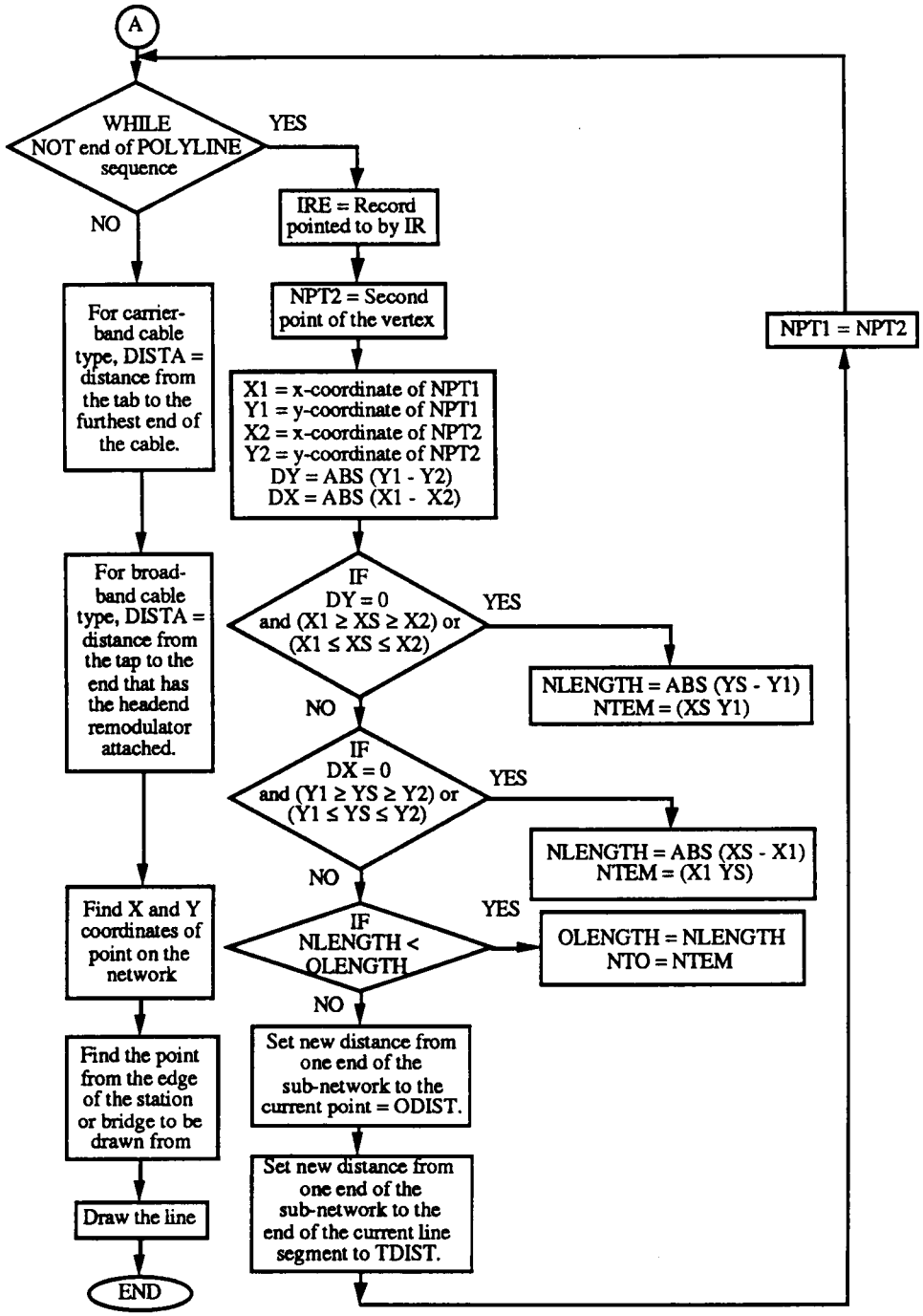


Figure C-7. (Continued)

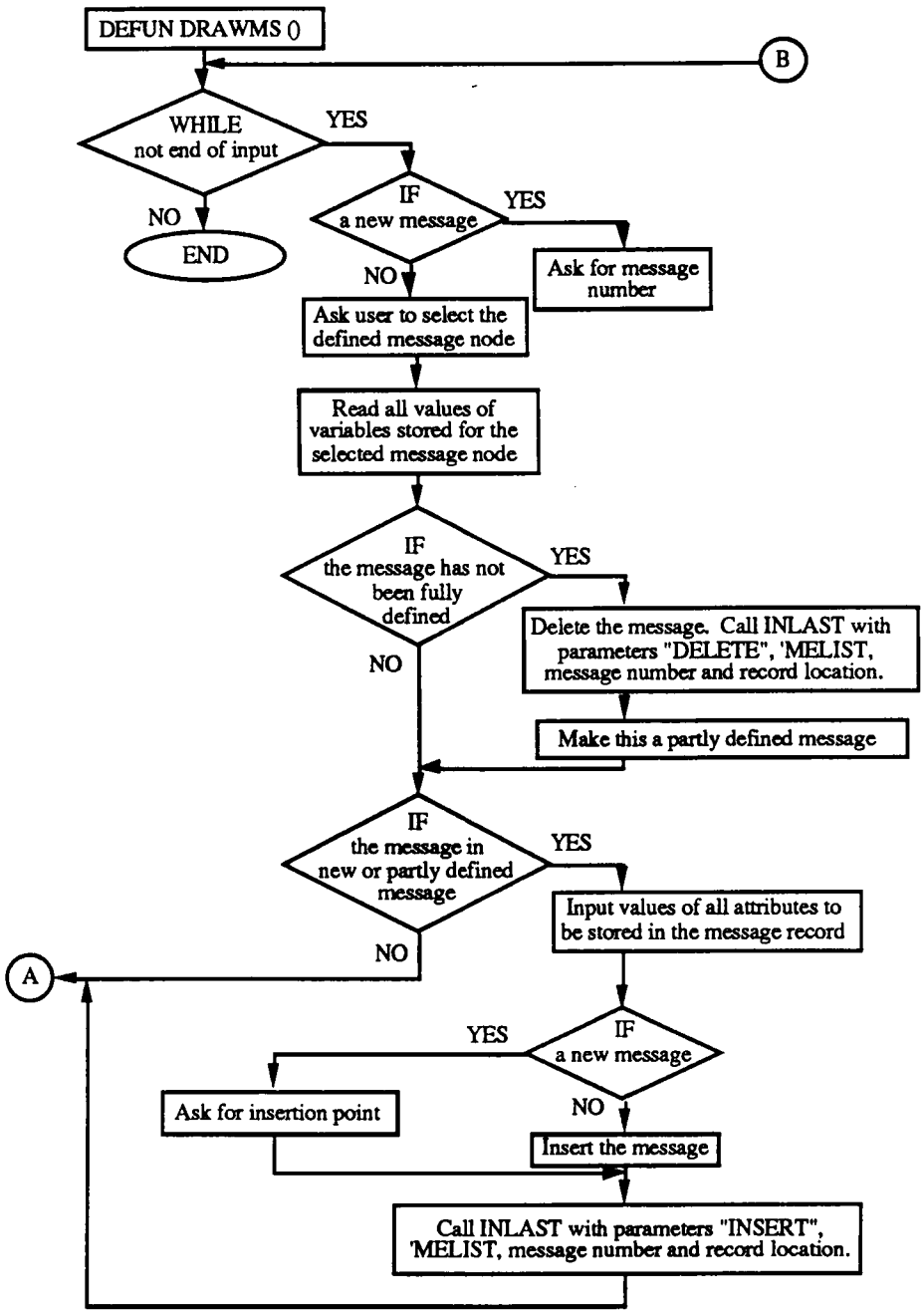


Figure C-8. DRAWMS Procedure

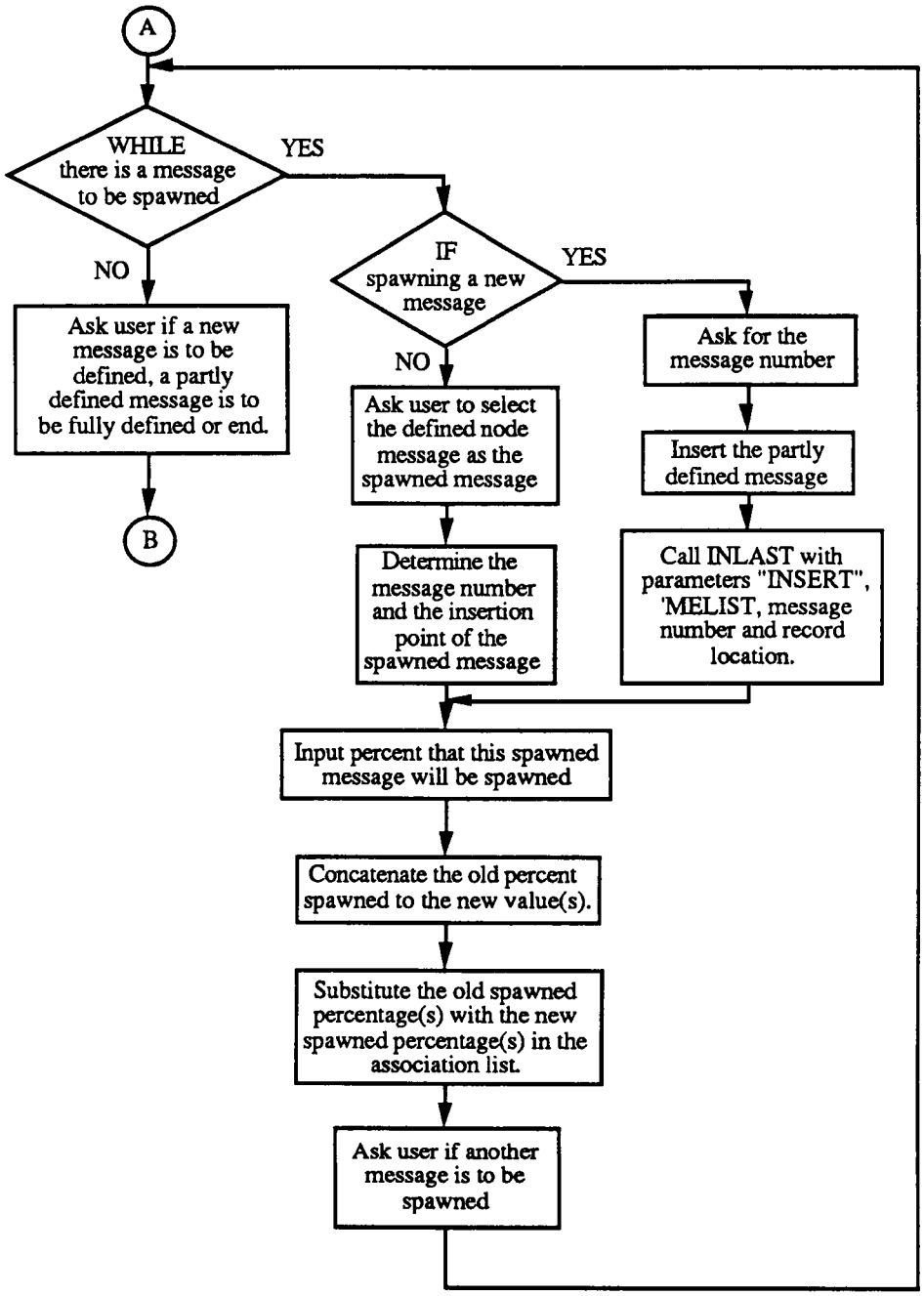


Figure C-8. (Continued)

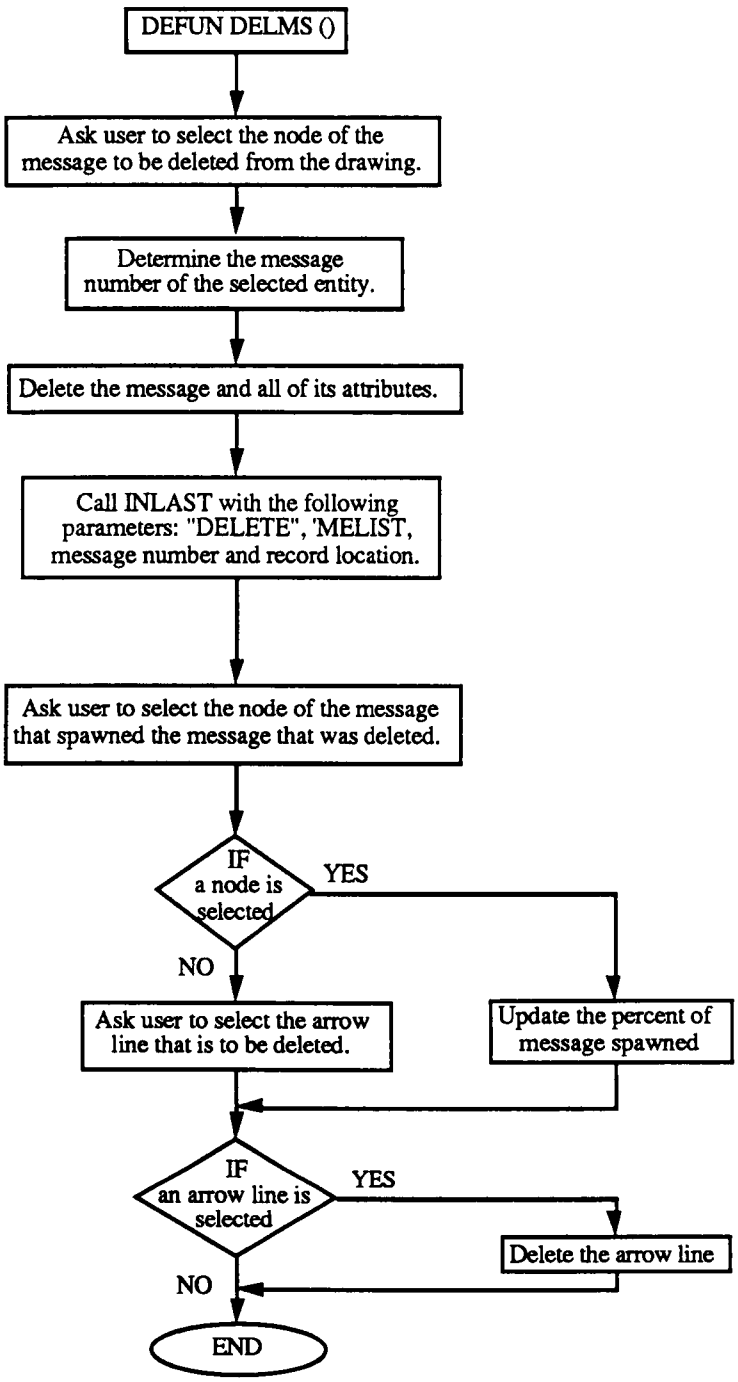


Figure C-9. DELMS Procedure

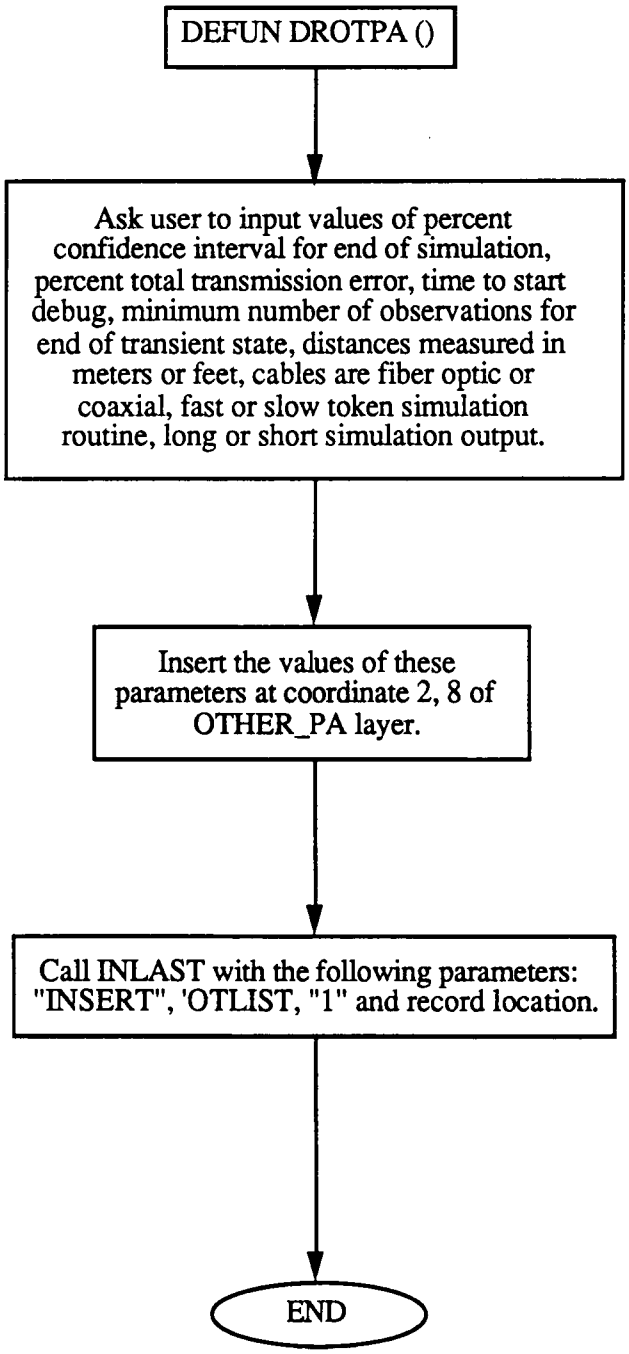


Figure C-10. DROTPA Procedure

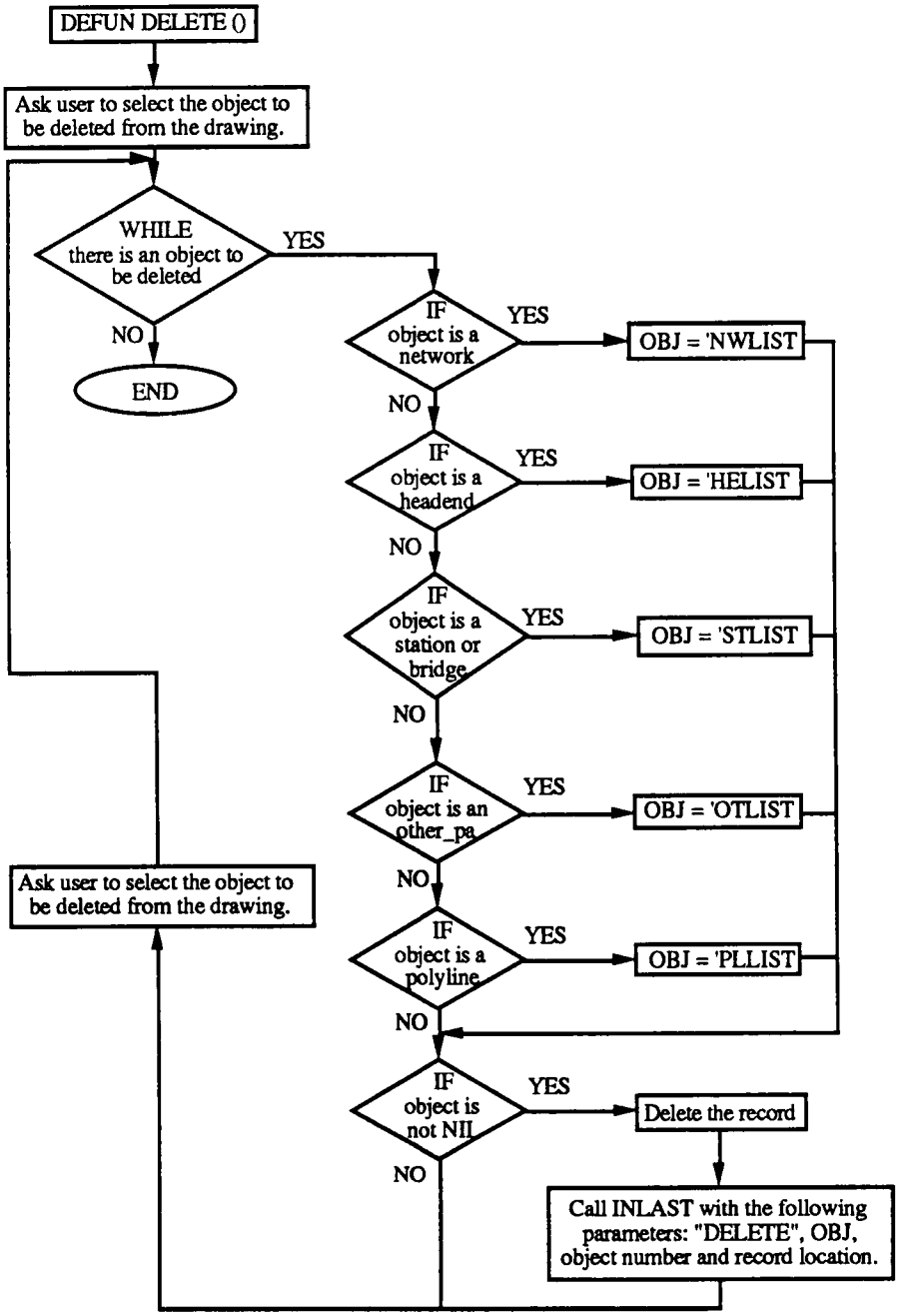


Figure C-11. DELETE Procedure

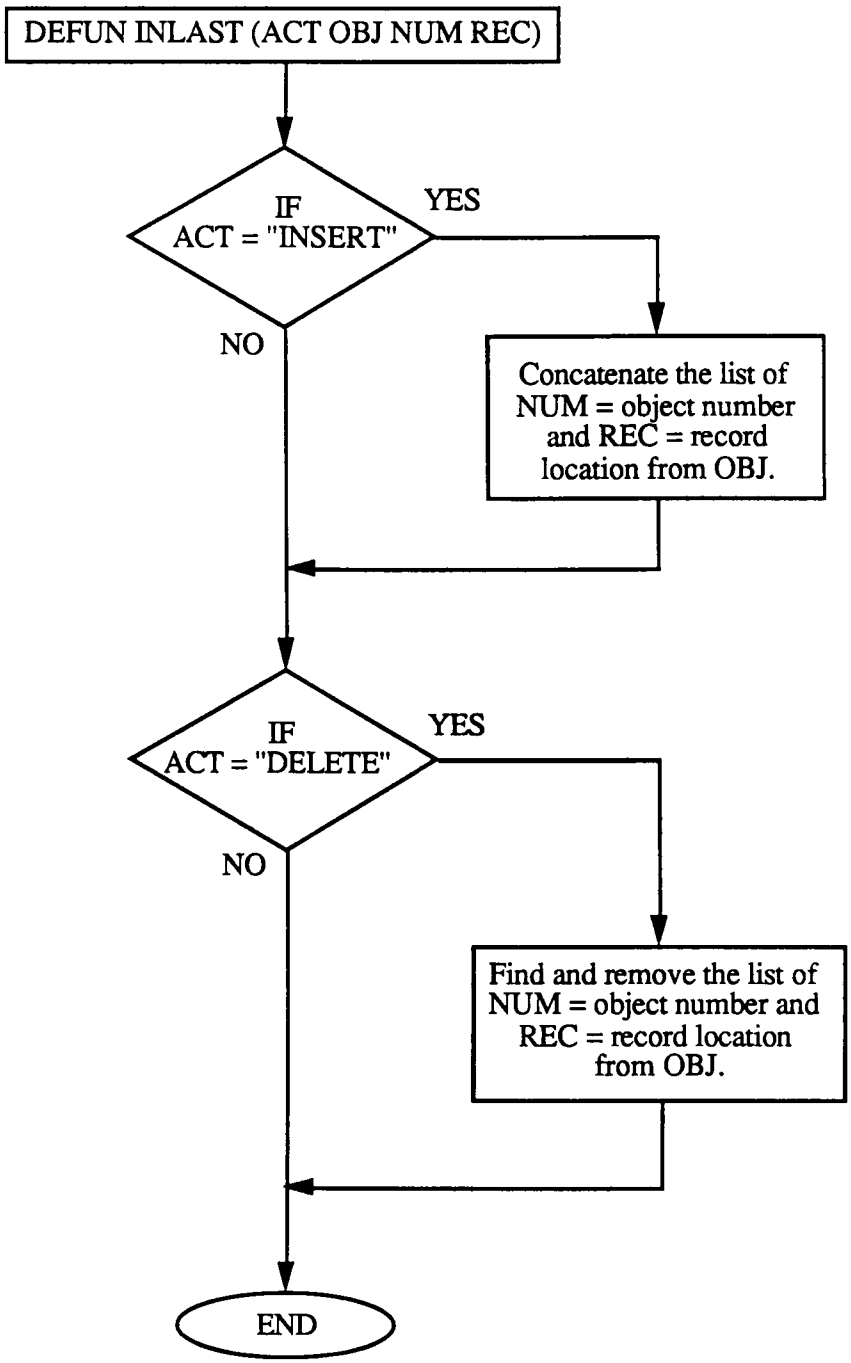


Figure C-12. INLAST Procedure

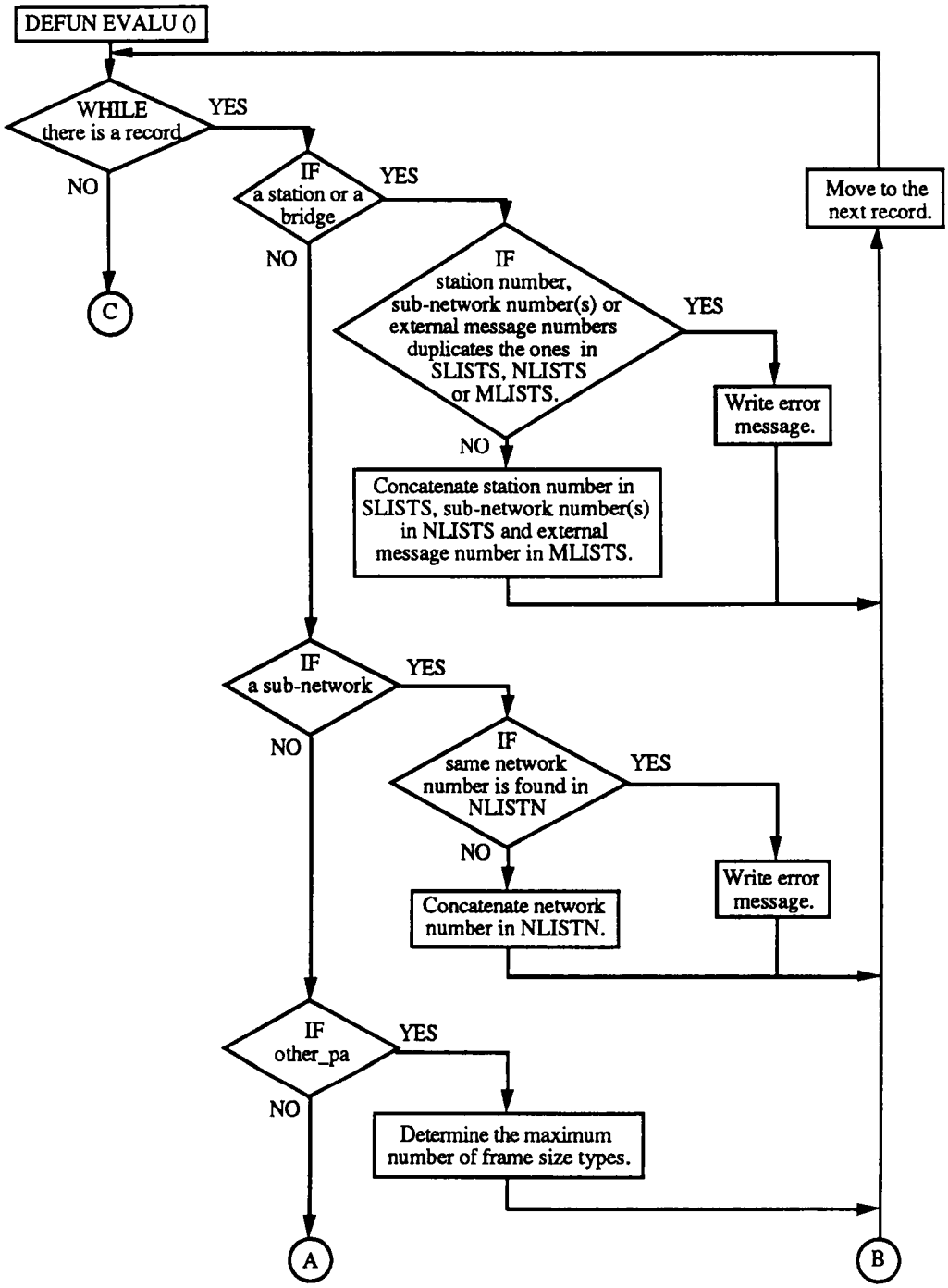


Figure C-13. EVALU Procedure

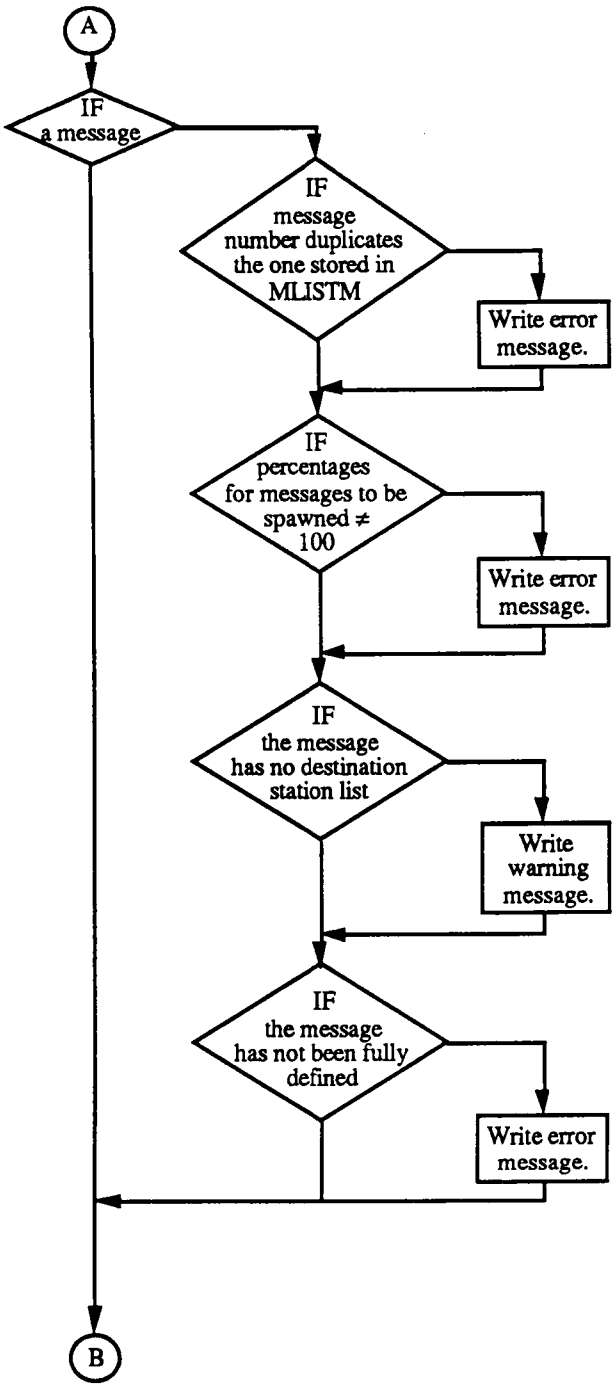


Figure C-13. (Continued)

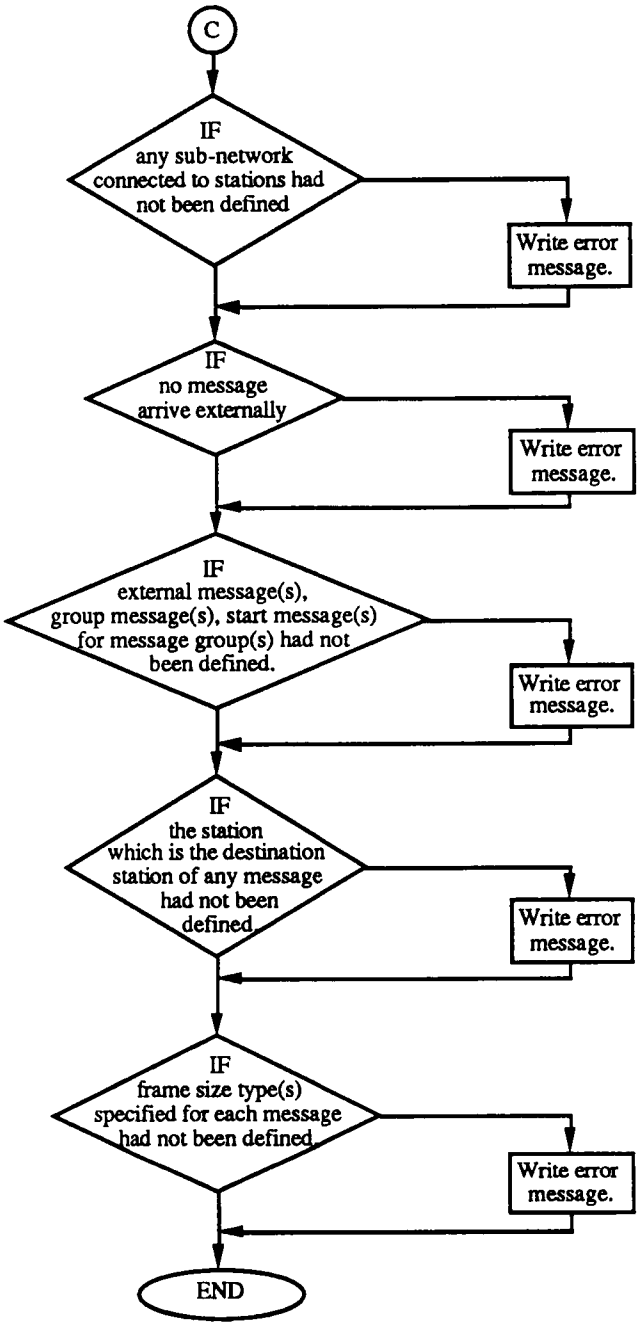


Figure C-13. (Continued)

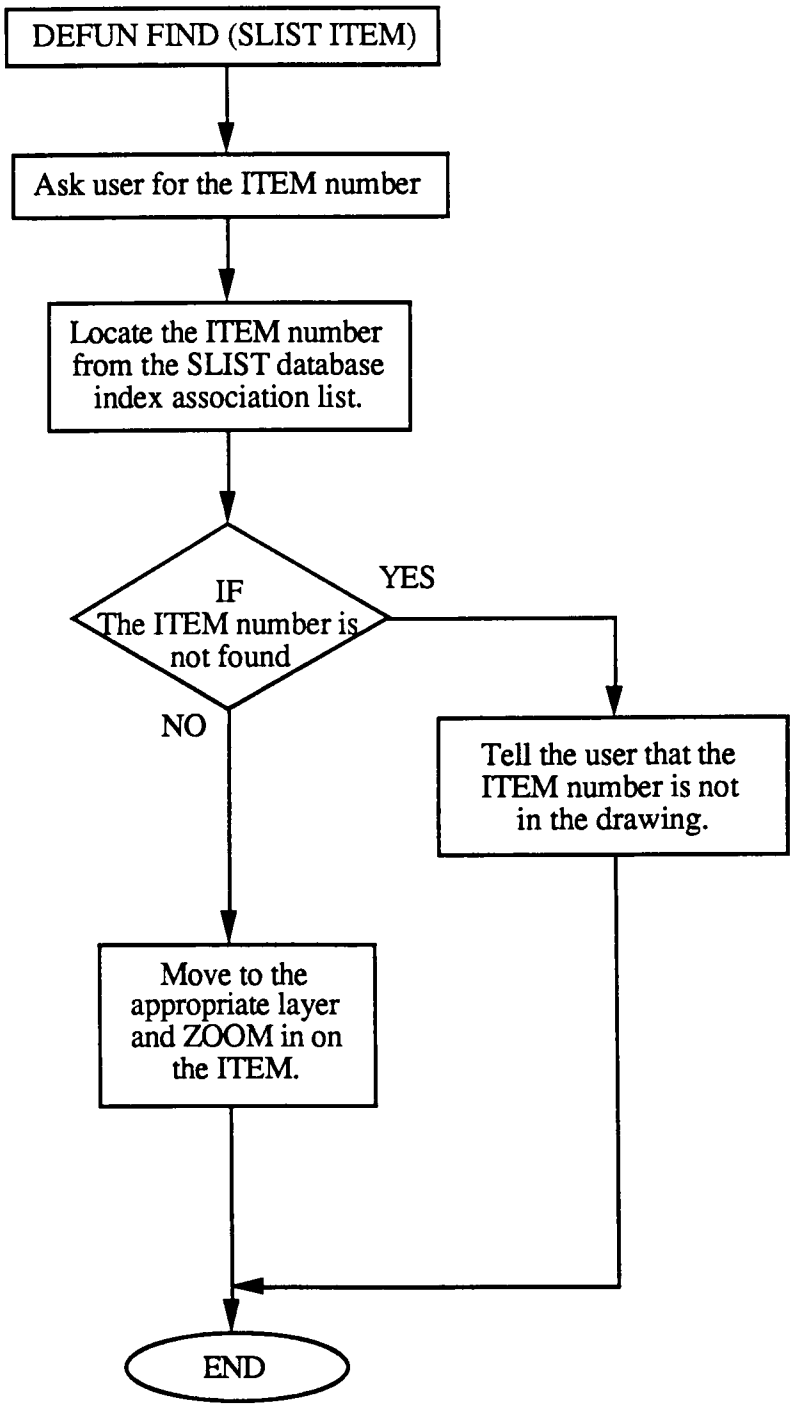


Figure C-14. FIND Procedure

APPENDIX D

CUSTOMIZED MENU FILE LISTING

```
***BUTTONS
;
;
;
^B
^O
^G
^D
^E
^T
***SCREEN
**S
[I L A N]$S=X $S=S
[* * * *]$S=OSNAPB
[DISPLAY]$S=X $S=DS

[EVALUATE]^C^CEVALUATE

[HARDWARE]$S=X $S=HARD ^C^C(CHLYER "HARDWARE")

[MESSAGE]$S=X $S=MESS ^C^C(CHLYER "MESSAGE")

[OTHER_PA]$S=X $S=OTPA ^C^C(CHLYER "OTHER_PA")

[SIMULATE]^C^CSIMULATE

[SETTINGS]$S=X $S=SET
[PLOT]$S=X $S=PLOT
[UTILITY]$S=X $S=UT
[QUIT]$S=X $S=QUIT ^C^CQUIT
[SAVE]^C^CSAVE
[EXIT:]^C^CEXIT
**HARD 3

[HARDWARE:]


[DISPLAY]$S=X $S=DS
```

[DRAW_HW]$S=X $S=DRHW (CHLYER "HARDWARE")
[EDIT_HW]$S=X $S=EDHW (CHLYER "HARDWARE")

[PLOT]$S=X $S=PLOT

**MESS 3

[MESSAGE:]

[DISPLAY]$S=X $S=DS
[DRAWMS]^C^CDRAWMS
[EDIT_MS]$S=XMS $S=EDMS ^C^C(CHLYER "MESSAGE")

[PLOT]$S=X $S=PLOT

**OTPA 3

[OTHER_PA:]

[DISPLAY]$S=X $S=DS
[DROTPA]^C^CDROTPA
[EDITOTPA]$S=XMS $S=EDPA ^C^C(CHLYER "OTHER_PA")

[PLOT]$S=X $S=PLOT

**X 3

```
[__LAST__]$S= $S=
[ DRAW_HW]^C^C$S=X $S=DRHW
[ EDIT_HW]^C^C$S=X $S=EDHW
**XMS 3
```

```
[__LAST__]$S= $S=

**DS 3
[ATTDISP:]$S=X $S=ATTDISP ^C^CATTDISP

[HARDWARE]^C^C(CHLYER "HARDWARE")
[MESSAGE]^C^C(CHLYER "MESSAGE")
[OTHER_PA]^C^C(CHLYER "OTHER_PA")

[FIND-MS]FIND_MS
[FIND_ND]FIND_ND
[FIND_NW]FIND_NW

[PAN:]'PAN
[REDRAW:]'REDRAW
[REGEN:]^C^CREGEN
[ZOOM:]$S=X $S=ZOOM 'ZOOM
**DRHW 3
[DRAW_HW:]


[NODE-1N]^C^CNODE-1N

[NODE-mN]^C^CNODE-MN

[HEADEND]^C^C(DRHEAD)

[NETWORK]^C^CNETWORK
```

```
[CONECT]^C^CCONECT


**EDHW 3
[EDIT_HW:]


[ATTEDIT:]$S=X $S=ATTEDIT ^C^CATTEDIT

[DELETE]^C^C^CDELETE

[ADD_DEV:]$S=X $S=ADD_DEV

[MOVE]^C^CMOVE

[UNDO:]$S=X $S=UNDO ^C^CUNDO

**EDMS
[EDIT_MS:]


[ATTEDIT:]$S=X $S=ATTEDIT ^C^CATTEDIT

[DELMS]^C^C^C(C:DELMS)

[MOVE]^C^CMOVE

[STRETCH]^C^CSTRETCH

[UNDO:]$S=X $S=UNDO ^C^CUNDO

**EDPA 3
[EDITOTPA]

[ATTEDIT:]$S=X $S=ATTEDIT

[CHANGEPA]^C^CATTEDIT ;;;;

[DELETE]^C^C^CDELETE

[MOVE]^C^C MOVE

[UNDO:]$S=X $S=UNDO ^C^CUNDO


**SET 3
```

```
[DDEMODES]'DDEMODES
[DDRMODES]'DDRMODES
[--------]

[APERTUR:]$S=X $S=APERTURE ^C^CAPERTURE
[COORDS:]$S=X $S=COO ^C^C'SETVAR;COORDS;
[GRID:]$S=X $S=GRID ^C^CGRID
[LIMITS:]$S=X $S=LIMITS ^C^CLIMITS
[LTSCALE:]^C^CLTSCALE
[OSNAP:]$S=X $S=OSNAPC ^C^COSNAP
[SNAP:]$S=X $S=SNAP ^C^CSNAP

[UNITS:]^CUNITS


**PLOT 3

[ C   D]
[ H   E]
[ O   V]
[ O   I]
[ S   C]
[ E   E]

[PLOTTER]^C^CPLOT

[PRINTER]^C^CPRPLOT
**UT 3


[FILES:]^C^CFILES

[MENU:]$S=X $S=CHTEMP ^C^CMENU




[External]$S=X $S=EXCOMDS
[Commands]$S=X $S=EXCOMDS

[END]$S=X $S=END
[QUIT]$S=X $S=QUIT ^C^CQUIT
**OSNAPB 3
[CENter]CENTER $S=
[ENDpoint]ENDPOINT $S=
[INSert]INSERT $S=
```

```
[INTersec]INTERSEC $S=
[MIDpoint]MIDPOINT $S=
[NEArest]NEAREST $S=
[NODe]NODE $S=
[PERpend]PERPEND $S=
[QUAdrant]QUADRANT $S=
[QUICK,]QUICK,^Z
[TANgent]TANGENT $S=
[NONE]NONE $S=
[CANCEL:]^C^C$S=
[U:]^C^CU $S=
[REDO:]^C^CREDO $S=
[REDRAW:]'REDRAW $S=
['SETVAR]$S= 'SETVAR
[__LAST__]$S=
**INSERT 3
[INSERT:]^C^CINSERT
?
[*]*\
corner
xyz
drag

Scale
Xscale
Yscale Zscale
Rotate
**ZOOM 3

[ZOO:]ZOOM




All

[Center]C

[Dynamic]D

Extents

Left

Previous

Window
```

```
Yes

No

**ATTDISP 3

[ATTDISP:]^C^CATTDISP


Normal

ON

OFF

**RGNAUTO 3

[RGNAUTO:]^C^CREGENAUTO


[ON]On $S=X $S=DS

[OFF]Off $S=X $S=DS

**LIMITS 3


[LIMITS:]^C^CLIMITS


ON

OFF

**END 3

END ?


[Yes]^C^CEND
```

[No]$S=X $S=S

**QUIT 3

[QUIT:]^C^CQUIT

[Yes]Y

[No]N $S=X $S=S

**SNAP 3

[SNAP:]^C^CSNAP

ON

OFF

[Aspect]$S=X ASPECT

[Rotate]$S=X ROTATE

[Style]S $S=X $S=SNAPSTYLE

**SNAPSTYLE 11 Iso

Standard

**OSNAPC 3

[OSNAP:]^C^COSNAP

[CENter]CENTER\

[ENDpoint]ENDPOINT\

[INSert]INSERT\

[INTersec]INTERSEC\

[MIDpoint]MIDPOINT\

[NEArest]NEAREST\

[NODe]NODE\

[PERpend]PERPEND\

[QUAdrant]QUADRANT\

[TANgent]TANGENT\

[NONE]^C^COSNAP NONE $S= $S=

[Quick,]QUICK,^Z

[,],\

**COO 3

[COORDS:]^C^C'SETVAR;COORDS;

[ON]1

[OFF]0

**GRID 3

[GRID:]^C^CGRID

[grd=snap]SNAP

ON

OFF

Aspect

**APERTURE 3

[APERTUR:]'SETVAR APERTURE

[PICKBOX:]'SETVAR PICKBOX

[1]$S= $S= 1

[2]$S= $S= 2

[3]$S= $S= 3

[4]$S= $S= 4

[5]$S= $S= 5

[6]$S= $S= 6

[7]$S= $S= 7

[8]$S= $S= 8

[9]$S= $S= 9

[10]$S= $S= 10

[15]$S= $S= 15

**ADD_DEV 3

[ADD_DEV]

[ANODE-1N]^C^CANODE-1N

[ANODE-mN]^C^CANODE-MN

**UNDO 3

[UNDO:]^C^CUNDO

[Back 1]^C^CUNDO 1

Auto

Back

End

Group

Mark

[Control]$S=X $S=UNDOC Control

Yes

No

[REDO:]REDO

**UNDOC 11

All

None

One

**ERASE 3

[ERASE:]^C^CERASE

Window

Last

Previous

Crossing

Remove

Add

Undo

[OOPS:]^C^COOPS

**ATTEDIT 3

[ATTEDIT:]^C^CATTEDIT

Window

Last

Previous

Crossing

Remove Add

Undo

[Value]Value $S=X $S=ATTEDIT2

Position

Height

Angle

Style

Layer

[Colr] $S=X $S=ATTCOLOR C

Nex

[__LAST__]$S= $S=

**ATTEDIT2 3

[Change]Change $S=X $S=ATTEDIT

[Replace]Replace $S=X $S=ATTEDIT

**ATTCOLOR 3

[COLOR]

[red]RED $S= $S=

[yellow]YELLOW $S= $S=

[green]GREEN $S= $S=

[cyan]CYAN $S= $S=

[blue]BLUE $S= $S=

[magenta]MAGENTA $S= $S=

[white]WHITE $S= $S=

**UNITS 2

[UNIT]

[TYPE]

[scientfc]1;

[decimal]2;

[enginrng]3;

[archtect]4;

[metric]5;

**ENGLISH 2

[Horizntl]

[Sheet]

[Size]

[A-8.5x11]11.0 8.5;

[B- 11x17]17.0 11.0;

[C- 22x34]34.0 22.0;

[18x24]24.0 18.0;

[D- 24X36]36.0 24.0;

[30x42]42.0 30.0;

[E- 36x42]42.0 36.0;

[OTHER >]^P0 0;^P

[VERTCAL>]$S=X $S=ENGVERT

**ENGVERT 2

[Vertical]

[Sheet]

[Size]

[A-8.5x11]8.5 11.0;

[B- 11x17]11.0 17.0;

[C- 22x34]22.0 34.0;

[18x24]18.0 24.0;

[D- 24X36]24.0 36.0;

[30x42]30.0 42.0;

[E- 36x42]36.0 42.0;

[OTHER >]^P0 0;^P

[HORZNTL>]$S=X $S=ENGLISH

**METRIC 2

[Horizntl]

[Sheet]

[Size]

[1189x841]1189.0 841.0;

[1000x707]1000.0 707.0;

[841x594]841.0 594.0;

[594x420]594.0 420.0;

[420x297]420.0 297.0;

[297x210]297.0 210.0;

[OTHER >]^P0 0;^P

[VERTCAL>]$S=X $S=METRICVERT

**METRICVERT 2

[Vertical]

[Sheet]

[Size]

[1189x841]841.0 1189.0;

[1000x707]707.0 1000.0;

[841x594]594.0 841.0;

```
[ 594x420]420.0  594.0;

[ 420x297]297.0  420.0;

[ 297x210]210.0  297.0;


[OTHER  >]^P0 0;^P



[HORZNTL>]$S=X $S=METRIC



**CHTEMP 3

[Change  ]

[Template]


[Arch   >]/aec/a/m/aec-a

[Mech   >]/aec/m/m/aec-m

**EXCOMDS 3

[CATALOG]CATALOG

[DEL:]DEL

[DIR:]DIR

[EDIT:]EDI


[SH:]SH
```

[SHELL]SHELL

[TYPE:]TYPE

APPENDIX E

SIMULATION OUTPUT

```
:-------------------------------------------------------------------:
:                                                                   :
:                                                                   :
:                        SIMAN Run Processor                        :
:                        OS/2  Version 3.52                         :
:                       License Number 8907024                      :
:                                                                   :
:-------------------------------------------------------------------:
:                                                                   :
:   Systems Modeling Corporation licenses this program for use by :  :
:                                                                   :
:                         Univ. of Tennessee                        :
:                                                                   :
:   This program may only be used or copied according to the terms  :
:   of that license.                                                :
:                                                                   :
:-------------------------------------------------------------------:
```

Please press <return> to begin the simulation.

Recalling the PROGRAM file ILAN.p

SIMAN Run Processor
OS/2 Version 3.52
Copyright(C) 1989 by Systems Modeling Corp.
All Rights Reserved

Beginning execution of run number 1

BATCH NSETO= 32000
BATCH NSETO= 64000

TOTAL STATISTICS COLLECTION TIME= 1.409850E+09

NUMBER OF FRAMES ARRIVED TO BE SENT AT NETWORK 1= 115273

TOTAL NO. OF FRAMES ARRIVED TO BE SENT = 115273

NUMBER OF FRAMES SENT FROM NETWORK 1 = 115273

TOTAL NUMBER OF FRAMES SENT = 115273

UTILIZATION OF NETWORK 1 =.13225190E-01

STATISTICS ON NUMBER OF FRAMES WAITING TO BE TRANSMITTED AT EACH STATION

STATION	MEAN	STD DEV	MINIMUM	MAXIMUM	CURRENT
1	.33710E-03	.18357E-01	.00000E+00	.10000E+01	.00000E+00
2	.32443E-03	.18009E-01	.00000E+00	.10000E+01	.00000E+00
3	.29812E-03	.17264E-01	.00000E+00	.10000E+01	.00000E+00
4	.29404E-03	.17145E-01	.00000E+00	.10000E+01	.00000E+00
5	.50485E-03	.22463E-01	.00000E+00	.10000E+01	.00000E+00
6	.99433E-03	.31517E-01	.00000E+00	.10000E+01	.00000E+00
7	.92244E-03	.30358E-01	.00000E+00	.10000E+01	.00000E+00
8	.10197E-02	.31916E-01	.00000E+00	.10000E+01	.00000E+00
9	.10820E-02	.32877E-01	.00000E+00	.10000E+01	.00000E+00
10	.12506E-02	.35341E-01	.00000E+00	.10000E+01	.00000E+00
11	.11862E-02	.34421E-01	.00000E+00	.10000E+01	.00000E+00
12	.16238E-02	.40263E-01	.00000E+00	.10000E+01	.00000E+00
13	.18683E-02	.43183E-01	.00000E+00	.10000E+01	.00000E+00
14	.74056E-03	.27203E-01	.00000E+00	.10000E+01	.00000E+00
15	.90143E-03	.30010E-01	.00000E+00	.10000E+01	.00000E+00
16	.97760E-03	.31251E-01	.00000E+00	.10000E+01	.00000E+00
17	.91725E-03	.30272E-01	.00000E+00	.10000E+01	.00000E+00
18	.10259E-02	.32013E-01	.00000E+00	.10000E+01	.00000E+00
19	.16482E-02	.40564E-01	.00000E+00	.10000E+01	.00000E+00
20	.16110E-02	.40104E-01	.00000E+00	.10000E+01	.00000E+00
21	.18240E-02	.42670E-01	.00000E+00	.10000E+01	.00000E+00
22	.18503E-02	.42975E-01	.00000E+00	.10000E+01	.00000E+00
23	.20105E-02	.44793E-01	.00000E+00	.10000E+01	.00000E+00
24	.32871E-02	.57239E-01	.00000E+00	.10000E+01	.00000E+00
25	.30525E-02	.55165E-01	.00000E+00	.10000E+01	.00000E+00
26	.30287E-02	.55212E-01	.00000E+00	.20000E+01	.00000E+00
27	.38192E-02	.61682E-01	.00000E+00	.10000E+01	.00000E+00

28	.38121E-02	.61624E-01	.00000E+00	.10000E+01	.00000E+00
29	.44369E-02	.66462E-01	.00000E+00	.10000E+01	.00000E+00
30	.48731E-02	.69637E-01	.00000E+00	.10000E+01	.00000E+00
31	.37796E-02	.61367E-01	.00000E+00	.20000E+01	.00000E+00
32	.50746E-02	.71055E-01	.00000E+00	.10000E+01	.00000E+00
33	.49721E-02	.70337E-01	.00000E+00	.10000E+01	.00000E+00
34	.48589E-02	.69536E-01	.00000E+00	.10000E+01	.00000E+00
35	.59164E-02	.76690E-01	.00000E+00	.10000E+01	.00000E+00
36	.52197E-02	.72059E-01	.00000E+00	.10000E+01	.00000E+00
37	.40392E-02	.63426E-01	.00000E+00	.10000E+01	.00000E+00
38	.37869E-02	.61421E-01	.00000E+00	.10000E+01	.00000E+00
39	.61567E-02	.78223E-01	.00000E+00	.10000E+01	.00000E+00
40	.70927E-02	.83919E-01	.00000E+00	.10000E+01	.00000E+00
41	.53643E-02	.73044E-01	.00000E+00	.10000E+01	.00000E+00
42	.46068E-02	.67732E-01	.00000E+00	.20000E+01	.00000E+00
43	.92185E-02	.95570E-01	.00000E+00	.10000E+01	.00000E+00
44	.92187E-02	.95571E-01	.00000E+00	.10000E+01	.00000E+00
45	.84266E-02	.91409E-01	.00000E+00	.10000E+01	.00000E+00
46	.65360E-02	.80581E-01	.00000E+00	.10000E+01	.00000E+00
47	.69581E-02	.83124E-01	.00000E+00	.10000E+01	.00000E+00
48	.84681E-02	.91632E-01	.00000E+00	.10000E+01	.00000E+00
49	.77228E-02	.87542E-01	.00000E+00	.20000E+01	.00000E+00
50	.64493E-02	.80070E-01	.00000E+00	.20000E+01	.00000E+00
51	.73690E-02	.85528E-01	.00000E+00	.20000E+01	.00000E+00
52	.80670E-02	.89465E-01	.00000E+00	.20000E+01	.00000E+00
53	.11972E-01	.10972E+00	.00000E+00	.20000E+01	.00000E+00
54	.11345E-01	.10594E+00	.00000E+00	.20000E+01	.00000E+00
55	.10956E-01	.10448E+00	.00000E+00	.20000E+01	.00000E+00
56	.12145E-01	.10953E+00	.00000E+00	.10000E+01	.00000E+00
57	.10619E-01	.10250E+00	.00000E+00	.20000E+01	.00000E+00
58	.12627E-01	.11176E+00	.00000E+00	.20000E+01	.00000E+00
59	.18603E-01	.13512E+00	.00000E+00	.10000E+01	.00000E+00
60	.69865E-01	.27235E+00	.00000E+00	.20000E+01	.00000E+00
61	.16462E+00	.41911E+00	.00000E+00	.50000E+01	.00000E+00
62	.48724E+00	.82463E+00	.00000E+00	.80000E+01	.00000E+00
63	.26159E+00	.49625E+00	.00000E+00	.40000E+01	.10000E+01
64	.63538E-01	.26450E+00	.00000E+00	.20000E+01	.00000E+00
65	.87828E-01	.31849E+00	.00000E+00	.20000E+01	.00000E+00
66	.41955E+00	.65143E+00	.00000E+00	.50000E+01	.00000E+00
67	.13541E-03	.11835E-01	.00000E+00	.20000E+01	.00000E+00
68	0	0	0	0	0
69	.48772E-03	.27653E-01	.00000E+00	.30000E+01	.00000E+00
70	.49959E-03	.27793E-01	.00000E+00	.30000E+01	.00000E+00
71	.49156E-03	.27412E-01	.00000E+00	.30000E+01	.00000E+00
72	.48203E-03	.26999E-01	.00000E+00	.30000E+01	.00000E+00
73	.47249E-03	.26578E-01	.00000E+00	.30000E+01	.00000E+00
74	0	0	0	0	0
75	0	0	0	0	0

76	0	0	0	0	0
77	0	0	0	0	0
78	0	0	0	0	0
79	0	0	0	0	0
80	0	0	0	0	0

AVERAGE QUEUE LENGTH = .0229

STATISTICS ON NUMBER OF FRAMES WAITING IN LAYERS 2 TO 7 QUEUES

STATION	MEAN	STD DEV	MINIMUM	MAXIMUM	CURRENT
1	.62069E-02	.78539E-01	.00000E+00	.10000E+01	.00000E+00
2	.12974E-01	.12620E+00	.00000E+00	.30000E+01	.00000E+00
3	.51470E-02	.73308E-01	.00000E+00	.30000E+01	.00000E+00
4	.47509E-02	.68763E-01	.00000E+00	.10000E+01	.00000E+00
5	.66318E-02	.81166E-01	.00000E+00	.10000E+01	.00000E+00
6	.72209E-02	.84966E-01	.00000E+00	.20000E+01	.00000E+00
7	.51677E-02	.71701E-01	.00000E+00	.10000E+01	.00000E+00
8	.12842E-03	.11332E-01	.00000E+00	.10000E+01	.00000E+00
9	.16519E-03	.12852E-01	.00000E+00	.10000E+01	.00000E+00
10	.18650E-03	.13655E-01	.00000E+00	.10000E+01	.00000E+00
11	.91159E-02	.10386E+00	.00000E+00	.40000E+01	.00000E+00
12	.39685E-03	.19917E-01	.00000E+00	.10000E+01	.00000E+00
13	0	0	0	0	0
14	.23352E-03	.15280E-01	.00000E+00	.10000E+01	.00000E+00
15	.28617E-03	.16914E-01	.00000E+00	.10000E+01	.00000E+00
16	.28633E-03	.16919E-01	.00000E+00	.10000E+01	.00000E+00
17	.20574E-03	.14342E-01	.00000E+00	.10000E+01	.00000E+00
18	.23624E-03	.16956E-01	.00000E+00	.20000E+01	.00000E+00
19	.68385E-04	.82692E-02	.00000E+00	.10000E+01	.00000E+00
20	.10356E-03	.10176E-01	.00000E+00	.10000E+01	.00000E+00
21	.15559E-03	.12472E-01	.00000E+00	.10000E+01	.00000E+00
22	.22224E-03	.14906E-01	.00000E+00	.10000E+01	.00000E+00
23	.15081E-03	.12280E-01	.00000E+00	.10000E+01	.00000E+00
24	.29172E-03	.17077E-01	.00000E+00	.10000E+01	.00000E+00
25	.39331E-03	.19828E-01	.00000E+00	.10000E+01	.00000E+00
26	.22478E-03	.14991E-01	.00000E+00	.10000E+01	.00000E+00
27	.25274E-03	.15896E-01	.00000E+00	.10000E+01	.00000E+00
28	.34479E-03	.18565E-01	.00000E+00	.10000E+01	.00000E+00
29	.35390E-03	.20129E-01	.00000E+00	.20000E+01	.00000E+00
30	.25731E-03	.16039E-01	.00000E+00	.10000E+01	.00000E+00
31	.18496E-03	.13599E-01	.00000E+00	.10000E+01	.00000E+00
32	.38651E-03	.19656E-01	.00000E+00	.10000E+01	.00000E+00
33	.15129E-03	.12299E-01	.00000E+00	.10000E+01	.00000E+00
34	.27741E-03	.16653E-01	.00000E+00	.10000E+01	.00000E+00

35	.51478E-04	.71746E-02	.00000E+00	.10000E+01	.00000E+00
36	.37807E-03	.19440E-01	.00000E+00	.10000E+01	.00000E+00
37	.17983E-03	.13409E-01	.00000E+00	.10000E+01	.00000E+00
38	.10419E-03	.10207E-01	.00000E+00	.10000E+01	.00000E+00
39	.65086E-04	.80673E-02	.00000E+00	.10000E+01	.00000E+00
40	.54633E-04	.73912E-02	.00000E+00	.10000E+01	.00000E+00
41	.86693E-04	.93105E-02	.00000E+00	.10000E+01	.00000E+00
42	.51705E-04	.71904E-02	.00000E+00	.10000E+01	.00000E+00
43	0	0	0	0	0
44	.25694E-04	.50688E-02	.00000E+00	.10000E+01	.00000E+00
45	.21641E-03	.14709E-01	.00000E+00	.10000E+01	.00000E+00
46	.20321E-03	.14254E-01	.00000E+00	.10000E+01	.00000E+00
47	0	0	0	0	0
48	0	0	0	0	0
49	.36049E-03	.20285E-01	.00000E+00	.20000E+01	.00000E+00
50	.41284E-03	.20314E-01	.00000E+00	.10000E+01	.00000E+00
51	.24391E-03	.15616E-01	.00000E+00	.10000E+01	.00000E+00
52	.35983E-03	.20266E-01	.00000E+00	.20000E+01	.00000E+00
53	.31234E-03	.17670E-01	.00000E+00	.10000E+01	.00000E+00
54	.24276E-03	.17138E-01	.00000E+00	.20000E+01	.00000E+00
55	.39214E-03	.21057E-01	.00000E+00	.20000E+01	.00000E+00
56	.19065E-03	.13806E-01	.00000E+00	.10000E+01	.00000E+00
57	.29708E-03	.18667E-01	.00000E+00	.20000E+01	.00000E+00
58	.26658E-03	.16325E-01	.00000E+00	.10000E+01	.00000E+00
59	.25603E-04	.50598E-02	.00000E+00	.10000E+01	.00000E+00
60	0	0	0	0	0
61	.98610E-03	.31387E-01	.00000E+00	.10000E+01	.00000E+00
62	.17789E+00	.55499E+00	.00000E+00	.60000E+01	.00000E+00
63	.12642E-01	.12154E+00	.00000E+00	.30000E+01	.00000E+00
64	0	0	0	0	0
65	0	0	0	0	0
66	.58129E-01	.28121E+00	.00000E+00	.40000E+01	.00000E+00
67	.12350E-01	.31549E+00	.00000E+00	.14000E+02	.00000E+00
68	0	0	0	0	0
69	0	0	0	0	0
70	0	0	0	0	0
71	0	0	0	0	0
72	0	0	0	0	0
73	0	0	0	0	0
74	0	0	0	0	0
75	.25557E-04	.50554E-02	.00000E+00	.10000E+01	.00000E+00
76	.25580E-04	.50576E-02	.00000E+00	.10000E+01	.00000E+00
77	0	0	0	0	0
78	0	0	0	0	0
79	0	0	0	0	0
80	0	0	0	0	0

AVERAGE QUEUE LENGTH = .0041

STATISTICS ON NUMBER OF FRAMES WAITING IN LAYERS 7 TO 2 QUEUES

STATION	MEAN	STD DEV	MINIMUM	MAXIMUM	CURRENT
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	0
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	0	0
20	0	0	0	0	0
21	0	0	0	0	0
22	0	0	0	0	0
23	0	0	0	0	0
24	0	0	0	0	0
25	0	0	0	0	0
26	0	0	0	0	0
27	0	0	0	0	0
28	0	0	0	0	0
29	0	0	0	0	0
30	0	0	0	0	0
31	0	0	0	0	0
32	0	0	0	0	0
33	0	0	0	0	0
34	0	0	0	0	0
35	0	0	0	0	0
36	0	0	0	0	0
37	0	0	0	0	0
38	0	0	0	0	0
39	0	0	0	0	0
40	0	0	0	0	0
41	0	0	0	0	0

42	0	0	0	0	0
43	0	0	0	0	0
44	0	0	0	0	0
45	0	0	0	0	0
46	0	0	0	0	0
47	0	0	0	0	0
48	0	0	0	0	0
49	0	0	0	0	0
50	0	0	0	0	0
51	0	0	0	0	0
52	0	0	0	0	0
53	0	0	0	0	0
54	0	0	0	0	0
55	0	0	0	0	0
56	0	0	0	0	0
57	0	0	0	0	0
58	0	0	0	0	0
59	0	0	0	0	0
60	0	0	0	0	0
61	0	0	0	0	0
62	0	0	0	0	0
63	0	0	0	0	0
64	0	0	0	0	0
65	0	0	0	0	0
66	0	0	0	0	0
67	0	0	0	0	0
68	0	0	0	0	0
69	0	0	0	0	0
70	0	0	0	0	0
71	0	0	0	0	0
72	0	0	0	0	0
73	0	0	0	0	0
74	0	0	0	0	0
75	0	0	0	0	0
76	0	0	0	0	0
77	0	0	0	0	0
78	0	0	0	0	0
79	0	0	0	0	0
80	0	0	0	0	0

AVERAGE QUEUE LENGTH = .0000

STATISTICS FOR TIME IN THE SYSTEM (MIC SEC) OF MESSAGES SENT FROM EACH STATION

STATION	MEAN	STD DEV	SD MEAN	CV	MINIMUM	MAXIMUM	OBS
1	.735E+05	.775E+03	.256E+02	.105E-01	.000E+00	.769E+05	913.

2	.741E+05	.398E+04	.147E+03	.537E-01	.000E+00	.111E+06	728.
3	.754E+05	.838E+04	.321E+03	.111E+00	.000E+00	.149E+06	684.
4	.736E+05	.127E+04	.480E+02	.172E-01	.000E+00	.915E+05	696.
5	.738E+05	.244E+04	.794E+02	.331E-01	.000E+00	.105E+06	948.
6	.746E+05	.468E+04	.149E+03	.627E-01	.000E+00	.111E+06	984.
7	.100E+06	.180E+05	.679E+03	.179E+00	.000E+00	.145E+06	701.
8	.901E+05	.275E+05	.899E+03	.305E+00	.000E+00	.291E+06	936.
9	.902E+05	.290E+05	.937E+03	.321E+00	.000E+00	.260E+06	955.
10	.885E+05	.265E+05	.865E+03	.300E+00	.000E+00	.255E+06	942.
11	.891E+05	.289E+05	.901E+03	.324E+00	.000E+00	.292E+06	1028.
12	.901E+05	.278E+05	.907E+03	.309E+00	.000E+00	.257E+06	941.
13	.758E+05	.646E+04	.211E+03	.852E-01	.000E+00	.122E+06	940.
14	.819E+05	.178E+05	.584E+03	.218E+00	.000E+00	.183E+06	932.
15	.808E+05	.171E+05	.562E+03	.212E+00	.000E+00	.184E+06	932.
16	.800E+05	.160E+05	.516E+03	.201E+00	.000E+00	.218E+06	969.
17	.809E+05	.186E+05	.615E+03	.230E+00	.000E+00	.219E+06	911.
18	.801E+05	.163E+05	.542E+03	.204E+00	.000E+00	.184E+06	905.
19	.782E+05	.113E+05	.373E+03	.145E+00	.000E+00	.149E+06	927.
20	.785E+05	.123E+05	.402E+03	.156E+00	.000E+00	.147E+06	930.
21	.780E+05	.119E+05	.377E+03	.153E+00	.000E+00	.154E+06	999.
22	.773E+05	.974E+04	.314E+03	.126E+00	.000E+00	.146E+06	963.
23	.775E+05	.102E+05	.326E+03	.131E+00	.000E+00	.146E+06	975.
24	.926E+05	.275E+05	.899E+03	.297E+00	.000E+00	.262E+06	937.
25	.913E+05	.263E+05	.882E+03	.289E+00	.000E+00	.260E+06	893.
26	.913E+05	.283E+05	.945E+03	.310E+00	.000E+00	.293E+06	900.
27	.917E+05	.280E+05	.896E+03	.305E+00	.000E+00	.258E+06	976.
28	.935E+05	.284E+05	.946E+03	.304E+00	.000E+00	.256E+06	900.
29	.922E+05	.265E+05	.844E+03	.287E+00	.000E+00	.256E+06	985.
30	.950E+05	.300E+05	.965E+03	.316E+00	.000E+00	.292E+06	968.
31	.915E+05	.282E+05	.925E+03	.308E+00	.000E+00	.233E+06	928.
32	.923E+05	.266E+05	.869E+03	.288E+00	.000E+00	.257E+06	940.
33	.907E+05	.252E+05	.828E+03	.278E+00	.000E+00	.220E+06	928.
34	.913E+05	.271E+05	.861E+03	.296E+00	.000E+00	.255E+06	988.
35	.821E+05	.162E+05	.528E+03	.197E+00	.000E+00	.233E+06	939.
36	.928E+05	.300E+05	.978E+03	.323E+00	.000E+00	.288E+06	940.
37	.811E+05	.152E+05	.498E+03	.188E+00	.000E+00	.208E+06	933.
38	.864E+05	.219E+05	.721E+03	.254E+00	.000E+00	.236E+06	927.
39	.921E+05	.271E+05	.873E+03	.294E+00	.000E+00	.256E+06	964.
40	.841E+05	.186E+05	.610E+03	.222E+00	.000E+00	.223E+06	934.
41	.832E+05	.178E+05	.588E+03	.215E+00	.000E+00	.235E+06	922.
42	.814E+05	.162E+05	.537E+03	.198E+00	.000E+00	.242E+06	905.
43	.865E+05	.211E+05	.680E+03	.244E+00	.000E+00	.264E+06	966.
44	.876E+05	.233E+05	.772E+03	.266E+00	.000E+00	.262E+06	907.
45	.969E+05	.330E+05	.109E+04	.341E+00	.000E+00	.317E+06	923.
46	.837E+05	.190E+05	.609E+03	.227E+00	.000E+00	.255E+06	975.
47	.836E+05	.228E+05	.746E+03	.273E+00	.000E+00	.306E+06	931.
48	.856E+05	.260E+05	.842E+03	.304E+00	.000E+00	.308E+06	952.
49	.924E+05	.292E+05	.944E+03	.316E+00	.000E+00	.383E+06	956.

50	.907E+05	.261E+05	.853E+03	.287E+00	.000E+00	.261E+06	934.
51	.909E+05	.266E+05	.875E+03	.293E+00	.000E+00	.340E+06	927.
52	.917E+05	.275E+05	.897E+03	.300E+00	.000E+00	.308E+06	938.
53	.988E+05	.359E+05	.118E+04	.363E+00	.000E+00	.407E+06	932.
54	.982E+05	.341E+05	.111E+04	.347E+00	.000E+00	.334E+06	939.
55	.939E+05	.325E+05	.102E+04	.347E+00	.000E+00	.373E+06	1019.
56	.962E+05	.388E+05	.127E+04	.404E+00	.000E+00	.398E+06	933.
57	.938E+05	.316E+05	.103E+04	.337E+00	.000E+00	.386E+06	938.
58	.955E+05	.346E+05	.112E+04	.363E+00	.000E+00	.338E+06	952.
59	.101E+06	.460E+05	.151E+04	.454E+00	.000E+00	.498E+06	930.
60	.114E+06	.501E+05	.943E+03	.440E+00	.000E+00	.542E+06	2820.
61	.122E+06	.593E+05	.805E+03	.485E+00	.000E+00	.561E+06	5423.
62	.109E+06	.524E+05	.372E+03	.481E+00	.000E+00	.556E+06	19858.
63	.114E+06	.548E+05	.571E+03	.479E+00	.000E+00	.616E+06	9215.
64	.128E+06	.561E+05	.129E+04	.436E+00	.000E+00	.593E+06	1879.
65	.132E+06	.596E+05	.123E+04	.451E+00	.000E+00	.505E+06	2349.
66	.118E+06	.585E+05	.510E+03	.494E+00	.000E+00	.563E+06	13176.
67	.202E+06	.202E+05	.143E+05	.100E+00	.000E+00	.216E+06	2.
68	NO VALUES RECORDED						
69	.627E+06	.451E+05	.202E+05	.719E-01	.000E+00	.688E+06	5.
70	.562E+06	.181E+05	.811E+04	.323E-01	.000E+00	.584E+06	5.
71	.496E+06	.427E+05	.191E+05	.860E-01	.000E+00	.548E+06	5.
72	.430E+06	.807E+05	.361E+05	.187E+00	.000E+00	.511E+06	5.
73	.365E+06	.120E+06	.537E+05	.329E+00	.000E+00	.475E+06	5.
74	NO VALUES RECORDED						
75	NO VALUES RECORDED						
76	NO VALUES RECORDED						
77	NO VALUES RECORDED						
78	NO VALUES RECORDED						
79	NO VALUES RECORDED						
80	NO VALUES RECORDED						

AVERAGE TIME IN THE SYSTEM = .10127822E+06

SIMAN Summary Report

Run Number 1 of 1

Project: MAP SIMULATION
Analyst: S.HARINSUTA
Date : 11/12/1989

Run Ended at Time : 1.410065E+09 Microseconds

Tally Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Number of Obs.
1	TRT NETWORK 1	.00000	.00000	.00000	.00000	0
2	TRT NETWORK 2	.00000	.00000	.00000	.00000	0
3	TRT NETWORK 3	.00000	.00000	.00000	.00000	0
4	TRT NETWORK 4	.00000	.00000	.00000	.00000	0
5	TRT NETWORK 5	.00000	.00000	.00000	.00000	0
6	TRT NETWORK 6	.00000	.00000	.00000	.00000	0
7	TRT NETWORK 7	.00000	.00000	.00000	.00000	0
8	TRT NETWORK 8	.00000	.00000	.00000	.00000	0
9	TRT NETWORK 9	.00000	.00000	.00000	.00000	0
10	TRT NETWORK 10	.00000	.00000	.00000	.00000	0
11	TRT NETWORK 11	.00000	.00000	.00000	.00000	0
12	TRT NETWORK 12	.00000	.00000	.00000	.00000	0
13	TRT NETWORK 13	.00000	.00000	.00000	.00000	0
14	TRT NETWORK 14	.00000	.00000	.00000	.00000	0
15	TRT NETWORK 15	.00000	.00000	.00000	.00000	0
16	TRT NETWORK 16	.00000	.00000	.00000	.00000	0
17	TRT NETWORK 17	.00000	.00000	.00000	.00000	0
18	TRT NETWORK 18	.00000	.00000	.00000	.00000	0
19	TRT NETWORK 19	.00000	.00000	.00000	.00000	0
20	TRT NETWORK 20	.00000	.00000	.00000	.00000	0
21	TRT NETWORK 21	.00000	.00000	.00000	.00000	0
22	TRT NETWORK 22	.00000	.00000	.00000	.00000	0
23	TRT NETWORK 23	.00000	.00000	.00000	.00000	0
24	TRT NETWORK 24	.00000	.00000	.00000	.00000	0

Counters

Number	Identifier	Count	Limit
1	END OF SIMULATION	0	1

Run Time : 1 Hour(s) and 0 Minute(s)

APPENDIX F

GLOSSARY

Access
The ability of devices to interconnect for the purpose of exchanging information.

ANOVA
Analysis of variance (ANOVA), a statistical procedure for testing the equality of several population means.

Application
A software program that performs some useful task. Examples of applications include database managers, communications packages and graphics programs.

Application Layer
The seventh layer of the Open Systems Interconnection model of the International Standards Organization. This layer supplies functions to applications or nodes enabling them to communicate with other applications or nodes.

Association
A logical connection between two host applications using services from their application layers.

AutoCAD
Computer Aided Design software with built-in drawing functions, programming language and drawing data base.

AutoLISP
A LISP based language within AutoCAD that can access AutoCAD drawing data base and functions.

Bandwidth
The range of frequencies that a communication channel can support.

Baseband
A form of modulation in which data signals can be inserted directly onto the transmission medium without frequency division.

Binary
A numbering system consisting of only two values, zero and one.

Bit
A binary unit of information. Contraction of the words BInary digiT.

Bridge
A store and forward device used in connecting networks with a common set of higher level protocols.

Broadband
A modulation technique that forms multiple channels by dividing the transmission medium into discrete frequency segments.

Bus Topology
A network topology that connects devices on a shared transmission medium. Transmission propagates the length of the medium and is received by every device on the network.

Byte
A set of eight bits of information. Also called an octet.

CAD
Abbreviation for Computer-Aided Design. A combination of software and hardware tools for technical drawing and drafting.

Carrierband
A single channel broadband.

Coaxial Cable
A transmission medium consisting of a center conductor and an outer, concentric conductor.

Collision
A condition in which multiple devices on a contention network attempt to access the network simultaneously.

Contention
One of the multiple access methods for a network. It allocates capacity on a first-come-first-served basis.

CSMA
Carrier Sense Multiple Access (CSMA), a contention network in which a station listens for the presence of a carrier before attempting to transmit.

CSMA/CD
Carrier Sense Multiple Access with Collision Detection (CSMA/CD), a refinement of CSMA in which the station detects the the presence of a collision by monitoring for a distorted pulse.

Data Packet
In this research, the same as a frame, a block data containing strings of information that may contain the overhead from the higher layers as well as the actual information that is to be transmitted.

Data Link Layer
Layer 2 of the Open Systems Interconnection model of the International Standards Organization. It is the layer that puts messages together and coordinates their flow.

Demodulation
The process of decoding digital signal from an analog carrier signal.

DOS
Abbreviation for Disk Operating System, an operating system for IBM and IBM-compatible personal computers.

Ethernet
A 10 Mb/s CSMA/CD baseband local area network developed by Xerox, Intel and Digital Equipment.

Fiber Optics
A transmission medium that consists of thin glass fiber capable of transporting high-speed data by rapid on/off pulses of light.

Filter
A frequency-sensitive device that accepts or rejects specific bans of frequencies.

Frame
In this research, is the same as a data packet, a block of data containing strings of information that may contain the overhead information of the higher layers as well as the actual information that is to be transmitted.

Gateway
A device that interconnects networks through protocol conversion.

Headend Remodulator
Equipment used for splitting the transmitting and receiving frequencies or to connect the transmitting cable to the receiving cable on broadband networks.

ISO
An abbreviation for International Standards Organization.

LAN
Local Area Network (LAN), a general-purpose communication network that provides connection of a variety of data communicating devices within a single organization.

Layer
In the OSI model, a collection of network processing functions.

MAP
An abbreviation for Manufacturing Automation Protocol, a token-passing bus LAN designed for factory environments.

Medium Access Control (MAC)
The method of determining which device on the network has access to the transmission medium at any time. Examples of MAC include CSMA/CD, token bus and token ring.

Modem
A device that converts analog signals to digital signals and vice versa.

Modulation
Any systematic change to a carrier signal to encode information. For example, digital signals are converted to analog signals for transmission over telephone lines.

Multiplexer
A device used for combining several signals over a single line when transmitting and separating them when receiving.

Node
A device on a network that can communicate with other devices on the same network.

Network Number
Same as Sub-network number.

Octet
Eight bits of data. Also known as a byte.

Overhead
Any non-information that is added to the information to be transmitted, such as headers, error checking bits, etc.

OS/2
The second-generation operating system for Intel 80286 and 80386 microprocessors. Key features include support for multitasking and programs larger than 640K bytes and graphical user interface.

OSI Model
An abbreviation for Open Systems Interconnection model developed by International Standards Organization. It provides a network design framework to allow equipment from different vendors to communicate.

Propagation Delay
Time delay associated with transmitting the signals across the transmission medium.

Protocol
Sets of rules and conventions used for establishing and maintaining communications between devices on the network.

Ring Topology
A network topology that connect stations to repeaters in a closed loop. Transmissions are made in one direction around the ring to all stations on the network.

Router
A device used in connecting two or more sub-networks by using an internet protocol which is a connectionless protocol operating at layer 3 of the OSI model.

SIMAN
A general purpose simulation language with process orientation, discrete event orientation and continuous orientation marketed by Systems Modeling Corp.

Station
A device on a network that can communicate with another device on the same network.

Station Number
The unique consecutive numeric assignment of each station on a network.

Sub-network Number
The unique consecutive numeric assignment of each sub-network on a network. Also referred to as the "network number."

Tap
The physical point of connection between a device on the network and the transmission medium.

Terminal
A device consisting of hardware and software enabling a direct user interface to a network.

Token
A control packet that is passed from node to node on the network.

Token Bus
A token passing scheme for a bus or tree topology.

Token Passing
A method of allocating network access. The node that possess the token has the right to transmit messages.

Token Ring
A token passing scheme for a ring topology.

Topology
The ways in which devices on the network are connected or linked together.

Tree Topology
A network topology that connects stations to a shared transmission medium. The transmission medium consists of branching cable with no closed circuits. The transmissions propagate throughout all branches and are received by all stations on the network.

Transmission Medium
The physical path on which transmission signals travel between transmitters and receivers in a communication network.

Trigger
The number of data elements to be read from a PLC.

Twisted Pair
A transmission medium consisting of two insulated wires arranged in a spiral pattern.

VITA

Saran Harinsuta was born in Bangkok, Thailand, on January 30, 1958. He studied at Assumption College in Bangkok until the age of 12 before transferring to Oundle School, Peterborough, Northhamptonshire, England. After spending 5 years in England, he enrolled at The University of Tennessee, Knoxville. He received a Bachelor of Science degree in Mechanical Engineering in 1979. He continued his study at the same university and received a Master of Science degree in Industrial Engineering in 1982. He continued to pursue a Ph.D degree in Engineering Science at UT, Knoxville.

While studying at UT, he held several positions at the university at different times. He taught a junior level course in Operations Research, worked part time as a programmer for the Construction Resources Analysis department, and was employed as a users' assistant and an operator at the Engineering Computer Center. For his research, he was sponsored by the Martin Marietta and General Motors Corporations. He received his Ph.D degree in May, 1990, from The University of Tennessee, Knoxville.

He is a member of the Phi Kappa Phi and the Golden Key honor societies.