F	AD-A089 493 UNCLASSIFIED		NAVAL PETRI MAR 8 NPS52	NAVAL POSTGRADUATE SCHOOL MONTEREY CA Petri-Net Simulations of Communications Networks.(U) Mar 80 S C Jennings, R J Hartel N9552-80-003						F/6 17/2 NL				
		1 ··· 2 4009403	æ.,											
Ī														
							1. Y							
À														



NAVAL POSTGRADUATE SCHOOL Monterey, California

Rear Admiral J. J. Ekelund Superintendent

ц,

i

Jack R. Borsting Provost

i

Reproduction of all or part of this report is authorized.

Released as a Technical Report by:

Toller W. M. TOLLES,

Dean of Research

# **DISCLAIMER NOTICE**

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

	REPORT DOCUME	NTATION PAGE	READ INSTRUCTIONS
ALPORT NU	MOCH	2. GOVT A	CCESSION NO. 3. RECIPIENT'S CATALOG NUMBER
NPS52-8	30-003	40-1	A089 4 93
4. TITLE (and	Subtitle)		S. TYPE OF REPORT & PEMOD COVERE
Dotri -1	loh Cimulations of	Communications	Master's Thesis;
Network	vet Simulations of	Communications	March 1980
NECWOLT	15		• PERFORMING ORG. REPORT NUMBER
7. AUTHORIA			8. CONTRACT OR GRANT NUMBER(.)
Stepher	n Craig Jennings		
Robert	John Hartel		
. PERFORMIN	ORGANIZATION NAME A		10. PROGRAM ELEMENT, PROJECT, TAS
Naval	Bostaraduata Scho	01	AREA & WORK UNIT HUMBERS
Montere	ev. CA 93940	OT.	
		0.00711	
	INTE UTTIGE MADE AND AD		March 1980
Naval I	Postgraduate Schoo	1	13. HUMBER OF PAGES
Montere	BY, CA 93940		178
16. MONITORIA	IG AGENCY NAME & ACORI	ESS/II different from Contr	oling Office) 18. SECURITY CLASS. (of this report)
			UNCLASSIFIED
			ISA. DECLASSIFICATION DOWNGRADING
Approve	ed for public rele	ase; distributio	on unlimited
Approve	ed for public rele	ase; distributio	on unlimited
Approve 7. DISTRIBUT	ed for public rele	ase; distributio	on unlimited
Approve 7. DISTRIBUT 8. SUPPLEME	ed for public rele	ase; distributio	on unlimited
Approve 7. DISTRIBUT 8. SUPPLEME 9. KEY WORDS	ed for public rele NON STATEMENT («/ /» «» NTARY NOTES	ase; distributio	on unlimited
Approve	ed for public rele NON STATEMENT (of the sad	ase; distributio	p Neck number)
Approve 7. DISTRIBUT 9. SUPPLEME 9. KEY WORDS Simula Petri-1	ed for public rele NON STATEMENT (of the and NTARY NOTES (Continue on reverse of the 11 tion Net	ase; distributio	on unlimited , If different from Report) y Mech number)
Approve 7. DISTRIBUT 8. SUPPLEME 9. KEY WORDS Simula Petri~ Evalua	ed for public rele NON STATEMENT (a/ the and NTARY NOTES (Continue on reverse aide if tion Net tion	ase; distributio	p Moch number)
Approve 7. DISTRIBUT 8. SUPPLEME 9. KEY WORDS Simula Petri-J Evalua	ed for public rele NON STATEMENT (of the second NTARY NOTES (Continue on reverse of the fit tion Net tion	ase; distributio	on unlimited
Approve 7. DISTRIBUT 8. SUPPLEME 9. «EY WORDS Simula Petri- Evalua 9. ABSTRACT	ed for public rele NON STATEMENT (of the and NTARY NOTES (Continue on reverse of do If tion Net tion	noccoccary and identify by	phoet number)
Approve 7. DISTRIBUT 8. SUPPLEME 9. KEY WORDS Simula Petri~ Evalua 9. ABSTRACT Curr	ed for public rele NON STATEMENT (of the end NTARY NOTES (Continue on reverse olde !! tion Net tion (Continue on reverse olde !! ent technologies i sing are becoming	neccessary and identify by in the fields of	<pre>pheck number; telecommunications and computer tegrated to the extent that</pre>
Approve 2. DISTRIBUT 0. SUPPLEME 0. SUPPLEME Simula Petri- Evalua 0. ADSTRACT Curr proces "distr	ed for public rele NON STATEMENT («/ // » «» NTARY NOTES (Continue on revorce of // tion Net tion (Continue on revorce of // ent technologies i sing are becoming ibuted" computer r	necessary and identify by in the fields of increasingly in hetworks are ass	<pre>&gt; Mesh number) &gt; Mesh number) &gt; Mesh number) &gt; telecommunications and computer tegrated to the extent that uming key roles in communications.</pre>
Approve 2. DISTRIBUT 2. SUPPLEME 3. SUPPLEME 5. Simula Petri- Evalua 5. ABSTRACT Curr proces. "distr The co	ed for public rele NON STATEMENT (of the second NTARY NOTES (Continue on reverse ofde if tion Net tion (Continue on reverse ofde if ent technologies if sing are becoming ibuted" computer r mplex computerized	necessary and identify by increasingly in networks are ass systems necess	<pre>&gt; week number; &gt; Neek number; &gt; Neek number; &gt; telecommunications and computer tegrated to the extent that uming key roles in communications. ary to support modern military</pre>
Approve 7. DISTRIBUTI 9. SUPPLEME 9. SUPPLEME 9. SUPPLEME 1. SUPPLEME 9. ADSTRACT CUIT: proces "distr The col comman	Continue on reverse side '' (Continue on reverse side '' tion Net tion (Continue on reverse side '' ent technologies i sing are becoming ibuted" computer r mplex computerized d and control requ	necessary and identify by in the fields of increasingly in hetworks are ass i systems necess irements are ex	<pre>&gt; Meet number) &gt; Meet number) &gt; Meet number) &gt; telecommunications and computer tegrated to the extent that uming key roles in communications. ary to support modern military pensive. Designing such systems</pre>
Approve 7. DISTRIBUT 8. SUPPLEME 9. SUPPLEME Simula Petri- Evalua 9. ABSTRACT Curr proces. "distr The co comman by tri	A for public rele NON STATEMENT (of the end NTARY NOTES (Continue on reverse ofde if tion Net tion (Continue on reverse ofde if ent technologies if sing are becoming ibuted" computer r mplex computerized d and control requ al and error is no	neccessary and identify by in the fields of increasingly in hetworks are ass in feasible, yet	<pre>&gt; Meet number) &gt; Meet number) &gt; Meet number) &gt; telecommunications and computer tegrated to the extent that uming key roles in communications. ary to support modern military pensive. Designing such systems no other viable alternatives exist.</pre>
Approve 2. Distribut 3. Supplement 5. Supplement	A for public rele NON STATEMENT (of the end NTARY NOTES (Continue on revorce of the fit tion Net tion (Continue on revorce of the fit ent technologies is sing are becoming ibuted" computer r mplex computerized and control requ al and error is no hesis offers an or	necessary and identify by in the fields of increasingly in hetworks are ass it systems necess tirements are ex of feasible, yet riginal methodol	<pre>&gt; Meet member) &gt; Meet member) &gt; Meet member) &gt; telecommunications and computer tegrated to the extent that uming key roles in communications. ary to support modern military pensive. Designing such systems ino other viable alternatives exist. ogy for evaluating the predicted</pre>
Approve 2. DISTRIBUTION 2. SUPPLENCE 3. SUPPLENCE 5. Simula Petri- Evalua 5. ASSTRACT Curr proces. "distr The con comman by tri This t	ACCOMPTENT Contract of the second statement (of the second statement (of the second statement of the s	necessary and identify by in the fields of increasingly in networks are ass is ystems necess birements are ex of feasible, yet riginal methodol	<pre>&gt; Meet member) &gt; Meet member) &gt; Meet member) &gt; telecommunications and computer tegrated to the extent that uming key roles in communications. ary to support modern military pensive. Designing such systems no other viable alternatives exist. ogy for evaluating the predicted</pre>
Approve 7. DISTRIBUT 9. SUPPLENE 9. SUPPLENE 9. SUPPLENE 9. ABSTRACT CUTT Proces. "distr The col comman by tri This ti 9. ABSTRACT 0. Jan 75 h	Continue on reverse side if (Continue on reverse side if tion (Continue on reverse side if tion (Continue on reverse side if tion (Continue on reverse side if ent technologies if sing are becoming ibuted" computer r mplex computerized d and control requal and error is no hesis offers an or \$(N 0102-01000)	necessary and identify by in the fields of increasingly in networks are ass is systems necess irements are ex of feasible, yet riginal methodol	<pre>&gt; Meet number) &gt; Meet number) &gt;</pre>
Approve 7. DISTRIBUT 8. SUPPLEME 9. SUPPLEME 9. ADSTRACT CUIT Process "distr The con comman by tri This t 0. (JAN 7) 1 Page 1)	ACCONTINUE OF FORMULE SALE (Continue on reverse side // tion Net tion (Continue on reverse side // tion (Continue on reverse side // tion (Continue on reverse side // ent technologies i sing are becoming ibuted" computer r mplex computer rized d and control requ al and error is no hesis offers an or S/N 0102-010-0600	neccessary and identify by in the fields of increasingly in networks are ass is systems necess in feasible, yet riginal methodol	<pre>&gt;</pre>

ų

UNCLASSIFIED

BEUMTY CLASSIFICATION OF THIS PAGE/What Rote Entered.

performance of military automated systems. Using Petri-Nets as a modeling tool, computer simulations with color graphics output are performed to demonstrate the feasibility of this approach as a systems design tool.

Accession For NTIS GRA&I DDC TAB Unannounced Justification Εv pist ۰£. οř. . al in the Dist

DD Form 1473 5/N 0102-014-6601

UNCLASSIFIED

N. 1.37

2 SECURITY CLASSIFICATION OF THIS PACE/Then Date Entered)



#### ABSTRACT

Current technologies in the fields of telecommunications and computer processing are becoming increasingly integrated to the extent that "distributed" computer networks are assuming key roles in communications. The complex computerized systems necessary to support modern military command and control requirements are expensive. Designing such systems by trial and error is not feasible, yet no other viable alternatives exist. This thesis offers an original methodology for evaluating the predicted performance of military automated systems. Using Petri-Nets as a modeling tool, computer simulations with color graphics output are performed to demonstrate the feasibilty of this approach as a systems design tool.

# TABLE OF CONTENTS

I.	INT	ROTUCTION	
II.	N ፻፹	WOPKS: MOTIVATION, TAXONOMY AND PEPFORMANCE 14	
	A.	INTRODUCTION 14	
	в.	MOTIVATION FOR THE ANALYSIS OF PEPFORMANCE 14	
	с.	TAXONOMY 16	
		1. Networks 16	
		2. Computer Communication Networks 16	
		3. Further Reading 18	
	D.	PERFORMANCE PREDICTION AND MODELING 18	
	F.	SUMMARY	
III.	MIL	ITAPY APPLICATIONS OF DISTRIBUTED SYSTEM	
	TEC	ENCLOGY	
	A.	INTRODUCTION 22	
	P.	PACKET SWITCHING 22	
	с.	THE ARPANET 25	
	D.	MILITARY APPLICATIONS 27	
	E.	THE ALOFA SISTEM 28	
	F.	PACKET RADIO INTRODUCTION 29	
	G.	CHANNEL ACCESS SCHEMES 31	
		1. Category I 32	
		2. Category II	
		3. Category III 37	
	Ħ.	CURRENT PROGRAMS	
		1. PLRS/JTIDS Hybrid 39	

١.

		·	
		2. The San Francisco Bay Experimental	
		Packet Padic Network	
		3. The Fort Bragg Testbed 41	
	Ι.	FUTURE IMPLICATIONS 43	
		1. Chain of Command 44	
		2. Network Management 45	
		3. How Much Redundancy? 46	
		4. Propagation Loss Due to Higher Frequency 46	
		5. Management Information Systems 47	
		6. Interoperability 47	
		7. Voice vs. Data Circuits 49	
		8. System Cost 48	
IV.	AN 1	INTRODUCTION TO PETRI-NETS 49	
	Α.	INTRODUCTION	
	Ξ.	HISTORY 49	
	с.	FOW PETRI-NETS WORK	
	D.	A SIMPLE EXAMPLE	
	£.	ADVANTAGES AND DISADVANTAGES	
۷.	PETI	RI-NET SIMULATION OF COMPUTED COMMUNICATION	
	NET	WORKS	
	Α.	INTPODUCTION	
	E.	TYPES OF NODES	
	c.	A SIMPLE APPLICATION EXAMPLE	
	D.	RANDOMNESS IN PETRI-NETS	
	E.	MEMORY STORAGE REPRESENTATION	
	F.	SYSTEM LOAD AVERAGE	
	G.	TIME REPRESENTATION 68	

ų

6

3 4 K 4 2 8 74

VI.	THE	EXPERIMENT
	A.	DESCRIPTION
	₽.	RESULTS
VII.	RECO	MMENDATIONS AND CONCLUSIONS
	A.	RECOMMENDATIONS
	Β.	CONCLUSIONS
APPE	DIX	A - USER INSTRUCTIONS FOR PETRI-NET SOFTWARE E2
	Α.	INTRODUCTION
	в.	THE INPUT FILE 82
		1. Places
		2. Transitions £5
		3. Initial Marking 86
		4. Execution
	с.	THE OUTPUT FILES
	D.	USTR OPTICNS
		1. Choice of Programs 88
		2. User Cuestions and Responses
		3. The Highlighting Feature
	Ę.	UNIQUE INSTRUCTIONS FOR VEPSICN 3
APPE	NDIX	B - RUN03
APPE	DIX	C - RUN03A
APPEN	DIX	F - RUNC3E
APPE	DIX	F - RUN P3C
APPEI	DIX	F - PUN20
APPEN	DIX	G - THESIS FILES
COMPL	JTER	PROGRAM - "simulator"
COMPU	ITTP	PPOGRAM - "transgraph"119

0)

П.

7

#### INTRODUCTION

As military planners look forward to the design of future command, control, and communications (CZ) systems, several key factors should be at the forefront of their thinking. First of all, it is readily apparent that there has been a proliferation of computer resources for military applications. Even the most "tactical" of systems today is becoming a sizeable collection of computers, databases, sensors and information-handling equipment.

Secondly. one can sense that the fields of telecommunications and computer science are becoming increasingly integrated. Although these once-separate disciplines have very different histories and traditions. they are experiencing a technological convergence which is having far-reaching implications for both the military and civilian societies. Today the concepts and techniques of computer processing have been integrated with communications to the extent that both fields share the same kind of logic, storage, switching and transmission. Because information handling systems now employ telecommunications and information in such an intimate mixture, it is difficult to distinguish what in the system is computer processing and what is communications.

A third key factor should be a realization that certain concepts of computer communication networks and distributed

system architectures offer advantages for military applications due to the potential for survivability and lack of centralization.

Computer networks are often created spontaneously by combining computers and communications. The growth of computer networks is one of the significant outcomes of the convergence of the two disciplines. An extensive array of computing resources can be connected over a wide geographic area via telecommunications channels. The potential architectures for such networks are limitless when one considers the variety of hardware, software, protocols and geographic distributions that might comprise the system.

Another key factor is that system planners reed to focus their efforts upon total system integration. Tactical automated systems tend to be developed individually to meet unique mission requirements. For instance, separate systems are typically justified and developed for missions of fire control, air defense, intelligence, personnel management and logistics, etc. While these separate systems may peform satisfactorily alone, there is difficulty in providing a suitable management information system by which the overall commander or decision-maker can have access to the information in all these separate systems in a format that is easy to understand and use.

Another important consideration is that the proliferation of automated systems places a severe burden upon the communications equipment that links systems

together. There is a tendency for automated systems to be separately developed with insufficient emphasis placed upon the communications equipment that will transmit the information. In other words, the sensors and processors of a system may work splendidly, but planners must not take it for granted that the data will arrive at the correct destination in an error-free condition.

The communications equipment and channelization that carry the information must be as carefully engineered as the other components of the system. In addition, the data on the channel must be in a format that is compatible with other systems. The U.S. Army is coming to grips with the fact that if the some 50 tactical automated systems on the drawing boards were to be fielded for use at the corps level, there exists no communications equipment capable of carrying the vast volume of information these systems would generate. In addition, when the communications equipment rust operate with specifications of transmission security. jam resistance, and low probability of intercept in a severe electronic warfare threat environment, the design difficulties are considerably magnified.

Lastly, military planners should be concerned about being able to accurately predict system performance. An accurate predictor of system performance is needed for use by those in procurement duties to ensure that performance specifications given by contractors will in fact prove to be true when a new system is fielded. Managers of currently

÷.

14

operational systems also require this service. They are asking in the course of daily duties such questions as: That would be the effect of increased buffer space at this busy location? What would be the impact on total system performance if a particular node or link in the network is removed? Such a desire for prediction of system performance has created great emphasis upon modeling techniques and computer simulations of systems to answer these questions.

All of the above factors have given impetus to this thesis. A particular modeling tool which addresses these needs is described (the Petri-Net) and is implemented to facilitate communications network modeling.

The paper presents three primary points of original work:

1. It is demonstrated that automated networks can be meaningfully modeled with the use of Petri-Nets.

2. It is shown that Petri-Net models of networks can be adapted for effective computer execution and display or a color graphics terminal. Simulations which incorporate graphics output are more easily understood and have considerable educational value.

3. The results of this research indicate that the implementation of such a modeling technique in a production environment as a predictor of system performance is feasible.

The above 3 points, although successfully implemented in the Naval Postgraduate School C3 laboratory, represent a new area of research in a preliminary stage of development. Future investigation is required to expand and validate this approach.

#### II. NETWORKS: MCTIVATION, TAXCNOMY AND PERFORMANCE

#### A. INTRODUCTION

Predicting the performance of an automated network can indicate a measure of a system's effectiveness and efficiency.[3] Evaluating a system's performance is a complex task. This task often requires modeling. Many performance modifications are more suitably performed on models than the actual system, because "trial and error" production is not feasible economically. This chapter discusses the motivations for network design and the practicalities of predicting network performance.

### B. MOTIVATION FOR THE ANALYSIS OF PERFORMANCE

The magnitude of information processing in the United States is unprecedented and still growing. Computer processing and communications make a major portion of currently accessable information available to federal agencies and commerical businesses. As the country is becoming increasingly dependent on the need for information, the existance of reliable, effective computer communication networks is essertial to transport computer-processed information.

With the added cost of energy, the attractiveness of moving resources to the user via computer communications networks is apparent. [5] The performance evaluation of these networks is, therefore, a measure of the system's ability to meet user requirements.

Adequate performance evaluation tools currently do not exist. Therefore, decision-makers responsible for selecting new systems usually rely on the designer's claims as to performance, and procure systems accordingly.

Both the civilian and military communities place very heavy demands on communication's facilities during crisis situations. Throughout history existing systems have often not been sufficient to carry such communications traffic. [8]

The civilian community shares these problems. Many networks are engineered to carry mean traffic loads and are not planned for crisis contigencies. The commerical telephone network, for instance, is inundated with traffic Christmas and Mother's Day, severely degrading system performance. The disruptive affects of a national emergency on all types of automated networks can only be imagined.

In the Defense Communications Agency (DCA), work is presently under way to develop tools for evaluating the performance of planned computer communication networks. This effort is underway because development of these expensive systems by "trial and error" can no longer be afforded. The necessity to have confidence in the system's ability to meet design specifications before production is essential, since normal federal procurement cycles stretch out over eight to ten years. This confidence can be insured by utilizing effective performance evaluation tools.

### C. TAXONOMY

A great deal of ambiguity exists in the jargon of the networking field. [7] Therefore, several recurring network descriptions are defined in this chapter to provide a consistent vorabulary. Using these definitions, network issues can then be succinctly conveyed to the reader.

#### 1. Networks

The term "network" conveys the concept of irdividualized cells and a degree of interconnectivity. A network exists for the purpose of achieving a desired objective. The individual characteristics of networks are related to network topology, hardware configurations, and software control features designed to accommodate the user's requirements.

### 2. Computer Communication Networks

A "computer communication network" is a system consisting of one or more computers and terminals, and a communications subsystem which connects them. [7] The primary purpose of this network is to facilitate the efficient flow of data, and provide the required supportive processing functions. The communications subsystem consists of transmission facilities and associated communications processors. Communications processors are computers dedicated to exclusively handling communications tasks.

The classification of computer communication networks often centers around the network topology, network connectivity, switching protocol and the degree of implementation of system-wide control features.

The term "centralized computer communication network" is used to define a network that possesses a high degree of centralized functions. Another network classification, "distributed systems", is used to describe a low degree of centralized functions.

The differences between centralized computer communication networks and distributed systems reveal themselves in the degree to which system functions are distributed. There are no concise metrics which delineate the exact classification of a network. The interpretation is subjective.

Further elaboration on specific advantages and disadvantages of the distribution of system functions and the definition of distributable functions can be found in CYSPER [6].

Centralized computer networks essentially formalize the system control structure into pre-selected system control nodes. These nodes contain what is often referred to as the network control programs (NCP). The NCP can control the processing, database management and communications management functions. This type of networking characteristically has a low degree of fault telerance because network direction eminates from only one node imposing network restrictions should this node be damaged.

Distributed systems, however, are typified by the distribution of system management functions. Although no

"pure" distributed systems exist in reality, the term is widely used to indicate a high degree of distributed functions within a network. This "peer" structure acts in a cooperative sense. Fouting algorithms, for example, rely on information that is "cooperatively" passed from node to neighboring node, thereby deriving information, not on a global basis but on a localized one. Distributed systems have a high degree of survivability and are, therefore, more fault tolerant than centralized computer communication networks. The advantage of distributed systems is, however, partially offset by the increased "overhead" necessary to coordinate system functions.

#### 3. Further Reading

Numerous terms used in networking contribute to confusion due to the lack of an industry-wide standardized taxonomy. The intent here was to define certain keywords used throughout this thesis. Further clarification of system's taxonomy, although important, is not discussed. Taxonomical studies recommended are [2], [9] and [23].

# D. PERFORMANCE PREDICTION AND MODELING

"Performance" is defined as the degree to which the system fulfills user requirements. In terms of networks, these measures are often determined by the network's workload capacity or throughput in the sense that the network can first perform the desired functions and secondly perform with a degree of timeliness. [4]

The ideal means of measuring network peformance is to extract data from the system itself. The collection of data from existing networks is often difficult. The performance testing of networks under heavy traffic loads or those operating in a degraded mode is a sensitive matter, because interference of the monitoring equipment can not be tolerated. Networks in design phases, of course, can not be measured and require alternative assessment methods.

The modeling of existing and planned networks has become an important component of performance evaluation. As an added benefit, modeling also facilitates the understanding of a system's design and interrelationships.

The modeling process can follow one of several different techniques or a combination thereof. The primary techniques are: 1) mathematical modeling and 2) simulation.

Mathematical modeling employs theories of queuing and flow by describing certain network characteristics in sets of equations. The process is, however, complex and often assumes away critical parameters. The primary disadvantage with mathematical modeling appears to be just this problem of assumptions. Too many assumptions impose an unacceptable degree of abstraction. Although the validity of mathematical modeling techniques has been confirmed [4]. the methodology is often understandable only by the modeler.

Simulation, the second alternative, leads the modeler to numerous techniques. A simulation is an abstraction of concepts pertinent to the problem being studied, and upon which the modeler can apply varying experimental variables. The model simulates only those features the modeler feels are relevent to the problem. Herein lies the critical danger of simulation. The danger may best be expressed by the question, Does the model bear relevancy to the real problem? [10]

The major problem of simulation, as well as analytical models, is therefore the validation of the model. Many simulation experts talk in terms of performance reliability factors but fail to state that these factors may be just derived from outputs of a model, and the closeness of the model to reality may or may not be substantiated.

The modeling process itself consists of the construction of the model, followed by the validation of the model, and finally modifications to the model based on results of the validation process.

Once the model has been defined, simulations are run with the model to evaluate network behavior. The problem is to predict peformance of real networks by evaluating behavior on the network models. The simulations then provide a means by which network design deficiencies can be identified and corrected.

Another issue in simulation analysis is the area of overdesign. Simulations should indicate those systems components that do not add to the capabilities of the network. This analysis can measure the device utilization of specific components within the network.

The key measurement of a computer communication network is the network's workload capacity or throughput. Throughput is generally measured in number of message units per time period, and provides a measure of effectiveness and efficiency of the system. The parameters involved in throughput are: 1) network configuration (topology), 2) network control algorithms, and 3) network reliability.

E. SUMMAPY

A solution to predicting performance in systems where the collection of data is difficult or the system is in design stages, is to build a model of the system. The model could then be tested over the entire sprectrum of performance specifications.

The goal of performance evaluation is the prediction of the degree to which the system fulfills the intended objectives. The major concern in computer communication networks is the degree to which the network can perform the task and the degree of efficiency with which the task is completed. A by-product of the evaluation should be the identification of areas of over and under-design. Once design failures have been identified, design tradeoff decisions can then be made. This process of performance analysis is aimed at optimizing the existing or planned network's performance and ensuring that the performance meets the contracted user requirements.

#### III. MILITARY APPLICATIONS OF DISTRIBUTED SYSTEM TECHNOLOGY

A. INTRODUCTION

This chapter discusses current military programs and research efforts that are applying the concept of automated networks to tactical missions. It is written to give the reader some background information on programs and terminology. With this background, the applicability of the simulations described in later chapters will be clearer.

B. PACKET SWITCHING

The transmission of computer to computer digital messages has had significant impact on communications switching techniques. In fact, the concept of packet switching was invented to a large extent because of the unique requirements of computer based systems. Packet switching was designed as an alternative to circuit switching.

The circuit switching technique of older communications systems is a method of establishing a route for communications traffic whereby a complete link between the calling and receiving station is set up and maintained exclusively for the exchange of those two stations. The connection is maintained until one of the stations breaks off transmission or reception. A technique such as this tends to be wasteful in computer communications because computer communications are typically "bursty" in nature; that is, the messages are very short in duration and require fast responses.

Packet switching is designed to make efficient use of a communications channel when the traffic is bursty. [11] In this technique messages are divided into discrete "packets." A packet is a block of information containing a fixed number of bits. Each packet contains the text of the message plus a control header. The header contains enough information (for example, source, destination, routing plan, message sequence number, etc.) to guarantee the packet will arrive at the proper destination. In addition, there will usually be some checks on each such block, so that any switch through which the packet passes may exercise some degree of error control.

Figure 1. shows a typical composition of a packet of information bits. This particular example is taken from a packet radio network. [16]

In a packet-switching network, the packet represents the fundamental unit of transportation. One message may be broken into several packets and each packet may be independently routed to its final destination. Of course, at the destination the packets must be re-assembled in the correct order to reconstruct the original message.

Because packets of the same message can be sent by different routes, congestion on the network can be decreased. Fach packet contains its own control information in the header, and there are no lengthy connection and disconnect times as in the case of circuit switched systems.



FIGURE 1 FORMAT OF PRNET PACKET (PRCAP3)

It should be noted that messages which are broken into packets are only meaningful within the network. When packets are passed through gateways into other packet-switched networks, a new intra-net level of protocol is required.

C. THE ARPANET

Perhaps the best known example in the military of a large scale distributed system is the ARPANET. This research effort has been sponsored by the Defense Department's Advanced Research Projects Agency (ARPA). The ARPANET is a non-secure, packet-switched, distributed computer communication network which links together the computing facilities at universities and military installations across the continental United States and reaches overseas to London and Hawaii.

The justification and advantages of a network such as the ARPANET are summarized in the following excerpts from the text <u>Computer-Communication Networks</u> written in 1973: [1]

"One of the most successful aspects of the experiments in the use of time-shared computer systems conducted during the past decade was the ability to share computing resources among all the users of the system. Controlled sharing of data and software, as well as the sharing of the time-sharing system hardware, has led to much higher programming productivity and better overall utilization of the computing and user resources." Even in these time-sharing systems, however, the system capacity was simply not large enough to perform all the storage requirements and computing potential that a decision maker required. There was the lack of a large enough community (critical mass phenomenon) in a single application area. Although it is possible to physically transfer programs or data from one community to another, this causes restrictions in language standards and hardware systems.

To quote further: "A viable alternative to program transferability, while permitting full rescurce sharing, is to provide a communications system that will permit users to access remote programs or data as if they were local users to that system. In addition, it should be possible for a user to create a program on his local machine that could make use of existing programs in the network as if they were available on his local machine. Pather than trying to move the programs from machine to machine, the network would allow the user or his pregram to communicate with a machines can be connected into such a network, the total community in any particular application area would be sufficiently large enough to reach critical mass." [1]

This, essentially, is the rationale for military applications of the ARPANET research, both on the strategic and tactical level.

# D. MILITARY APPLICATIONS

We have spoken in a previous chapter of the tremenious growth of automated data systems. Tactical data systems, be they for the purpose of intelligence, pescnnel management, fire direction, logistics, or command and control, tend to be engineered and developed separately without the consideration of total military mission. Nevertheless, the battlefield commander cannot make wise decisions based on input from just one of these systems. There is a good deal of interplay between all of these functional areas, and all must be considered.

One single tactical computer in a single command post could not be built to store and update all of the data represented in the combined systems. And if it could, such centralization would be unwise. The answer seems to be a distributed computer network built to interconnect and share the resources of the individual systems. This means overlaying an ARPANET-like architecture onto a series of distributed processors on the battlefield.

The APPANET distributed architecture recgraphically data and computing resources. This separates bases distribution tends to decentralize a network, moving from a hierarchical configuration to a grid or traditional mesh-type configuration. Such decentralization is important to overall system survivability and reliability. A network architecture such as this is capable of remaining operational if one or Tore nodes 15 rendered

non-operational. The architecture, combined with packet switching technology, allows for sufficient alternate routing capability to ensure a robust system. It offers significant improvement over some of the present "backbone" (hierarchical) systems in which the failure of one node along the chain would completely disrupt communications on the entire network.

It is obvious that a mobile and tactical application of ARPANTT technology would offer new challenges to system engineering. Most apparent is the fact that ARPANET sites are interconnected by high speed, low-error, fixed telephone circuits. This kind of interconnectivity is not possible on a dynamic battlefield. The only other alternative is to utilize mobile, digital radio equipment to achieve connectivity.

#### E. THE ALOHA SYSTEM

Several years are researchers at the University of Hawaii began work on such hardware. Because there was an unusually high error rate on the local telephone lines, remote users of the university computer were unable to effectively communicate with the computing facility. This led to a research program to investigate the use of burst radio transmission in place of telephone lines for error-free, line-of-sight communications to the computer center. The resulting effort became known as the Aloha Syster, a series of packet-switched, ultra-high frequency (UHF), radio terminals. [13]

The Aloha System is essentially a broadcast. multi-access network. The "broadcast rapability" of a radio channel implies that a signal generated by a radio transmitter may be received over a wide area by any number of receivers. "Multi-access capability" of a radio channel means that any number of users may transmit over a common channel. Hence, all users within line-of-sight of one form a network that is completely connected, another independent of the number of users.

#### F. PACKET RADIO INTRODUCTION

Research done at the University of Hawaii led to the development of packet radio. Packet Radio extends the Aloha System to military uses. The goals of military experimental versions of a packet radio system differ from those of the original Alcha net in the following areas: [13]

(1) Distributed control of the network management functions should be provided among multiple stations for reliability, and the use of a netted array of possibly redundant repeaters for area coverage as well as for reliability should be included.

(2) The system should use spread-spectrum signaling for coexistance with other possibly different systems in the same band and for antijam protection. Surface acoustic wave technology has become a viable current choice for matched filtering in the receiver.

(3) The provision of authenticaion and anti-deception mechanisms is required.

(4) System protocols should be incorporated that perform network mapping to locate and label repeaters, route determination and resource allocation, remote debugging, and other distributed network functions.

(5) The use of various implementation techniques to provide efficient operational equipment such as repeater power shutdown except while processing packets should be included.

Because packet radios operate within a broadcast network, and all the network radios use a common frequency band (1712-1650 MFz), this technology has some favorable implications for frequency management and frequency conservation. According to current military doctrine, the frequency spectrum is allocated roughly in accordance with each user's stated requirement. In an Army or Marine Division this results in a frequency management problem of too many nets requiring too few frequencies and a constant threat of degraded communications due to mutual inteference problems. Once a frequency is allocated for a particular mission, it is not available for use by others in the same area.

This might be an effective management technique if each assigned band were actually used most of the time. In practice this is not usually the case, and much of the frequency spectrum is idle (not engaged in carrying traffic).

A broadcast network in which a number of users share a common broad frequency band offers improvement to this situation. The limited frequency spectrum could be used more efficiently if (1) the shared frequency band was wide enough to allow all users to transmit required traffic, (2) a channel access scheme was defined such that all users could access the channel when needed while at the same time allowing little or no mutual interference, and (3) the channel usage was high enough to ensure minimum empty time when the channel was not in use.

#### G. CHANNEL ACCESS SCHEMES

One primary means of categorizing radio broadcast systems is the method employed for channel access. As mentioned earlier, packet communications have found important applications in ground-based radio information distribution, and in this situation there exists a common broadcast channel that is available and shared by a multiplicity of users. Because these users demand access to the channel at unpredictable times, some access scheme must be introduced to coordinate their use of the channel in a way which prevents degradations and mutual interference.

A large number of channel access ideas have been invented, analyzed and described in current literature. For a summary of these schemes, see [14]. All of these schemes, however, might be placed in one of three broad categories. [15] Fach category has its own advantages and costs.

# 1. Category I

The first category involves random access contention schemes whereby little or no control is exerted on the users in accessing the channel. This results in the occasional collision of packets on the air. A collision implies that at least one colliding packet is unintelligible and that channel usability for the time of the collision may be lost. Access schemes which fall into this first category are the pure Aloha, the slotted Aloha, and to a lesser extent. Carrier Sense Multiple Access. These are the access methods used by packet radio systems.

To better understand random access contention schemes, consider the example shown in Figure 2. There are some number of users, each of which transmits some number of packets of time duration "tau" at random times. Line four, labeled SUM, indicates the total traffic that all the users attempt to send in a fixed time.

In this example all traffic is able to be transmitted without conflict except for the collision indicated by the hashed area. These two packets (P-1 and X-1) may both be unrecognizable to the receiving station (at least one will be unreadable) and both could require retransmission.

In the non-slotted Aloha random access technique, packets are transmitted as soon as they reach the top of the transmit queue at the radio. No consideration is made of current channel activity. Therefore, they risk collision
with other packets on the air.

In the slotted Aloha rethod, time is broken into discrete quartities equal to the maximum time of propagation within the network. Users are restricted to transmission only at the beginning of each time slot. Again, collisions may be frequent.

In the carrier-sense mode, the radios listen before they talk and thereby reduce the risk of packet collisions. The radio senses the state of the channel before transmitting. If the channel is occupied, the radio waits a random amount of time and senses the state of the channel again before attempting to transmit.

An important measure of peformance for evaluating these random access techniques is "throughput." Throughput in packet radio technology is defined slightly differently than the definition given in Chapter II. Here it is defined as the percentage of time that the channel is actually occupied by useful traffic. Or, to put it another way, it is the message density on the channel. Is the channel carrying useful traffic (non-colliding packets) most of the time, or is there a lot of time wasted between packets when the channel is empty?

Figures 3 and 4 show how throughput is calculated in both techniques.







It should be obvious that throughput is higher in the carrier-sense mode due to the decreased number of collisions. In fact, analysis has shown that the maximum throughput possible with the non-slotted Aloha method is approximately 1/2e or 18.4%. Throughput in the slotted Aloha method is twice as high as the non-slotted method (36%). Throughputs as high as .80 to .90 have been obtained using the carrier-sense mode for cases in which the channel propagation time is small compared to message duration. Both of these methods are classified as asynchronous.

Another important measure of peformance for distributed computer systems is a low delay time. This means that queries and responses between the system and the user take a minimum amount of waiting time. A short delay time is important when there are many interactive users on the net. Typically, in order to decrease delay time packet length is shortened. Long packet lengths, however, are necessary to increase throughput, shorten queue lengths and decrease processing overhead per bit.

On the battlefield, calls for fire from a forward observer would typically be short messages requiring a fast response, whereas, intelligence summaries are normally several pages long and require no reply. If both types of messages are carried by the same communications channel, one can readily understand why there are tradeoffs between throughput and delay time.

### 2. Category II

At the opposite extreme of categories of radio channel access methods, there are schemes which use completely static reservation access methods. These schemes pre-assign capacity to users and effectively create "dedicated" as opposed to multi-access charnels. Such schemes as Time Division Multiple Access (TDMA). Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA) fall into this category. The Joint Tactical Information Distribution System (JTIDS), Phase I, and the U.S. Marine Corps Position Location Reporting System (PLFS) use the TIMA scheme in which time is broken down into discrete intervals. The largest time cycle (called an "epoch" in JTIDS) is divided into thousands of smaller time slots or windows. Each user on the broadcast retwork is assigned one or many time slots in which he can transmit messages. After the passage of the cyclic time period his time slot again appears and he can transmit again. This technique, obviously, is nighly dependent upon all users maintaining accurate time synchronization. The FDMA and CDMA scheres have also been developed to statically assign each user a fixed portion of the channel according to a unique frequency or code respectively. Here the problem is that a bursty user will often not use his preassigned capacity, in which case it is wasted. When a user is idle his portion of the channel cannot be used by other stations with traffic.

## 3. Category III

Between these two extremes the are dynamic reservation systems which only assign channel capacity to a user when he has data to send. In these schemes a certain portion of the channel is set aside in which to dynamically schedule transmission times. Several schemes fall into this category. In a Pollirg scheme the user waits passively to be asked if he has data to send. In an active reservation scheme the user asks for capacity when he needs it. In the Mini-Slotted Alternating Priority scheme a token is passed among numbered users in a prearranged sequence, giving each permission to transmit when he receives a token. The cost of these schemes is the overhead required to implement the dynamic reservations.

The following Figure 5. summarizes the costs of these three categories. [15]

Access Method	Collisions	Control Overhead	Idle Capacity
Random Access Contention	n yes	no	٥a
Dynamic Reservation	no	yes	ro
Fixed Allocation	no	no	yes

#### The Cost of Distributed Resources

H. CURRENT PROGRAMS

As was previously mentioned, before large scale distributed data systems can be introduced to a mobile battlefield, higher capacity communications hardware is required. The U.S. Army is working on two programs to meet this requirement. The two systems under development are the PLRS/JTIDS hybrid and packet radio.

An Army Letter of Agreement [25] which addresses the need for these systems reads: "There is an urgent need for communications capable of supporting existing and programmed automated systems for Air Defense, Field Artillery. Intelligence, and Command and Control. Characteristics of machine-to-machine communications, coupled with the need for fast reaction times and a high degree of mobility, result in requirement for a specialized distributed а data communications system. Without this data communications improvement, highly sophisticated and highly effective weapon systems fielded in the early 1982's will not operate to full potential."

In further describing present communications systems this document speaks of existing equipment as being "technologically old, generally manually connected, too large and immobile (multichannel), and requiring intensive maintenance and logistical support." "The current communication system cannot, without the addition of a digital data communications capability, meet the demand imposed by the emergence of automated systems in the early

1980's."

The precise extent of this communications shortfall is a matter of intense study by the Integrated Tactical Communications Study (INTACS) Update Study effort.

The PLRS/JTIDS hybrid offers a solution to this problem in the mid-1980's. Packet radio, also, is a promising technology to satisfy future tactical data distribution missions. Its development schedule, however, effectively eliminates it as a short term candidate.

1. PLRS/JTIDS Hybrid

A detailed description of the PLES/JTIDS hybrid is not possible here. Briefly stated, however, the system is a computer based system which provides real time, secure data communications, and position location and reporting information for tactical forces. It combines desirable features of both the PLRS an the JTIDS systems, using modified equipment from both systems. This system is planned for introduction to the field by 1986.

In addition to work being done on the PLRS/JTIDS hybrid, the Army is monitoring or participating in some interesting tests with packet radio, described in the following sections.

2. San Francisco Bay Experimental Packet Radio Network

The current experimental packet radio network being supervised by SRI International is located in the San Francisco Bay area and has been operational since July, 1976. Figure 6 [16] shows a map of the retwork sites.



There are two station PRU's located at the Menlo Park site. Each station has an associated PDP-11 computer attached. The network has four fixed repeater sites spread over the area and a variable number of packet radio user terminals (typically four to six). There are two vehicular packet radio terminals. These mobile terminals are an important aspect of the network, and a hand-off of a mobile terminal from one repeater to another is frequently exercised.

3. The Fort Bragg Test Bed

The U.S. Army has also recently set up a test bed for the evaluation of Packet Radio at Ft. Bragg, North Carolina. ARPA is sponsoring this effort, known as the Army Data Distribution System (ADDS). "The purpose of the ADDS experiment is to develop an environment in the resident XVIII Airborne Corps to permit user participation in the development, refinement and evaluation of innovative concepts for deployment of distributed data in support of future tactical Army data distribution requirements." [17]

In this multi-phased experiment, Ft. Brage is experiencing a step-by-step build up of resources. The first phase of the experiment began in January, 1979. Three computer terminals, a network processor (called a TIP-Terminal Interface Processor) and a host computer were installed at Ft. Bragg and connected via commercial telephone lines into the ARPANET. After installation, operator training began in 1-2 day courses.

The second phase began in April, 1979 and the number of terminals was increased to fifty. The training in these phases acquainted operators of all ranks with the basic preprogrammed capabilities of the ARPANET including electronic mail, file management, directory maintenance, text editing, and printing. More specific applications geared to tactical information flow requirements are also being tested in garrison.

Phase III of the ADDS experiment is currently going on and involves the introduction of Packet Radio into the testbed. The PR network at Ft. Bragg will eventually grow to approximately 20 radios and 2 control stations. Initially, the radios are being employed in garrison to replace hard-wired connections. In the future, they will be deployed to the field in support of Corps exercises.

Packet Radio is expected to fullfill at least two major roles in the Ft. Bragg testbed. The first is to determine if this communications technology will satisfy the tactical data communications requirements of a corps on the battlefield. In this role, the system will be placed into the testbed as would any other communications system proposed for Army use. A second role is to provide a broad band communications channel for other systems under development which requires a data transfer capability. In this role the PRNET would provide communications for TACFIEF or some other intelligence or air defense system while maintaining its ARPANET connectivity as well. [17]

Reports from the Ft. Bragg testbed indicate that the XVIII Airborne Corps personnel have rapidly and enthusiastically adapted to the computer based communications technology. It is possible that the use of this data distribution technology can become as routine to commanders and staffs as voice communications are today. [17]

In any case, the Ft. Bragg experiment represents an innovative and unique approach to investigating advanced concepts in Army doctrine and tactics. The testbed is a departure from traditional Army tests designed to arrive at production decisions, and is being driven by the urgent need for increased tactical data distribution capability.

### I. FUTURE IMPLICATIONS

۲¢ لا Is is not an exageration to say that these are very crucial days for the U.S. military. Resource investment decisions are more important now than ever before in history, and the consequences of bad judgement offer potential for great loss. Some would argue that an ever-increasing dependency upon technology by the armed forces is a very dangerous trend. Nevertheless, it is apparent that electronics and telecommunications advances are beginning to have significant impact upon strategy. Chapter One of the U.S. Army Combat Communications Field Manual is entitled "Command, Control, and Communications (C3)". A quotation from this document helps to stress the intensity of future battles and the necessity of accurate

and realistic planning for that engagement. [24]

"The U.S. Army has arrived at a point where technology and reality have outrun our old tactics on fighting and left ther in the dust. We have come to the shocking realization that the old way of doing things will not work any more."

"A good example of the change in combat reality facing today's soldier is an often used statistic from the Arab-Israeli War of 1973. In 20 days, over 1700 tarks were destroyed between the two sides. That's as many tanks as there are in five U.S. armored divisions. Technology has improved the weapons systems to the point where a tank has a 50-50 chance of being hit by the first round fired at it. We must retcol our tactics to meet the reality of the next fight."

Certainly the capabilities of an army's command and control system has a great influence upon the capability of the force as a whole. The trends and technologies described in this chapter have far-reaching implications to doctrine and pose some problems that have yet to be solved. It is beyond the scope of this thesis to dwell on these implications in great length. A listing of the most obvious ones, however, is interesting and instructive.

1. Chain of Cormand

1 **1** 

First of all, how will distributed systems change or affect the traditional chain of command structure? C3 system architectures typically reflect the chain of command within an organization, and this results in most C3 systems being very hierarchical in structure. Distributed systems tend to be grid-like, peer structures. In a distributed system, is it advantageous to practice the concept of "skip echelon" reporting, in which certain levels of command may send traffic directly to the highest or lowest elements without intermediate "information only" stops? Certainly this would tend to decrease the time of delivery of messages and would avoid congestion on the network. But can the intermediate commanders afford to miss information that might prove to be critical 0 F essential? What affect do distributed architectures have on the traditional role of communications centers and message centers? Would the requirement for these functions be eliminated? When the network is organized in a peer structure, how are protocols designed in order to preserve a "priority" system to message traffic? These are questions that remain to be answered.

### 2. Network Management

i ∳

Next, a very important question is the manner in which network control and data base management is exercised. Although distributed systems appear outwardly to be decentralized, a static, inflexible network control design would make the so-called "distributed" system as vulnerable as its hierarchical predecessor. Clearly the responsibility for network management and data base updating must be transferable within the network, and the capability should be shared by more than one station. There is a tradeoff reached, however, between one station and multi-station operation. The more nodes there are which can exercise network control, the more vulnerable the entire network becomes to spoofing and deception techniques, and the more costly and complex it becomes. It is interesting to note also, that some of the hardest decisions and longest delays in the JTIDS program have dealt with this question of network management.

## 3. How Much Redundancy?

In addition to these important questions one must ask also, "How much redundancy is enough?" Although distributed networks are more survivable and redundant, what kind of back-up systems are still required? Certainly an increased dependency upon satellite communications in today's world has decreased the investment in long-haul, high frequency (HF) communications systems. Some have argued with convincing reasoning that this is a dangerous position. It is generally more attractive to engineers and plenners to invent and employ new systems rather than to improve the old. Nevertheless, it is necessary to retain and maintain older, proven hardware as back-up, secondary equipment. The question remains, "Fow much is enough?"

### 4. Propagation Loss Due to Higher Frequency

Because of the congestion existing at lower frequencies and the high bandwidth requirements of new automated systems, new communications equipment is being designed to operate at very high carrier frequencies. This restricts propagation to strictly line-of-sight distances.

While this situation is acceptable in ground-to-air and ground-to-satellite communications, it poses serious constraints upon ground-to-ground mobile communications, especially in rugged or forested terrain. The amount of propagation loss in this situation is highly significant, and the utility of these systems has yet to be proven.

5. Management Information Systems

It was mentioned previously that the development of an effective Management Information System (MIS) to integrate and display data to a commander and his staff is a formidable task. The Army, in fact, has been wrestling with this problem for over twenty years while attempting to field its Tactical Operations System (TOS, which is, in essence a MIS). After that amount of time, one questions whether such a system is nearer completion now than it was in 1960.

6. Interoperability

The military is fast finding out that interoperability means much more than mutually compatible equipment. It also means compatability of procedures, software, and message formats. One should watch closely the continued development of JTIDS and attempt to judge the success of joint service programs. The interoperability requirement tends to introduce system complexity in an attempt to make the system "all things to all people." When one moves to the problem of interoperability in the NATO environment, the question again becomes, "How interoperable can equipment be without becoming too costly, too bulky and

### generally ineffective?"

7. Voice vs. Data Circuits

Another important question yet to be resolved is the proper trade-off of voice and data circuits. Again, JTIDS can be the case in point. Some services seem to be side-stepping procurement committments partly due to a lack of definition of system capability in this area. Although tactical commanders tend to prefer voice channels, voice channels require an enormously greater bandwidth allocation than data channels. If a commander is given the choice of having one voice channel or ten data channels into his command post, which will he choose? Which should he choose? What will the system offer?

8. System Cost

Finally, the question that will be asked most often is simply: "How much will the system cost?" Costs are divided into at least four categories. There are hardware and software costs, (it is common knowledge that the latter are now a greater consideration than the former) and there are initial procurement costs and life cycle costs.

Included in these costs is the manpower question. Is it realistic to think that as systems become more and more highly technical that the personnel who fix, operate and manage the systems will be better educated and more professionally competent?

## IV. AN INTRODUCTION TO PETRI-NETS

### A. INTRODUCTION

The purpose of this chapter is to introduce the reader to the particular modeling tool which forms the basis for this research. The history of Petri-Nets is first discussed. Then an explanation of how Petri-Nets work is presented. Following some simple examples, the chapter concludes with a brief summary of the Strengths and weaknesses of this modeling tool.

### B. HISTORY

The Petri-Net is named after its discoverer, Carl Adam Petri. These nets were developed in his early work in 1962 in Germany. They soon came to the attention of Anatol Holt who was then leading an Information Systems Theory Project for Applied Data Research. Inc. The work of this group eventually led to the theory of "systemics" [5] which dealt with the representation and subsequent analysis of systems and their behavior. At this point the modern formalism and notation of Petri-Nets was introduced. Holt also demonstrated the usefulness of the Petri-Net model in the representation of systems characterized by concurrent processes.

Perhaps the single largest source of research and literature regarding Petri-Nets has been Project MAC at the Massachusetts Institute of Technology. The Petri-Net model

was introduced to the researchers at Project MAC due to the association of Holt's group to J. Dennis' Computation Structures Group. This group has produced several PH.D thesis, together with many reports and technical memos dealing with Petri-Nets. In addition, MIT has sponsored two important conferences dealing with Petri-Nets. The first was the Project MAC Conference on Concurrent Systems and Parallel Computation held at Woods Hole in 1970. The second was the Conference on Petri-Nets and Related Methods, held at MIT in 1975.

This work, begun at MIT and continuing at other centers in the United States, until recently tended to concentrate on the formal or mathematical aspects of Petri-Nets. This work bears resemblance to the research in automata theory. It attempts to analyze systems by representing them as Petri-Nets, formally manipulating the representation in such a way as to derive information relating to the behavior of the modeled system. Because of the simplicity and power of Petri-Nets, they are excellent tools to use in the analysis of concurrent or asynchronous systems. They are finding their way into a number of diverse applications.

Petri, himself, is still actively researching, expanding his original theory. His extensions have led to a form of general systems theory called "net theory". Holt is continuing his research, concentrating on system representation and analysis of the formal representations. [5]

### C. HOW PETPI-NETS WORK

Simply stated, a Petri-Net is a model. More specifically, it is an abstract, formal model that analyzes the flow of information in systems. [5,19] Petri-Nets also describe not only the information flow, but the controls and constraints of such flow. A Petri-Net graph models the static structure of a system in much the same manner as a flowchart models the structure of a computer program. As a modeling tool, Petri-Nets are especially useful in modeling systems that exhibit asynchronous and concurrent activity.

A Petri-Net consists of a collection of "events" and "conditions." In graphical notation, conditions are conventionally represented by circles and events are represented by bars. The Petri-Net is given structure and the capacity for interaction by connecting events and conditions with arrows.

An arrow from a condition to an event signifies an input condition to that event and implies that every occurence of the event terminates the "holding" of that condition. An arrow from an event to a condition signifies an output condition, and in this case, the occurence of the event commences the holding of the output condition. The graphic notation for a condition which holds is the marking of that condition by a "token".

The behavior of a system may be thought of as the occurence of events as time progresses. If all input conditions to an event hold, the event can occur. This

results in the holding of the event's output conditions.

Conditions may also be called "places" or "nodes" and events may also be referred to as "transitions." In the Petri-Net model of a system, directed arcs connect places to transitions and transitions to places.

The Petri-Net is first given a particular structure of places and transitions, and then it is "executed." Execution is governed by a "firing" protocol.

Simply stated, a transition may "fire" (symbolizing the occurence of an event) when all input conditions or places into that transition are marked with a token. When all inputs into a transition are marked, the transition is said to be "enabled". Figure 7 shows an "enabled" transitior.

Execution of the Petri-Net involves the cyclic checking of all transitions once during each time interval. Fach transition that is found to be enabled is fired, and tokens are moved from the input places of the enabled transition to the output places of that transition. This procedure continues for a set number of iterations. The flow of tokens in the Petri-Net thus symbolizes the flow of information or control in the modeled system.

By devising special methods for marking the number of tokens at the Petri-Nets nodes, the system status can be accurately and effectively recorded. The state of the system is reflected by an ordered set of mark status indicators which correspond to the nodes of the graph structure.



In this thesis such an effective marking method is uniquely employed to give the viewer an accurate snapshot of network status.

Petri-Nets have rapidly gained acceptance over the last decade. Along with this aceptance has been the furthering of the understanding of Petri-Net properties.

## D. A SIMPLE EXAMPLE

Figures 8 through 11 show the various states of a simple Petri-Net as execution occurs during four successive time intervals. [21] Notice four places (P1, P2, P3, and P4) and three transitions (TR1, TR2, and TR3). The directed arcs denote the interaction and relationships between input and output conditions. For instance, TR1 will become enabled when P1 (its only input condition) is marked with a token. At the time that TR1 fires, the token will be removed from its input condition (P1) and placed in its two output conditions (P2 and P4). In this manner flow of information or control is followed through the modeled system. Figure & shows the network at time =  $\emptyset$  with one token placed in P1. Figures 9 through 11 depict the Petri-Net as it continues execution through time = 3.

### E. ADVANTAGES AND DISADVANTAGES

The following characteristics of Petri-Nets were found by the authors to be strengths when using this particular modeling tool for simulations in the context of this research:

# A MARKED PETRI-NET (TIME = 0)

- ---

1



Figure 8

# A MARKED PETRI-NET (TIME = 1)



Figure 9

# A MARKED PETRI-NET $(TIME \approx 2)$



Figure 10

# A MARKED PETRI-NET (TIME = 3)



Figure 11

1. The rules governing Petri-Net execution are simple and easy to understand. This methodology can be quickly grasped by those with non-technical backgrounds who would ordinarily be unable to understand mathematical or analytical modeling. Yet Petri-Nets retain a high degree of precision and accuracy.

2. There is much flexibility inherent in the Petri-Net graph to model wide ranges of complexity. For instance, a model can be further abstracted by the replacement of a complex network of nodes by a single node.

3. There is a large degree of flexibility in assigning time intervals during execution.

4. Petri-Nets are well-suited to "snapshot" portrayal of network states. This advantage is important in simulation languages and is considered a strong point of languages such as GPSS and SIMSCRIPT. Petri-Nets possess this characteristic by nature.

5. Petri-Nets nave the potential for a wide variety of uses. Basically, any process that can be flow-charted could be expressed by Petri-Nets. Applications could include: flow of information or control in an organization, information flow in electronics hardware, representation of computer software or, procedures and stages of development in a ranagement program.

Petri-Nets lend themselves well to computer graphics display.

7. Petri-Nets are very effective when modeling concurrent, asynchronous activity in a network or system.

Certain weaknesses also became apparent to the authors in the course of this work. They are listed as follows:

1. Although Petri-Nets are basically simple to understand, the small building blocks of a network soon become exceedingly large and complex when large systems are modeled. The input files to some of the larger simulations in this paper were more than 1400 lines long. These networks must be drawn on paper before their entry into the computer, and this kind of effort soon becomes very tedious and prone to error.

2. Petri-Nets are best suited to concurrent, asynchronous behavior. When non-concurrent, synchronous behavior in a system is modeled the Petri-Net assumes a large amount of overhead.

3. The fact that Petri-Nets are not generally well known in the computer communications community could be a disadvantage when the user wishes to prove the accuracy of his model.

4. The fact that the simulator employed in this thesis effort was deterministic could be considered either as an advantage or disadvantage depending upon the application. Many simulations have value because of their stochastic nature. Certain classes of experiments, nowever, need to be understood not because of "chance" happenings but because of the operation of the "laws" of nature working upon the

elements of the experiment.

As with any modeling technique, success is achieved through the modeler's familiarity with the modeling tool. Petri-Nets provide an excellent means to model those applications best characterized by their asynchronous, concurrent nature.

# V. PETRI-NET SIMULATIONS OF COMPUTER COMMUNICATION NETWORKS

### A. INTRODUCTION

Once the basic rules of Petri-Net execution are understood, it is a simple matter to apply these rules to a communications network. The system modeled is the network itself. The movement of tokens in the Petri-Net represent the flow of information within the system: in this case, message traffic in the network. Each token becomes a discrete amount of information contained within the message.

The places in the Petri-Net graph are used to represent communications nodes within the network. The directed arcs of the Petri-Net graph are used to represent the communications links or channels which interconnect the nodes. The transitions between nodes indicate the availability of the channel to carry traffic. If the transition is enabled, the channel is clear, and the message is relayed from input node to output node.

The careful structuring of the Petri-Net graph imposes upon the modeled system a variety of network protocols. An advantage of using Petri-Nets for simulation purposes is that the logic and protocol of the system are entirely contained in the structure of the Petri-Net graph rather than in a complicated mathematical algorithm within simulation software.

### B. TYPES OF NODES

The graphical output of the simulation attaches significance to the shape of the nodes displayed to the screen to facilitate recognition and interpretation. The experimental packet radio network in the San Francisco Fay area defines three primary types of nodes: terminals. stations, and repeaters. A terminal is a user node at which traffic is inputted or to which traffic is destined. It could be a fully automatic sensor, a handheld device, or a keyboard with CRT; but a terminal is a place where users connect to the network. The station is the node at which network control is exercised. The station typically keeps network statistics, monitors flow and congestion control, assigns routing, and performs data base management for the It is a terminal with increased processing system. capability usually provided by an attached mini-computer. The repeaters are stand-alone devices placed in numerous, dispersed positions throughout the network to act as relay sites. Repeaters do not act as origins or destinations of traffic, but they serve the purpose of extending the geographical range the network beyond a typical of line-of-sight distance.

While all networks do not use this identical terminology, these three functional nodes summarize the requirements of communication hardware in most networks. In the simulations in this thesis, three types of figures and labels represent the functions of nodes as described above.

## C. A SIMPLE APPLICATION EXAMPLE

In order to understand more completely the application of Petri-Nets to communications circuits, refer to Figure 12. In this diagram two distinct one-way communications channels are represented. The first goes from T1 to T2 and the second goes from T3 to T5 and through T4. The individual tokens are representative of packets of information in a packet switched environment. The three additional nodes forming a triangle in the center of the diagram impose a special firing order upon the transitions in the communications channel. These center noies are systems overhead which ensure that only one "packet" can be transmitted during a single time frame. In fact, if the three additional center nodes are thought of as a clock, then the entire network is a representation of Time Division Multiple Access in a network. A terminal can only transmit during a particular assigned time slot. After the time slot passes, the user must wait until the clock cycles back to his slot again. Because each transmitter has a unique time slot assignment, no two terminals can transmit during the same time, and collisions of packets on the radio broadcast channel are eliminated.

This explanation should give the reader a simple idea of the manner in which various protocols are represented. Obviously the Petri-Net in Figure 12 allows only non-concurrent activity on the communications channels.



Figure 12.

-----

The removal of the additional overhead would allow concurrent activity. In fact, this is sometimes desirable. For instance, those networks such as AUTODIN II which will make use of leased landlines will allow concurrent activity in the network. The Petri-Net is more efficient when modeling concurrent activity. The requirements to ensure non-concurrency as in Figure 12, causes one additional node and one additional transition to be placed in the Petri-Net graph for every transition on the communication links. Using the fundamental Petri-Net described here as a small building block, a system of considerable complexity can be built with many origin and destination terminals, and which allows packets to flow two directions with multiple, alternate paths from source to destination.

### D. RANDOMNESS IN PETRI-NETS

. ð ' í The Petri-Net simulator used in this thesis work is deterministic. After the simulator begins execution there is no means to interactively alter the sequence of events, and there is no element of randomness within the simulator. Because of this situation, the same input file will always give identical output. Although the capacity for alternate routing within the communications network is implemented, the tokens do not randomly "choose" their routes during execution. They can only follow their pre-assigned routes from origin to destination. This means effectively that fixed routing instead of adaptive routing is represented in the simulations. This is not necessarily a disadvantage,

however, as some simulations have shown better throughput and time of delivery results using fixed routing over adaptive routing [20].

It might be advantageous to modify the Petri-Net simulator to make it more stochastic in character. This might be accomplished in at least two ways. First, the inital marking of the network could be varied randomly at the beginning of each run. Certain key nodes could be marked or left unmarked according to the result of a call to some random number generator. Then the initial random state of the network would affect the end result of the output file.

A second way to add randomness deals with the Petri-Net concept of "dynamic conflicts" (see Figure 13, for an example of this particular network state). In this figure, the reader will notice that both Transitions 1 and 2 are enabled, but both cannot fire. Only one transition can fire, since in so doing it removes the token from T2 and disables the other transition. Thus TR1 and TR2 are said to be in conflict. This basic relationship can be used to create either deterministic or nondeterministic behavior. If the Petri-Net simulator is deterministic, the firing order for transitions in conflict is fixed according to a certain rule. This describes the case in this paper. The firing order of transitions is explicitly defined by their ordering in the input file. In the case of Figure 13, if TR1 is listed in the input file before TR2, and if a dynamic conflict occurs between ther. TR1 will always fire first and



TR2 will not fire during that time interval because it will have been disabled. This set of rules gives a strict priority of firing to the network.

The firing order of transitions in conflict could be modified so that it occurred in a random fashion. This would allow the Petri-Net to be executed in a non-deterministic manner and add the missing aspect of randomness to the outcome if so desired. In fact, Petri's first networks were non-determinestic because of this factor.

### E. MEMORY STORAGE REPRESENTATION

The amount of memory storage in any particular communications node is also easily represented in the simulation. In this work a maximum number of seven packets is allowed at any one node. This number reflects the buffer size of seven packets in packet radio technology. If at any time a node accummulates more than seven packets, the buffer size has been exceeded and packets would theoretically be lost. When the buffer space is exceeded, the number of overflow packets is displayed outside the node in a red warning color.

### F. SYSTEM LOAD AVERAGE

The system load on the network can be represented by the number of packets in transit at any one time interval. The system load then can be varied by controling the frequency of message generation. The shorter the interarrival time between generated messages, the busier the network will be. At some point the network will be saturated and unacceptably congested if message input is greater than message throughput.

It is a simple matter to construct a message generator using Petri-Nets. See Figure 14. In this figure Transition 1 is firing to two outputs. One can be thought of as external and one output. R1, can be considered internal to the network. The external node can represent entry into the communications channel. The internal output feeds a token back into the "generator" and will, therefore, fire other packets into the network at regular intervals. The other repeaters (R2, R3) can be thought of as delays which slow down the frequency of message generation. This configuration of places and transitions constitute a message generator. The frequency of generation can be staggered and then several generators may be placed at the input of every communications circuit. In this manner the system load on the network may be varied.

#### G. TIME PEPRESENTATION

The Petri-net model is very good at representing the net status at distinct time periods. In fact, each tire interval displayed to the screen gives an excellent "snapshot" of network status. This is an important advantage inherent to the Petri-Net simulation. Another advantage is the flexibility afforded in assigning the time interval. The user has the prerogative of making each time interval as long or as short as is necessary. Successive snapshots may
represent the passage of time of 1 millisecond, 120 milliseconds, or 1 hour, depending upon the application and network being modeled.



### VI. THE EXPERIMENT

## A. DESCRIPTION

experiment was designed to demonstrate the An feasibility of using the Petri-Net simulator as a predictor of network performance. The experiment was performed by keeping certain parameters constant and varying others. A series of six input files were run through the simulator. In all six files the network architecture and the fixed routes were kept constant. Figure 15 shows the network. There were five origins of message traffic and five destinations. Four nodes were designated as terminals plus one station. There were four repeaters that performed relay functions within the network. Fach of the five origins could send traffic to of four destinations. Each source to destination one combination had two routes for traffic to take. This made a total of 38 possible fixed routes in the network.

The controlled variables for the experiment were system load, concurrent vs. non-concurrent activity, and polling frequency of various circuits. The first three runs of the experiment were done with a high load. For the second three runs the message generators were slowed down to give a low system load. Some input files allowed concurrent network activity and some required non-concurrent activity. On two of the non-current runs the frequency of polling certain selected circuits was increased. This would represent the



equivalent of assigning a certain user more time slots in a TTMA scheme than another. It would give priority to those subscribers who have more traffic to send.

The only statistic gathered from the experiment was throughput measured in the number of packets which successfully reached their destination. This number could be extracted directly from the output queues at each terrinal or station. The following points sumarize the characteristics of each of the six input files:

1. Runs 22-24 were run at high load.

2. Funs 25-27 were run at low load.

3. Runs 22 and 25 exhibit concurrent network activity.

4. Runs 23 and 26 exhibit non-concurrent network activity with equal polling frequency of all circuits.

5. Runs 24 and 27 exhibit non-concurrent network activity with weighted polling on certain circuits that are terminated at T1.

### B. FESULTS

Figure 16 shows a summary of throughput statistics from the experiment. Certain results are no doubt obvious, but the quantitative nature of output statistics validates prior assumptions.

The following otervations are noteworthy;

1. Allowing concurrent activity on the network greatly increases throughput. The number of packets successfully transmitted in RUNS 22 and 25 was on the order of four times larger than the non-current runs. This question of

# THROUGHPUT IN NUMBER OF PACKETS

EIGH LOAD

	RUN22	PUN23	RUN24
T1	92	16	25
T2	62	16	15
T3	77	18	16
<b>T4</b>	58	18	17
51	62	17	16
TOTAL	352	85	89

LOW LOAD

	RUN25	RUN26	RUN 27
<b>T1</b>	36	17	19
<b>T</b> 2	38	17	16
<b>T</b> 3	35	18	16
<b>T4</b>	33	16	15
<b>S1</b>	_31_	17	_16
TOTAL	173	85	82

Figure 16.

KALL TY

concurrency has significant implications to radio broadcast systems. For instance, what would be the advantages gained in the Packet Radio Network if terminal radio output power were reduced so that the terminal could only talk to its nearest neighbor rather than the entire network? This situation could allow simultaneous transmission of packets without the threat of collision. Also, a multiplexing scheme within the network could allow concurrent activity. These types of considerations could be modeled easily with the appropriate modifications to the Petri-Net graph.

2. Increasing the frequency of polling on selected circuits increases the throughput of those circuits. The reader should note the number of packets received at T1 in RUNS 23, 24, 26, and 27. RUNS 24 and 27 show a slight increase because certain circuits destined for terminal T1 were polled more frequently. This situation could easily reflect the assignment of more time slots to certain priority subscribers in a TDMA scheme. Again this modification was performed simply by restructuring the input file to the Petri-Net simulator.

3. The reader will note that the total throughput in the non-concurrent runs is largely the same regardless of the high load - low load factor. This is because the system is basically "saturated" during non-concurrent activity at both high and low loading. Although the user might desire to improve throughput by generating more messages and trying to force them into the system, the network will give the same

results because it is already operating at full capacity.

4. A visual inspection of the output as PUN22 executes reveals that the system is essentially operating at peak efficiency under maximum load. The viewer will notice that the buffers at every location are frequently filled to capacity but seldom are overflowing.



### VII. RECOMMENDATIONS AND CONCLUSIONS

### A. RECOMMENDATIONS

The potential exists for significant follow-on work to this thesis. Major topics for future work include the following:

1. The development of a language to describe networks, and the inclusion in the software of a "front-erd" program that would make the input file less cumbersome. The program could be static, like a compiler, or interactive, writter to query the user about a number of basic network parameters. For instance: How many nodes do you desire in the network? What are the paths of traffic from source to destination? Do you wish to allow concurrent or non-current activity? The software would take the user responses to such parameters and construct the input file from the responses.

2. A statistics gathering package should be written to collect and collate vital network parameters as the simulator is executed. Such a package would keep a running total of such items as:

a. average number of packets at a node

b. average number of packets on a circuit

c. percent use of a circuit

d. average time delay between transmission and reception along each fixed route

e. number of messages lost

f. rumber of messages which successfully reach destination.

These statistics could be formatted and displayed by a post-processor.

3. As previously described, the addition of randomness to the simulator might be considered desirable, and , if so, the proper modifications could be added.

4. As the system approaches "production status", the entire question of model validation needs to be addressed carefully. Much could and indeed has been written on the subject of how to demonstrate your simulation is accurate. [10] As in most simulations there is a tradeoff between the degree of complexity represented in the model versus the largeness of the network. If a network is small or if only a small segment of the network is modeled, the degree of detail can be great. If the network is very large, however, the date storage capability might not allow the same level of detail.

There were two constraints to validation posed in this work. First, the memory capacity of the PTP-11/70 was stretched to the limit on the larger runs. The mini-computer offers 128,000 bytes of memory which are partitioned into thirds. This gives any one user 128/3 or 42.6 % bytes of memory. The graphics programs approached and then exceeded this bound before the work was completed. This forced the division of one program into two separate programs of 36% each. The file sizes for the output files from the Petri-Net simulator are also very large. The larger of the input files produced output files on the order of 300% bytes. Figure 17. shows the relationship between number of nodes in the Petri-Net versus the size of the output files. In order to make claims of simulation accuracy based on real world networks, more memory is needed.

It should be added at this point that limitation of memory caused changes in the graphics display programs and in the overall organization of the software. The potential exists in the actual Petri-Net simulator (written in fortran and discussed in Appendix A) for the execution of Petri-Vets of well over a thousand places. If the user does not require a graphics output from this software package, then larger networks can be simulated on the PIP 11/70.

Secondly, the networks that were of most interest to the authors are largely experimental, unproven technologies. The Autodin II network is not yet operational and no statistics are available for validation purposes. The JTIPS technology is likewise not established operationally and many of the system characteristics are classified. Packet radic, which formed a good deal of background for the simulation, is still in its infancy. Also, packet radic employs a random channel access scheme which is contrary to the deterministic nature of the Petri-Net simulator.

Because these are new technologies, new routing and flow control algorithms and a host of different kinds of protocol are presently being developed. It is difficult to model a

system that has not reached its production state. Pather than being concerned about modeling a particular network that is still in a process of change and then trying to prove the simulation valid, it seemed wiser to leave the models in a more general state. By executing a variety of different input files, the simulations demonstrate the feasibility of future validation.

B. CONCLUSIONS

The conclusions of the authors are fourfold:

1. First, computer communication networks can be meaningfully modeled with the use of Petri-Nets. The background research to this thesis discovered no previous work which employs Petri-nets in the manner described in this paper.

2. Secondly, Petri-Net models of networks can be executed and displayed effectively on a color graphics terminal. The results of such a simulation are more easily understood than the common, hard-copy outputs produced by most analytical or queuing theory simulations. The color graphics output also could have considerable educational value. Again, background research uncovered ro instance in which Petri-Nets were displayed and executed on a color graphics terminal.

3. Thirdly, and perhaps most importantly, the implementation of such a modeling technique in a production environment as a predictor of system performance appears

C 3

4. Fourthly, there appears to be considerable benefits to encouraging future, carry-on work in the subject matter of this thesis.



NUMBER OF PETRI-NET NODES IN THE INPUT FILE

Figure 17.

# APPENDIX A - USER INSTRUCTIONS FOR THE PETRI-NET SOFTWARE

A. INTRODUCTION

This chapter is written to describe certain procedures and syntax peculiar to the simulation software. Assuming that an interested student or faculty member is somewhat familiar with the theory and structure of Petri-Nets. the instructions in this section will allow him to apply the simulation and graphics output to his particular modeling problem.

Figure 18 shows the various components of the entire software package, the program source code sizes, the programming language, and the output files. The reader should refer to this figure as he reads the instructions in this appendix.

### B. THE INPUT FILE

• •

The input file written by the user contains all of the information necessary to uniquely describe the Petri-Net model. This file is read by the fortran fip program named. "simulator". The input file must be named "RUNYX, where XX can be any number from 01 through 90. There are three main divisions of the file.

Part	I	Places
Part	II	Transitions
Part	III	Marking





Figure 18.

Street,

Appendix B. shows a sample input file. The reader should refer to this example as he reads the following instructions.

### 1. Places

The first line of the file specifies the total number of places to be read in. Beginning on line number two, the names and locations of the places are listed. The following format applies:

Place Name .... X-cordinate .... Y-cordinate .... Plot Flag

The place name must be less than 12 characters long. Only the first two characters will be displayed as a label on the graphics terminal. There are unique instructions for labeling places in the multi-routing version. These will be discussed later.

The first letter of the name specifies the type of figure that will be displayed. The letter, "T", identifies a "terminal" and will appear as a circle on the output screen. The letter "S" identifies a "station" and will appear as a rectangle on the screen. The letter "R" identifies a "repeater" and will be displayed as a truncated triangle. Input and output queues can be represented by placing an "I" or "O" as the first character of the place name. These modes are displayed as small rectangles, large enough to contain a two digit number. Names may begin with letters other than those listed above. They will not, however, be displayed to the screen.

After the name is listed, on the same line, the screen location of the place is specified by means of an x, y coordinate system. The x and y values are in the range of 0-511, with the (0,0) point located at the top, left-hand corner of the graphics display unit.

The third item of information in the "places" line is a "plot on/off" entry. The user will frequently have places identified in the Petri-Net which are necessary control elements, but which do not need to be displayed on the output. A value of "1" will cause the place to be displayed with its label. A value of "0" is used for listing places which are not displayed to the output screen.

2. Transitions

After every place in the Petri-Net is listed (one line per place), the transitions are listed. As before, the first line specifies the total number of transitions to be read in.

Then, the transitions are each listed in a three-line format as follows:

Transition Name...X Coordinate...Y Coordinate...Plot Flag Places into Transition Places out of Transition

Transitions are named "TRXX", followed by their x, y coordinates, and a "1" or "0" to indicate whether they are to be displayed to the screen. The second line concerns the

input to the transition.

The first field of this line specifies the total number of inputs, and the following numbers indicate which places enter that transition. The numbers,

3 8 11 14

for instance, indicate that three places are inputs to this transition. The particular places are identified by their implicit, line-number ordering as entered in the list of places. In this example the three places are the 6th, 11th, and 14th places entered in the input file.

The third line of the transition entry concerns the outputs from that transition. The format is identical to that of the line above, i.e., the numbers

2 3 4

indicate that the transition fires to two outputs, the 3rd and 4th places listed in part one of the input file.

3. Initial Marking

After all the places and transitions are listed, the Petri-Net must be given its initial state of marking. The initial placement of tokens is specified by the following format:

MARK Line # of Place # of Tokens Thus, the entry "MARK 3 1" specifies that at the beginning of the simulation, Place 3 (the third entry in the list of places) is marked with one token. Several places may be marked initially, but each marking requires a separate line.

Every line in the input file begins in the first column. Do not indent the beginning character of the line. The fields within each line must be separated by one (or more) blank spaces. The final line of the input file is the command "END".

### 4. Execution

When the user completes the input file he should exit from the edit mode and is now ready to execute the program "simulator" by typing "simulator.out". The program will ask him which imput file he wishes to read, and the user responds by typing "RUNXX" (the file he previously created) and a <cr>.

### C. THE OUTPUT FILES

, •¥

¥

The program "simulator" produces six separate output files. When RUNØ1 is entered into "simulator", files named RUNØ1A, PUNØ1B, RUNØ1X, RUNØ1Y, and RUNØ1Z are produced. The files suffixed with 4 through C are formatted files. Files X through Z are unformatted. Files A and X contain the essential data structures that have been read in "simulator" from the input file. Files B and Y contair the markings for each place at successive time frames which will appear on the graphics output. Files C and Z contain information concerning which links or transitions fire at any particular time frame and are used to highlight present activity on the screen. The graphics display programs, written in "C", require unformatted input files. The formatted files are necessary for the user to validate that correct input data is reaching the graphics program, and to troubleshoot when locating a problem. Examples of these files are contained in Appendices C through E.

D. USER OPTIONS

### 1. Choice of Programs

After the program "simulator" has executed the input file, the "C" programs read the output files and display the results of the simulation to the screen. At this point the user has several options concerning the method of display.

There are two separate programs the user can select--"transgraph" and "linkgraph". By executing "transgraph" the viewer will be able to observe the nodes of the network together with their associated transitions. "Linkgraph" does not display the transitions, but links node-to-node in the common way that communications networks are most frequently represented. For simple networks or to explain the basic working of Petri-Nets, the user will probably desire to see the transitions. "Transgraph" will only run with less than 100 places. For more intricate networks that contain several hundred places and transitions, the "linknode" program is necessary to avoid congestion on the screen.

### 2. User Questions and Responses

After selecting which program to run, the user is given a series of questions from the CRT. Question one asks the user to select which input file he wishes to execute. He responds with the command RUN X. For instance, if the input file named RUN21 has been executed by "simulator", and the user wishes to see the results, he enters RUN21X in response to question one. (Numerous files may be waiting in the users' directory which can be called in for display.)

Question 2 asks the user if he wants to view the data structures of the program. By entering a "1", the data structures will be printed to the CRT. A "2" will move on to the next question without viewing the data structures.

Cuestion 3 asks the user which of three versions of the program he wishes to see. Version 1 displays the marking of tokens in the conventional Petri-Net fashion, with numbers printed at the center of the places. Version 2 represents tokens by single, yellow boxes printed inside the nodes. These boxes are designed to represent "packets" of information in the packet switching concept. As previously described, each node has the capacity to hold seven packets. If the number of packets goes over seven, a red overflow number appears beside the saturated place.

The third version is the multi-routing, multi-destination version. Version 3 uses color in a unique way. Because packets may be originating at different nodes and traveling to several destinations, the linking channels require two-way transmission. The graphics display is color-coded to highlight this information. Packets traveling to a particular place are colored to match the label of that

place. For instance, when a green box appears in the network, the viewer can trace its progress to the destination whose label is displayed in green. In Version 3 (multi-routing), a backet originating at a certain node and destined for another marticular node may take different paths to arrive at the destination.

The user at this point must select one of the three versions by entering a "1", "2", or "3" followed by a  $\langle cr \rangle$ .

Question 4 asks the user to select one of the three Genisco graphics terminals in the C3 lab on which the output will appear. Correct entries to this query are "0", "1", or "2".

The fifth question asks the user if he desires a time pause of two seconds duration between time frames of the simulation. If he desires the capability to look closely at each network snapshot, he enters a "1". If not, he should enter a "0" and the simulation will run without pauses. This gives the viewer more of a "real-time" impression.

At the end of this final question, the screen displays the initial condition of the network. After noting the initial condition, the user should type a carriage return to continue execution.

If the user has typed a "1" in response to question 5, he also has the ability to indefinitely suspend execution of the program at any time frame. He can interrupt the program by typing a "PRK" from the CRT keyboard. After studying that particular snapshot of network status, ne may

continue normal execution by typing a carriage return.

This interrupt does not work if the program is executing without pauses. Typing two consecutive "PRK's" will enable the user to exit the program entirely.

3. The "Highlighting" Feature

The linking algorithm in both "transgraph" and "linkgraph" draws lines in a dark blue color. The highlighting feature in both programs changes the blue connecting link color to bright yellow on those links which are carrying traffic at any particular time frame. This feature performs in the following sequence. (1) The link lights up at the point where the future action will cocur. (2) The packet(s) in question moves from one end of the highlighted link to the other. (3) The highlight remains on the link to emphasize where the action occurred. The user will notice that direction of movement on the highlighted link is indicated by an arrow pointing in the appropriate direction. See Figures 19 and 22.

E. UNIQUE INSTRUCTIONS FOR VERSION 3:

In order for the user to implement the capabilities of Version 3, special network design information must be included in the input file. This paragraph describes these special instructions.

The simulation is structured to function in a fixed routing manner. That is, the multi-routed paths are predefined by the user.





Figure 22.

ļ



There can be as many alternate paths from source to destination as the user wants to include. But once these paths are specified in the input file, they do not change dynamically during execution.

The multi-routing feature is made possible by "stacking" numerous places on top of one another and displaying these piggy-backed places at the same coordinates. In this manner the nodes appear on the screen as a single place, although in reality they may be buried several deep.

As the total network is conceptualized by the user, he must begin by mapping out all origins, all destinations and all relay nodes. Every node becomes unique to a particular path. For instance, a simple case would be to send a packet from T9 to T5 (see Figure 21.) by two different routes--Poute 1 goes through R1 and Route 2 goes through R2.

In this case T9 and T5 would be "stacked" two deep. The routes are T9-P1-T5 and T9-R2-T5. Because they are plotted at the same point, the terminals appear as a single node.

The "header" information to perform this routing is contained in the name of the place. In Version 3, every node that will be displayed to the screen must be assigned a seven unit name in the input file.

The first unit of the name specifies the type of figure to be displayed ("T", "S", "R", "I", "O" as previously explained). The second unit specifies the color that the name will be printed in. This number is derived from the particular color table ( 16 colors are in a color table ) that is being used by the graphics program. The color of the label must correspond to the color of the packets bound for that route's destination. Units 3 and 4 of the name designate the route number that the node lies upon. Route numbers are arbitarily given by the user and utilized for his own identification purposes. Unit 5 designates the color of the packet that will travel along that particular route. Every node on that route have the same color designator. The packet color of the route is determined by the destination of that route. Units 6 and 7 of the name specify how many nodes are stacked at that location. Places that are stacked must be listed together in the input file.

An example of this 7 unit name might appear as follows:

### T205419

Field 1 designates that a circle will be drawn.

Field 2 specifies that the label will be displayed in color number 2 of the program's color table.

Fields 3 and 4 show the place on route number 05.

Field 5 ensures that every packet which passes through this place will be displayed in color number 4.

Fields 6 and 7 specify that places are stacked 19 deep at this coordinate.

As the program executes, the stacking algorithm totals all the packets which are located at a particular stacked location and displays that total number of boxes in the appropriate colors.



Figure 21.

1 v 2 44

RUN03	Page 1	Fri Feb	8 05:09:47	1980
1 10 2 T1 3 T2 4 T3 5 T4 6 G1 7 G2 8 G3 9 G4 10 G5 11 G6 12 8	$\begin{array}{cccccccccccccccccccccccccccccccccccc$			
13 TR1 14 2 1	250 175 1			
15 1 16 TR 17 2 18 2 4 19 TR 20 1	4 2 250 325 1 2 3 · 4 8 3 0 0 0 5			
21 2 22 TR 23 1	1 6 4 0 0 0 6			
24 1 25 TR 26 1	10 5 0 0 0 7			
27 2 28 TR 29 1	29 6000 8			
30 1 31 TR 32 1 33 1 34 TR	3 7 0 0 0 9 7 88 0 0 0		•	
35 1 36 1 37 M4	10 5 ARK 5 1			
38 MA 39 MA 40 EI	ARK 7 1 ARK 3 1 ND			

# Prod3A Prod 1 The first is is is is is interest of the state is interest.

ļ

.]

Ì

.

1

RUN	38	3	P	ag	e	1		F	r i	Feb	8	05:19:47	1980
1	1	1	1 0	0	1	0	0	1	0				
Ś	0	Q	1 1	. Q	Q	1	0	0	1				
5	0	1	1 1		6	0	0	1	0				
4	1		1 2		1	1	1						
5	1	1	1 1	1	ň	ĭ	ň	5	0				
7	ĭ	1	1 1		ĭ		ŏ	ĭ	ň				
8	ò	ō	1 4	Ō	ò	ĭ	ŏ	ō	ĭ				
9	ŏ	1	1 4	1	Ō	Ō	Ö	ī	Ō				
10	1	Ò	0 5	0	1	1	1	0	0				
11	1	1	1 5	0	0	0	0	1	1				
12	0	0	1 6	1	0	1	0	0	0				
13	1	1	1 6	0	1	0	0	1	0				
14	0	Ç	1 7	0	0	1	0	0	1				
15	0	1	17	1	0	0	0	1	0				
10	1			0	1	1	1	•					
1.0	1	1	1 0		ň		Ň	1	1				
19	ĭ	1	1 0		ĩ	ò	ŏ	1	ň				
20	ò	0	110	0	ō	ĭ	ŏ	ò	ĭ				
21	ŏ	ĭ	110	1	ō	ō	ō	ī	ō	•			
25	1	ō	011	Ö	1	1	1	Õ	Ō				
23	1	1	111	0	0	0	0	1	1				
24	0	0	112	1	0	1	0	0	0				
25	1	1	112	0	1	0	0	1	0				
26	0	0	113	0	0	1	0	0	1				
27	0	1	113	1	0	0	0	1	0				
28	1	0	019	0	1	1	1	0	0				
29	1	1	119	0	0	0	0	1	1				
30			113	1 : 1	1	1	0	1	~				
31	Å		116	0		ĭ	ň	ò	1				
33	ň	ĭ	116	Ĩ	ŏ	ò	ŏ	ĭ	ò				
34	ī	ō	017	0	1	ī	1	ō	Õ				
35	1	1	117	0	Ō	Ō	Ő	1	1				
36	0	0	118	1	0	1	0	0	0				
37	1	1	118	0	1	0	0	1	0				
38	0	0	119	0	0	1	0	0	1				
39	0	1	119	1	0	0	0	1	0				
40	1	0	020	0	1	1	1	0	.0				
41	1	2	120	1	0	•	0	1	7				
42	1	ĩ	121		1	0	0	ĭ	ň				
43	1		122	Ň	6	ĭ	ň	0	ĭ				
45	ŏ	ĭ	122	1	ŏ	ò	ŏ	1	ò				
46	ī	ō	023	ō	ī	ī	Ĩ	ō	õ				
47	1	1	123	Ō	0	Õ	0	1	1				
48	0	0	124	1	9	1	0	0	0				
49	1	1	124	0	1	0	0	1	0				
50	0	0	125	i 0	0	1	0	0	1				

E

RUN03C	Page	1	Fri	Feb	8	05:19:49	1980
· 5 t							
235							
4147							
52658							
73 8237						•	
93							
10 4 5 6							
12 1 7 8							
14 3 5							
16 1 4 7							
172 1858							
193							
21 3				•			
22 4 5 6							
24 1 7 8							
26 3 5							
28 1 4 7					_		
29 2 30 5 8							
31 3						•••	
33 3							
35 3							
36178 372							
38 3 5 19 3							
40 1 4 7							
41 2 42 5 8							
433 44237							
45 3						•	
47 3							
48 1 7 8 49 2							
50 3 5 51 3		!					
52 1 4 7							
54 5 8							
55 3 56 2 3 7							
57 3 58 4 6 A							
59 3							
60 1 7 8							

RUN03C	Page	2	Fri	Feb	8	05:19:49	1980
61 2 62 3 5 63 3							
64 1 4 7 65 2							
673 68237						•	
693 70456							
713 72178 732							
74 3 5 75 3							
76 1 4 7 77 2 78 5 8						•	
793 80237				•			
81 3 82 4 5 6							
835 84178 852							•
86 3 5 87 3							
88 1 4 7 89 2 90 5 8		•					
91 3 92 2 3 7							
933 94456							
96 1 7 8 97 2							
98 3 5 99 3							
100 1 4 7			•				

A Care

	•						-	-	
RUN	20	Pag	e 1	ł	<sup>p</sup> ri	Feb	8	05:17:04	1980
1	122								·
Ž	\$3012	12	250	100	1				
- 3	\$3022	12	250	100	1				
4	\$3031	15	250	100	1				
5	\$3041	12	250	100	1				
6	53050	12.	250	100	I				
/ A	53000	12	250	100	1				
, q	\$3083	12	250	100	i				
10	\$3153	12	250	100	i				
11	53163	15	250	100	1				
15	\$3193	15	250	100	1				
13	\$ \$ 203	12	250	100	1				
14	P1022	12	75	200					
16	P1101	05	75	200	i				
17	R1142	05	75	200	i				
18	R1153	05	75	200	1				
19	R2050	05	425	200	1				
50	R2083	)5	425	200	1				
21	R2120	)5 - )5	425	200	1				
22	P2202	י כי וכי	423	200	1				
24	12012	16	250	300	i				
25	12022	16	250	300	i				
50	T2041	16 1	250	300	1				
27	15090.	16 /	250	300	1				
56	12073	16	250	300	1		•		
29	12083	16	250	500	1			<b>-</b> ,	
21	12041		270	300	1				
32	T2110	16	250	300	i				
33	T2120	16	250	300	1				
34	T2132	10	250	300	1				
35	T2142	16	250	300	1				
30	12165	16	250	500	1				
37	12182	10 1	250	300	1				
39	12193	16	250	300	i				
40	T1031	<b>D</b> A	100	400	1				
41	T1041	80	100	400	1				
42	11091	8	100	400	1				
45	111010	28 ' \e	100	400	1				
44	T1142	DA DA	100	400	1				
46	T1153	58	100	400	i				
47	T1163	18	100	400	ĩ				
48	10050	8	400	400	1				
49	T0060	8	400	400	1				
50	T0110	. 90	400	400	1				
52	101200	70 ( )A /	400	400	1				
53	T0182	08	400	400	i				
54	T0193	18	400	400	ī				
55	102030	8	400	400	1				
56	81		100	100	0				
57	52		100	100	0				
50 E0	53 84		100	100	0				
74	85		100	100	ŏ				
94			* * *	1.4.6	v				

RUNZO	Page 2	Fri	Feb	8	05:17
61 B6	100	100 0			
62 87	100	100 0			
02 00	100	100 0			
45 810	100	100 0			
66 B11	100	100 0			
67 812	100	100 0			
68 813	100	100 0			
69 B14	100	100 0			
70 B15	100	100 0			
71 B16	100	100 0			
72 817	100	100 0			
73 818	100	100 0			
74 819	100	100 0			
75 820	100	100 0			
70 021	100	100 0			
78 823	100	100 0			•
79 B24	100	100 0			
80 825	100	100 0			
81 826	100	100 0			
82 827	100	100 0			
63 528	100	100 0			
84 829	100	100 0			
85 630	100	100 0			
86 531	100	100 0			
07 032 AA All	100	100 0			
89 B34	100	100 0		•	
90 A1	500	500 0			
91 A2	500	500 0			
92 A3	500	500 0			
93 A4	500	500 0			
94 A5	500	500 0			
95 AD	500	500 0			
90 A/ 97 AA	500	500 0			
98 49	500	500 0			
99 A10	500	500 0			
100 A11	500	500 0.			
101 A12	500	500 0			
102 A13	500	500 0			
103 A14	500	500 0			
104 415	500	500 0			
105 410	500	500 0			
107 418	500	500 0			
108 A19	500	500 0			
109 A20	500	500 0			
110 A21	500	500 0			
111 A22	500	500 0			
112 423	500	500 0			
113 A24	500	0 000 0 500 0			
115 A24	500	500 0			
116 427	500	500 0			
117 A28	500	500 0			
118 A29	500	500 0			
119 A30	500	500 0			
120 A31	500	) 500 0			

:04 1980

103
RUN20	Page 3	F	ri Fe	<del>в</del>	05:17:04
121 412	500	500	0		
121 432	500	500	0		
128 433	500	500	Ō		
124 68					
125 TR1	220	200	1		
126 2 1	56				
127 1 23					
128 TR2	140	140	1		
129 2 2	58				
130 1 13					
131 TR3	140	260	1		
132 2 13	60				
133 1 24			•		
134 TR4	100	100	0		
135 2 3	62				
136 1 14	• • • •	100	٥		
13/ 187	100	100	v		•
130 2 14	04				
137 1 37	100	100	0		
141 2 4	66		•		
142 1 25					
143 TR7	. 100	100	0		
144 2 25	i 68	•			
145 1 40	)				
146 TR8	330	160	1		
147 2 5	70				
148 1 18	3				
149 TR9	375	5 320	1		
150 2 18	3 72				· · -
151 1 47	1				
152 TR10	0 100	) 100	0		
153 2 6	74				
154 1 20	5		^		
155 TH1	1 100	100	v		
150 2 20	o / o				
15/ 1 40	0 2 28)	200	1		
150 1410	7 79	200	•		
157 2 2	, ,,				
161 181	3 36'	5 260	1		
162 2 2	8 80				
163 1 1	9				
164 TR1	4 10	0 100	0		
165 2 1	9 82				
166 1 8					
167 TR1	5 17	5 400	1		
168 2 2	984				
169 1 4	1		•		
170 TR1	6 17	0 230	1		
171 2 3	0 56				
172 1 1	ר ז י	0 100	1		
175 181	7 7 5 8 A	0 300	•		
174 2 1	2 00				
175 1 4	с А 115	0 300	1		
177 2 6	u 92		-		
178 1 2	2				
179 181	9 36	0 140	1		
180 2 2	2 90		-		
107 6 6					

RUN20	Page 4	F	ri	Feb	8	05:17:04
181 1 12						
182 TR20	100	100	0			
183 2 53	96					
184 1 38						
185 TR21	100	100	0			
186 2 38	94					
187 1 11						
198 1855	100	100	0			
189 2 52	100					
190/1 21						
191 TR23	330	230	1			
192 2 21	98					
193 1 37						
194 TR24	325	335	1			
195 2 51	102					
196 1 36						
197 TR25	100	100	0			
198 2 46	106					
199 1 35						
200 TR26	100	100	0	•		
201 2 35	104					
202 1 10						
203 TH27	100	350	1			
204 2 45	108					
205 1 17						
206 TR28	170	160	1			
207 2 17	110					
208 1 9						
209 1829	100	100	0			
210 2 44	112					· · _
211 1 16						
212 TR30	100	100	0			
213 2 16	114					
214 1 34						
215 TR31	180	310	1			
216 2 43	116					
217 1 33						
218 TR32	100	100	0			
219 2 32	120					
250 1 50			•			
221 TR33	190	100	0			
555 S 50	118					
223 1 50						
224 TR34	300	400	1			
225 2 31	122					
226 1 49						
227 TR35	100	100	0			
228 1 55						
229 2 57	56					
230 TR36	100	100	0			
231 1 57						
232 2 59	58					
233 1R37	100	100	0			
234 1 59						
532 5 90	61					
236 TR38	100	100	0			
237 1 61						
538 5 63	62					
239 1439	100	100	0			
240 1 63						

RUN20	Page	5	F	ri	Feb	8	05:1	7:04
241 2 65 242 TR40 243 1 65	64 1	00	100	0				
244 2 67 245 TR41 246 1 67	66 1	00	100	0				
247 2 69 248 TR42 249 1 69	68 1	00	100	0				
250 2 71 251 TR43 252 1 71	70	00	100	0				
253 2 73 254 TR44 255 1 73 256 2 75	12 74	00	100	0				
257 TR45 258 1 75 259 2 77	1 76	00	100	0				
260 TR40 261 1 77 262 2 79	1 78	00	100	0		•		
263 TR47 264 1 79 265 2 81	1 80	00	100	0				
267 1 81 268 2 83 269 TR49	82	00	100	õ				
270 1 83 271 2 85 272 TR50	84	00	100	0				÷
273 1 85 274 2 87 275 TR51	86 1	100	100	0				
278 1 87 277 2 89 278 TR52 279 1 89	88	100	100	0				
280 2 91 281 TR53 282 1 91	92	100	100	0				
283 2 93 284 TR54 285 1 93	90	100	100	0				
280 2 93 287 TR55 288 1 95 289 2 97	94	100	100	0				
290 TR56 291 1 97 292 2 99	100	100	100	0				
293 TR57 294 1 99 295 2 10	1 98	100	100	0				
297 1 10 298 2 10 299 TRS	)1  3 10	2 100	100	õ				
300 1 10	3	-						

RUN20	Page 6	F	٢ì	Feb	8	05:17:0
701 7	05 106					
- 301 C 1	LOS 100	100	a			
302 100			•			
303 1 1	103					
304 6 104	107 104	100	0			
305 1	107		•			
300 1	109 108					
307 2 3	62 100	100	0			
300 1	109		-			
310 2	111 110					
311 TR	63 100	100	0			
312 1	111					
313 2	113 112					
314 TR	64 100	100	0			
315 1	113					
316 2	115 114			•		
317 TR	65 100	100	0			
318 1	115					
319 2	117 116					
320 TR	66 100	100	0		•	
321 1	117					
322 2	119 120					
323 TR	67 100	100	0			
324 1	119					
325 2	121 118					
326 TR	68 100	100	0			
327 1	121					
328 2	55 122					
329 MA	RK 1 1					
330 MA	RK 2 1					
331 MA	RK 3 1	•				
332 MA	RK 4 1					
333 MA	RK 5 1					
334 MA	RK 6 1					
335 MA	RF 27 1					
336 MA	RK 28 1					
337 MA	RK 29 1					
338 MA	RK 50 1					
339 MA	ARK 31 I					
540 MA	PRA 32 1					
341 MA	ARK 43 1					
542 MA	ARK 44 1					
343 MA	AKK 93 1					
244 MA	APK 51 1					
347 M	APK 52 1					
340 94	ARK 53 1					
ZAR MA	ARK 54 1					
349 M	ARK 61 1					
350 F	ND OIL					
334 61	••					

Page 6 Fri Feb 8 05:17:04 1980

i

12345	1	WELLTHE PROBRAMS & TEXT		
5 7 8	FILE	DESCRIPTION	VERSION	EXECUTION
9		· · · · · · · ·		
10	DISECTORY	LIST OF THESTS FILLS	•	AB DIRECTORY
11	RU1.01x	GETERATOR	2	transgraph
15	HU1-02X	E SHLE THANSITION	2	transgraph
1.3	R11.05X	ar SHC COMPLICIS	2	transgraph
14	*++0++X	HTCRASCHICAL STRUCTURE	2	linkgraph
15	RUNDAY	HIFTSCHICAL SIRUCIURE	5	transgraph
16	RUNDAR	AT G STRUCTURE	2	linkaraph
17	R11.07Y	FFFNHACH RELEPATOR	1	transgraph
14		DEWILLS CANCENDATION	1	transgraph
10	#0511#	THE MERTPLE DEPOILS (5)	5	transgraph
50	R11.,12×	ALL- HADRA T TEPUT	2	transaraph
51	H10.13X	COL3 - FRANKCONCER	2	transgraph
55	PU514X	CO SUMPREPARADUCER	2	transgraph
23	RU-15X	THIAL RUN HELDAX	5	linkgraph
51	RUHIOK	SYNCHEOMIZATION II	2	transgraph
25	₽0 <i>2</i> 9X	насертныхоты сётабек	3	linkaraoh
20	K11921X	ST PER THETTHR HTTPAG	3	link/transgraph
27	F11155x	CHOC PREME RETADRA	3	linkaraph
28	41/11/23X	JT F-SUCTIED COMMUNICATIONS	i 3	linkoraph
Sa	6U124X	FREPRITY TILE-SEDITED	3	linkaraph
3,0	R11- 25x	CO CHRRENT RETAINE (-)	3	linkgraph
31	AU-26⊀	EFFERSIONTED COMMENTEATION	(-) 3	linkgraph
3.2	PU/27.	991-1114 T1-8-81(11F8 (-)	3	linkaraph
33	R11561-29	I HEL FILER TO SIMULATOR	•	simulator.out
34	advisors.c	THEOIS ADVISORS	•	aav <b>isors</b>
35	end.c	ETHE FRAME FOR FILM	•	end
30	exceriment.c	THE REPORT RESULTS OF EXPRE	시 -	experiment
37	intro.c	INTHINGTION FOR FILM	•	intro
<b>қ</b> н	linear.c	UNCERFEILE SIZE RELATIONSHI	<del>ہ</del> د	linear
30	linkingeh.c	LO SHLISH GRAPHICS	-	linkaraph
10	003.0	TITLE FRACE FOR 198	-	0C\$
:1	outline.c	SERT, AND THE SLAS	-	outline
42	situlator	PETELE ET STULATOR	-	simulator.out
43	title.c	TABLE BOARD BOARDER	-	title
4.4	trans-reach.c	LEV STITUT GRAPHICS	-	transgraph

108

1.1

Page 1 Fri Feb 8 07:10:27 1980 simulator PROGRAM SIMULATOR 1 2 3 C ORIGINAL VERSION OF THIS PROGRAM (PPOGRAM TESTAT) WRITTEN BY L.A.COXO Modified to operate on unix (PDP 11/70) by S.C.JENNINGS & R.J.HARTELO 4 C 5 C PROGRAM SINULATOR READS USER INPUT FILE AND PRODUCES 6 OUTPUT FILES 6 7 с MAINLINE 8 9 CALL INIT CALL INPUTE 10 CALL DUMPPP 11 12 CALL MOVENET(150) CALL EXIT 13 END 14 15 16 17 SUBROUTINE INIT 18 19 C INIT OPENS USEP IMPUT FILE & CREATES 6 OUTPUT FILES -- RUN.. ENAMI STORES THE IMPUT FILE -----20 С FNAM2 STORES THE FORMATTED INPUT DATA STRUCTURES ----- RUN...A 21 C FNAM3 STORES THE FORMATTED ITERATIONS OF THE NETWORK - RUN..B FNAM4 SIGRES THE FORMATTED LINKS OF THE NETWORK ----- RUN..C 22 C 23 C 24 C 25 C FNAM7 STORES THE UNFORMATTED GRAPHICS STATES ----- RUN .. Z 26 C 27 28 29 BYTE FNAM1 FNAM2 30 BYTE 31 BYTE FNAM3 32 BYTE FNAM4 BYTE FNAMS 33 FNAM6 34 BYTE 35 BYTE FNAM7 36 37 COMMON/USHFILE/FNAM1(6), FNAM2(7), FNAM3(7), FNAM4(7), FNAM5(7), FNAM6(7)0 FNAM7(7) 38 1 COMMON/EVENT/JEVENT(400,6),NXTEVT 39 40 COMMON/TRANS/ITRANS(400,7), NXTTRN, IINTR, IISTOPE(100) 41 BYTE NAMES COMMON/NAME/NAMES(4000) , NXTNAM 42 COMMON/IUTAB/IOTABL(4000),NXTTRE 43 1000 FORMAT(' INITIALIZING PROGRAM') 44 TYPE 1000 45 1002 FORMAT(' BEGIN TEST-GRAPH-NET') 46 TYPE 1002 47 2007 TYPE 2000 48 2000 FORMAT 49 (\* \*\*\*-> INPUT FILE? NAME MUST BE ENTERED AS: RUNO1 - RUNOD ACCEPT 2001, FNAML 50 51 2001 FORMAT (6A1) 52 53 FNAM1(6) = 0OPEN (UNIT = 1, NAME = FNAM1, TYPE = 'OLD', ERR = 2006) GO TO 2004 TYPE 2002, FNAM1 54 55 2006 56 2002 FORMAT (' ERROR OPENING FILE ', X6A1) 57 GO TO 2007 2004 DO 2005 I = 1, 5 58 59 FNAM2(1) = FNAM1(1)FHAM3(I) = FNAM1(I) 60

109

Fri Feb 8 07:10:27 1980 simulator Page 2 FNAM4(I) = FNAM1(I)61 FNAM5(I) = FNAM1(I)65 FRAME(I) = FRAMI(I)63 FNAM7(I) = FNAM1(I)64 65 2005 CONTINUE 66 FNAM2(6) = 'A' 67 FNAM2(7) = 0FNAM3(6) = 'B' 68 69 FNAM3(7) = 0FNAM4(6) = 'C' 70 FNAM4(7) = 071 72 FNAM5(6) = 'X' FNAM5(7) = 073 FNAM6(6) = 'Y' 74 FNAM6(7) = 075 FNAM7(6) = 'Z' 76 77 FNAM7(7) = 078 NXTEVT=1 79 NXTTRN=1 NXTMAM=1 80 NXTTRE=1 3000 FURMAT(' INITIALIZATION COMPLETE') 81 82 TYPE 3000 Α3 84 RETURN 85 END 86 87 SUBROUTINE DUMPPP 88 89 90 C OPENS FNAM2 & FNAM4 91 BYTE FNAM1 92 BYTE FNAM2 93 BYTE FNAM3 94 **FNAM** 8YTF 95 BYTE FNAMS 96 BYTE FNAM6 97 BYTE FNAM7 98 CUMMUN/USRFILE/FNAM1(6), FNAM2(7), FNAM3(7), FNAM4(7), FNAM5(7), FNAM6(7) 99 FNAM7(7) 1 COMMON/EVENT/IFVENT(400,6),NXTEVT 100 101 COMMON/TRANS/ITRANS(400,7), NXTTRN, IINTR, IISTORE(100) 102 COMMON/IDTAB/IOTABL(4000), NXTTRE BYTE NAMES 103 COMMIN /NAME /NAMES(4000), NXTNAM CFE4(UHIT=1, NAME=FNAM2, TYPE='NEH', INITIALSIZE=40000) 104 105 1000 FORMAT(' NXTEVT=', 14) 106 1001 FOPMAT(5x,618) 107 108 WRITE(1,1000) NXTEVT 109 DU 1500 1=1,NXTEVT-1 1500 WRITE(1,1001) (IEVENT(1,J),J=1,6) 110 2000 FORMAT(/, ' NXTTRN=', 14) 111 2001 FOPMAT(1X,718) 112 WRITE(1,2000) NXITRN 113 114 00 2500 I=1,NXTTRN-1 115 WRITE(1,2001) (1TRANS(1,J),J=1,7) 2500 CONTINUE 116 117 3000 FORMAT(/,' IOTABLE:',/,60(1014,/)) 118 119 WRITE(1,3000) (INTABL(I), I=1, NXTTRE) 120 4000 FORMAT(/, ' NAMES: NXTNAM=',14)

-

-----

simulator Page 3 Fri Feb 8 07:10:27 1980 4001 FORMAT(X,100A1) 121 WRITE(1,4000) NATNAM 125 WPITE(1,4001) (NAMES(I),I=1,NXTNAM) 123 5000 FORMAT(1H1) 124 125 #RITE(1,5000) CLOSE(UNIT=1,DISPOSE='SAVE') 126 127 OPENCUNIT=1, NAME=FNAMS, TYPE='NEW', INITIALSIZE=14000, 128 129 1 FORM="UNFORMATTED") 130 WRITE(1) (NXTEVT) 131 DO 5001 1=1,NXTEVT-1 132 5001 WRITE(1) (1EVENT(1,J),J=1,6) WRITE(1) (NXTTRN) 133 DO 5002 I=1,NXTTRN-1 5002 WRITE(1) (ITRANS(I,J),J=1,7) 134 135 130 WRITE(1) (NXTTRE) 137 wRITE(1) (IUTABL(1),I=1,NXTTRE) 138 WRITE(1) (NXTNAM) 139 WRITE(1) (NAMES(1), I=1, NXTNAM) 140 CLOSE (UNIT=1, DISPOSE='SAVE') 141 RETURN 142 END 143 144 145 SUBROUTINE INPUTI 146 COMMON/SCAN/INORD(15,10), NUMBER 147 8000 FORMAT( INPUT BEGINS!) 148 8001 FORMAT(' INPUT COMPLETE\*) 149 150 **TYPE 8000** 151 1=0 152 CALL SCANR 153 CALL XINIGP(1, I) 00 1000 J=1,I 154 CALL INPUTE 155 156 1000 CONTINUE 157 158 CALL INPUTT 159 2000 CONTINUE 160 161 CALL SCANR IF('AATCHS(1,'END',3).ER.1) GO TO 3000 JF(MATCHS(1,'MARK',4).EQ.1) CALL MARKER 195 163 164 60 TO 2000 165 3000 TYPE 8001 166 CLOSE (UNIT=1, DISPOSE='SAVE') 167 168 RETURN 169 END 170 171 SUBROUTINE INPUTE 172 173 174 COMMON/EVENT/IEVENT(400,6),NXTEVT 175 SYTE IWORD 176 COMMON/SCAN/INORD(15,10),NUMB 177 J≥0 178 C READ A SINGLE EVENT LINE FROM INPUT AND 179 C STURE IT APPROPRIATELY CALL SCANR 180

111

NAN 254

Page 4 Fri Feb 8 07:10:27 1980 simulator CALL STONAM(1, IEVENT(NXTEVT, 1), IEVENT(NXTEVT, 6)) 181 182 183 DO 1000 I=2,4 184 CALL XINTGR(I,J) 1000 IEVENT (NXTEVT, I+1)=J 185 186 NXTEVT=NXTEVT+1 187 188 IF(NXTEVT.GT.409) GO TO 9000 189 RETURN 190 9000 TYPE 9900 191 9900 FORMAT(' EVENT/PLACE TABLE OVERFLOW') 192 CALL EXIT 193 RETURN 194 END 195 196 197 SUBROUTINE INPUTT 198 199 COMMON/TRANS/ITRANS(400,7), NXTTRN, IINTR, IISTORE(100) 200 BYTE INORD COMMON/SCAN/INORD(15,10),NUMB 201 202 203 1=0 204 K=0 205 CALL SCANR 206 CALL XINTGR(1,I) 207 DO 1000 J=1,I 208 209 CALL SCANR CALL STONAM(1, ITRANS(NXTTRN, 1), ITRANS(NXTTRN, 7)) 210 211 212 DO 2000 L=1.3 LL=L+3 CALL XINTG7(L+1,K) 213 214 215 2000 ITRANS(NXTTPN,LL)=K 216 217 CALL SCANR CALL STUIDT(NUMB, ITRANS(NXTTRN, 2)) CALL SCANR 218 219 220 CALL STOTOT (NUMB, ITRANS (NXTTRN, 3)) 155 NXITRN=NXITRN+1 222 IF(NXTTRN.GT.400) GO TO 9000 223 1000 CONTINUE 224 RETURN 225 559 227 3000 TYPE 9900 855 9900 FORMAT(' TRANSITION TABLE OVERFLOW') 229 CALL EXIT RETURN 230 231 END 232 233 234 SUBROUTINE STONAM (NWORD, NPOINT, KOUNT) 235 STORE STRING 'NWORD' FROM SCANNER INTO 236 C NAME TABLE AND RETURN A POINTER 'NPOINT' 237 C 238 239 BYTE INORD, NAMES, HLANK 240 COMMON/SCAN/IMORD(15,10),NUMB

ŝ

112

Fri Feb 8 07:10:27 1980 simulator Page 5 241 COMMON/NAME/NAMES(4000), NXTNAM DATA BLANK/IH / 242 DO 1000 I=1,10 243 244 KOUNT=I-1 IF(INORD(NNORD, I).EQ.BLANK) GO TO 2000 245 1000 CONTINUE 246 247 2000 CONTINUE 248 IF (NXTNAM+KUUNT .GT.4000) GU TO 9000 249 220 DO 3000 I=1,KOUNT 251 3000 NAMES(NXTNAM+I-1)=INORD(NWORD,I) 252 253 NPOINT=NXTNAM 254 NXTNAM=NXTNAM+KOUNT 255 256 RETURN 257 259 9000 TYPE 9900 9900 FORMAT(' NAME TABLE OVERFLOW') 259 290 CALL EXIT RETURN 261 END 595 593 264 SUBROUTINE STOIDT(NUMBER, LINK) 265 599 COMMON/IUTAB/IOTABL(4000),NXTTRE 267 268 STORE IMPUTS AND OUTPUTS OF TRANSITIONS 269 C IN THE TABLE, RETURN THE LINK 270 C 271 IF(NXTTRE+NUMBER .GT. 4000) GO TO 9000 272 273 274 K=0 275 DO 1000 I=1,NUMBER CALL XINTGR(I,K) 276 277 J=(NXTTRE=1)+I 1000 IDTABL(J)=K 278 279 LINK=NXTTRE ' 280 281 NXTIRE=NXTTRE+NUMBER 282 RETURN 283 9000 TYPE 9900 284 9900 FORMAT(' IO TABLE OVERFLOW (TRANSITIONS)') 285 CALL EXIT 286 287 RETURN 288 END 289 290 291 SUBROUTINE MARKER 292 293 COMMON/EVENT/IEVENT(400,6),NXTEVT 294 1=0 295 J=0 CALL XINTGR(2,1) 296 297 CALL XINTGR(3,J) 298 IF(J.LT.1 .OR. I.GT.NXTEVT) RETURN 209 IEVENT(1,2)=J 300 RETURN

113

and the second second

```
simulator
              Page 6
                         Fri Feb 8 07:10:27 1980
301
           END
302
303
304
305
           SUBROUTINE MOVENET(NTIMES)
306
           EXECUTE THE PETRI-NET 'NTIMES' OR STEPS
307 C
30R
309
           COMMON/TRANS/ITRANS(400,7), MXTTRN, IINTR, IISTORE(100)
           COMMON/EVENT/IEVENT(400,6), NXTEVT
310
           BYTE
                     FNAM1
311
315
           BYTE
                     ENAM2
                     FNAM3
313
           BYTE
314
           BYTE
                     FNAM4
           BYTE
                     FNAMS
315
316
317
           BYTE
                     FNAM6
                     ENAM7
           BYTE
           COM 40N/USRFILE/FNAM1(6), FNAM2(7), FNAM3(7), FNAM4(7), FNAM5(7), FNA46(7)
318
319
                           FNAM7(7)
          1
          DATA ITIME/1/
320
321
     1000 FORMAT(' EXECUTING TIME=',14)
322
323
           OPEN(UNITE1, NAME=FNAM3, TYPE='NFW', INITIALSIZE=120000)
324
           OPEN(UNIT=2, NAME=FNAM6, TYPE='NEW', FORM='UNFORMATTED',
325
326
327
            INITIALSIZE=120000)
          1
           OPEN(UNIT=3,NAME=FHAM4,TYPE='NEA',INITIALSIZE=12000)
          DPEN(UNIT=4, NAME=FNAM7, TYPE='NEW', FORM='UNFORMATTED',
328
329
          1 INITIALSIZE=12000)
330
331
332
333
          DO 2000 I=1,NTIMES
          TYPE 1000, ITIME
           1INTR=0
334
335
          CALL MOVE
336
      100 FORMAT(3512)
337
          wR1TE(1,106) (IEVENT(J,2), J=1, NXTEVT-1)
338
          wRITE(2) (IEVENT(J,2),J=1,NXTEVT-1)
      107 FORMAT(13)
339
          WP1TE(3,107)
                         IINTR
340
      108 FORMAT(10013)
          WRITE(3,108) (IISTORE(J), J=1,IINTR)
WRITE(4) IINTR
341
342
343
344
           ARITE(4) (IISTORE(J), J=1, IINTR)
345
346
          ITIVE=ITIME+1
     2000 CONTINUE
347
348
          CLOSE (UNIT=4, DISPOSE='SAVE')
349
          CLOSE (UNIT=3, DISPOSE='SAVE')
350
           CLOSE (UNIT=2, OISPOSE='SAVE')
351
352
          CLOSE(UNIT=1,DISPUSE='SAVE')
353
354
          RETURN
355
          END
356
357
358
          SUBROUTINE MOVE
359
360 C
          EXECUTE THE NET ONE STEP
```

٠J

1 1

114

The Marine State

```
Fri Feb 8 07:10:27 1980
simulator
              Page 7
361
362
           COMMON/TRANS/ITRANS(400,7), NXTTRN, IINTR, IISTORE(100)
           DIMENSION MARKS(400)
363
           ITEST=0
364
365
           CHECK ALL TRANSITIONS TO SEE WHICH ARE ENABLED
366 C
367
           DO 0500 I=1,NXTTRN=1
     0500 MARKS(I)=NABLED(I)
368
369
           DD 1000 I=1,NXTTRN-1
     IF (MARKS(1).ER.0) GO TO 1000
0600 FORMAT(' DYNAMIC CONFLICT, TR#=',14)
370
371
372
373
           IF(NABLED(I).EQ.1) GO TO 0800
374
           TYPE 0600.1
375
           GO TO 1000
     0800 CONTINUE
376
           CALL UNMARK (I, ITEST)
377
378
           IF(ITEST_EQ.1) GO TO 1000
379
           IINTR=IINTR+1
380
           IISTORE(IINTR)=I .
           CALL MARKEM(I)
381
     1000 CONTINUE
382
383
           RETURN
384
385
           END
386
387
388
           FUNCTION NABLED (NUMBER)
389
           RETURN 1 IF TRANSITION # 'NUMBER' IS ENABLED, READY
390 C
391 C
           TO FIRE. ELSE RETURN 0.
392
393
           COMMON/TRANS/ITRANS(400,7),NXTTRN
           COMMON/EVENT/IEVENT(400,6), NXTEVT
394
395
           COMMON/INTAB/INTABL(4000),NXTTRE
396
397 C
           CHECK LIST OF INPUTS TO SEE IF ALL ARE MARKED
398
399
           MARK=0
400
           IPT=ITRANS(NUMBER,2)
           KOUNT=IOTABL(IPT)
401
402
403
           00 1000 I=IPT+1, IPT+KOUNT
           NEVENT=IOTABL(I)
404
405
           IF(IEVENT(NEVENT,2).GT.0) MARK=MARK+1
     1000 CONTINUE
406
407
408
           NABLED=0
409
           IF (KOUNT.EQ.MARK) NABLED=1
410
           PETURN
           END
411
412
413
           SUBROUTINE UNMARK (NUMBER, IERROR)
414
415 C
                 UNMARK (IE. DECREMENT THE NUMBER OF TOKENS
All of the input events to transition # 'Number'.
416 C
417
    Ċ
                      RETURN TERROP=0 .....UNLESS.....
ONE EVENT IS A MULTIPLE INPUT OF THE SAME
418 C
419
    C
                      THANSITION AND WE DONT HAVE ENOUGH MARKERS.
420 C
```

A444322

simulator Page 8 Fri Feb 8 07:10:27 1980 421 C WHEN THIS HAPPENS, REPLACE ANY REMOVED 422 C TOKENS AND RETURN IERROR=1. 423 C COMMON/EVENT/IEVENT(400,6),NXTEVT 424 425 COMMON/THANS/ITHANS(400,7),NXTTRN COMMON/IOTAB/IOTABL(4000), NXTTRE 426 427 428 IPT=ITRANS(NUMBER,2) KOUNT=IOTABL(IPT) 429 430 TERROR=0 431 DO 1000 I=IPT+1, IPT+KOUNT 432 433 NEVENT=IUTABL(I) 434 J=I 435 IEVENT(WEVENI,2)=IEVENT(NEVENI,2)-1 IF(IEVENT(NEVENT,2).LT.0) GO TO 2000 436 437 1000 CUNTINUE 438 RETURN 439 2000 CONTINUE 440 00 3000 I=IPT+1,J 441 442 NEVENT=IOTABL(I) 3000 IEVENT(NEVENT,2)=IEVENT(NEVENT,2)+1 443 444 IERROR=1 445 446 RETURN 447 END 448 449 450 SUBROUTINE MARKEM(NUMBER) 451 C MARK ALL OUTPUT EVENTS OF TRANSITION # 'NUMBER' 452 C 453 C 454 COMMON/EVENT/IEVENT(400,6),NXTEVT COMMON/TRANS/ITRANS(400,7),NXTTRN 455 COMMON/IOTAB/IOTABL(4000),NXTTRE 455 457 458 IPT=ITRAWS(NUMPER,3) KOUNT=IUTABL(IPT) 459 DO 1000 I=IPT+1, IPT+KOUNT 460 461 NEVENT=IOTABL(I) 462 1000 IEVENT(NEVENT,2)=IEVENT(NEVENT,2)+1 463 RETURN 464 END 465 SUBROUTINE SCANR 466 467 468 BYTE IMORD, ISC, IBLANK 469 CUMMON/SCAN/IWORD(15,10),NUMBER 470 BYTE NHUFFR 471 COMMON/SCAN1/NBUFFR(80) DATA ISC/1H;/ 472 473 DATA IBLANK/1H / 0001 FORMAT(80A1) 474 475 READ(1,0001,END=9999,ERR=9999) (NBUFFR(I),I=1,80) 476 IP01NT=1 SET POINTER TO FIRST CHARACTER IN THE RUFFER 477 C NOW PROCESS THE FIRST 15 TOKENS DELIMITED BY EITHER 478 C 479 C A BLANK (OR MULTIPLE BLANKS) OR A SEMICOLON. 480 C

116

ų

. i

```
simulator
              Page 9
                         Fri Feb 8 07:10:27 1980
           DO 0002 NUMBER=1,15
481
482
           IFLAG=0
           SET INORD (NUMBER, X) = IBLANK (SET WORD TO ALL BLANKS)
483 C
           00 0003 I=1,10
484
485
     0003 IWORD(NUMBER, 1)=IBLANK
486 C
           START SCANNING LINE FROM POINTER ON TO FIND NON-BLANK
           KOU'IT=1
487
           "KOUNT" KEEPS TRACK OF THE NO. OF CHAR. IN THE TOKEN
DO 0004 KPOINT=IPOINT,80
488 C
489
490
           IF (NBUFFR(KPOINT).NE.IBLANK .AND. NBUFFR(KPOINT).NE.ISC)
491
                      GO TO 0005
          1
492
           IF(IFLAG.E0.0) GO TO 0004
           IF(IFLAG.EQ.1) GO TO 0006
493
494
     0005 CONTINUE
495
           IFLAG=1
496
           IWORD (NUMBER, KOUNT) = NBUFFR (KPOINT)
497
           KOUMT=KOUNT+1
498
           IF(KOUNT.GT.10) GO TO 0006
499
     0004 CONTINUE
500
     0006 CONTINUE
501
          END OF TOKEN FOUND, RESET SOME POINTERS
502 C
           IPUINT=KPDINT+1
503
           IF(IPOINT.GT.80) GO TO 0010
504
505
     3002 CONTINUE
506
507 C
          END OF BASIC TOKEN GETTING LOOP
508
509
    0010 NUMBER=NUMBER-1
510
          RETURN
511
     9999 CONTINUE
     END OF FILE OR I/O ERROR DETECTED
9998 Format(' Eof or Error on Scanner input from UNIT 1')
512 C
513
           TYPE 9998
514
           NUMBER=0
515
516
          RETURN
517
           END
518
519
520
521
           SUBROUTINE XINTGR(NWORD, IVALUE)
522
               CONVERT THE ENTRY IN "IWORD" TO INTEGER
523 C
524 C
                  RETURN INTEGER "IVALUE"
525
526
           BYTE INORD
           COMMON/SCAN/IWORD(15,10),NUMBER
527
           BYTE ISTRNG
528
529
           DIMENSION ISTRNG(10)
           BYTE IBLANK
530
           DATA IBLANK/1H /
531
532
           DO 0001 I=1,10
533
           KOUNT=I
           TSTRNG(I)=IWORD(NWORD,I)
534
535
           IF(IWORD(NWORD,I).E9.IBLANK) GO TO 1000
536
     0001 CONTINUE
537
     1000 CONTINUE
           KOUNT=KOUNT-1
53A
539
540
    2004 FORMAT(X,10A1)
```

117

sim	ulator	Page 10	Fri Feb 80	07:10:27 1980
541	2005	FORMAT(X,1110)		
542		DO 2006 I=1,KO	JNT	
543		J=11-1	.*	
544		K=(KOUNT+1)-I		
545	2006	TSTRNG(J)=TSTR	NG(K)	×
546		L=10-KOUNT	•	
547		DG 2007,I=1,L	· ·	
548	2007	TSTRNG(I)=IBLA	(K	``````````````````````````````````````
549		OPEN(UNIT=2,NA	E='ISTORE',	TYPE='NEW', INITIALSIZE=20)
550		WRITE(2,2004)	(TSIRNG(I),I	=1,10)
551		CLOSE(UNIT=2,D	ISPUSE 🕿 SAVE	• )
552		OPEN(UNIT=2,NA	E='ISTORE',	TYPE='OLD')
553		READ(2,2005) 1	ALUE	
554		CLOSE(UNIT=2,D	SPOSE='DELE	TE')
555		RETURN		•
556		END		
557			A	
558			3	
559		FUNCTION MATCH	S(NUMB,STRIN	G, NCHAR)
560			•	
561	Ç	THIS FUNCT	ON DETERMIN	ES IF SCANNER TOKEN
562	ç	INORD (NUMB	MATCHES TH	E CHARACTERS IN "STRING"
563	C	AT LEAST FI	IR THE FIRST	"NCHAR" CHARACTERS.
564	_			
565	C	IF THERE IS	ња МАТСН, I	T RETURNES THE INTEGER "1"
566	C	NO MATCH RI	TURNS "0".	
567				
568		RYTE IWORD		
569		COMMON/SCAN/IWO	RD(15,10),N	UMBER
570		BYTE STRING		
571		DIMENSION SIRIN	G(10)	
572		MATCHS=0		
5/5		<b>BB B B B B B B B B </b>		
5/4		00 0001 1=1,NC	AN	
5/5		TF(INUMD(NUMB,)	).NE.STRING	(I)) RETURN
5/0	0001	LUNTINUE		
5//	~			
570	L	ATTING AN ATTING	C, INEY AER	E INE SAME
5/4		MAICH331 Detion		
70V 501		REFURN ,		
201		ENV		
502				
SAL	r	END OF PONCOAM	STMULATOR	
204	-	LIV OF FRUURAM		

<u>.</u>

Page 1 Fri Feb 8 05:01:00 1980 transgraph.c 1 # 2 3 4 5 /\*\*\*\*\*\*\* PROGRAM TRANSGRAPH.C 67 /\*\*\*\*\*\* \*\*\*\*\*\*\*\* /\*\*\*\*\*\*\* STEPHEN C. JENNINGS JC91 USMC 8 /\*\*\*\*\*\* 9 ROBERT J. HARTEL CS91 USA /\*\*\*\*\*\*\* 10 /\*\*\*\*\*\*\* \*\*\*\*\*\*\* WRITTEN FALL QUARTER 1979 11 /\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*/ NAVAL PUSTGRADUATE SCHOOL 12 /\*\*\*\*\*\*\* MONTEREY, CALIFORNIA 13 /\*\*\*\*\*\*\* \*\*\*\*\*\*\* 14 /\*\*\*\*\*\*\* \*\*\*\*\*\*\* 15 16 17 18 19 /\* 20 /\*\* EXTERNAL DECLARATIONS \*\*/ 22 53 24 /\*\*\* LITERALS \*\*\*/ 25 26 #define header 4 27 #sefine pictures 50 28 #define bounds 100 29 #define limit 500 30 31 32 /\*\*\* STRUCTURES \*\*\*/ 33 34 struct { /\* data structure information on net nodes .. \*/ 35 /\* store control char not used in program ... \*/ 36 int strill? 37 int namentri /\* index to names array ..... \*/ /\* initial marker state of the network ..... \*/ 38 int marker; /\* x cordinate of place ..... \*/ 39 int xcord; 40 /\* v condinate of place ..... \*/ int ycord; /\* whether or not place is to be plotted .... \*/ 41 int plot: /\* length of name associated with place ..... \*/ 42 int length? 43 44 }file1 (hounds) . \*hp1; /\* pointer into data structure ..... \*/ 45 46 47 struct 4 /\* data structure information on transitions. \*/ . 48 49 int ctr127 /\* store control char not used in program ... \*/ /\* index to names array ..... 50 •••• \*/ int trnotri 51 int intro? /+ pointer to inputs for a transition ..... \*/ /\* pointer to outputs for a transition ..... \*/ 52 int outtrn; 53 int excord? 54 /\* y cordinate of transition ......\*/ int vycordi 55 int trnplot? /\* whether or not transition is to be plotted \*/ 56 /\* length of name associated with transition #/ int trolen; 57 58 )file2 (bounds) +to2; /\* nointer into data structure .....\*/ 59 60

.

1

## Fri Feb 8 05:01:00 1980 transgraph.c Page 2 61 /\*\*\* INTEGERS \*\*\*/ 62 63 int al, a2, a3, a4; /\* global storage for each data structure ... \*/ 64 int buffer[bounds]; /\* buffer into which each frame is read ..... \*/ /\* variable containing # transitions fired .. \*/ 65 int cntr1 [1]; /\* keep track of nodes overflow status ..... \*/ 66 int ctroverflow; /\* a default color for indicating overflows . \*/ 67 int dfltcalor; /\* file descriptor for RUN..Y files ..... \*/ /\* file descriptor for RUN..Z files ..... \*/ 68 int fdfbuf; 69 int fdabuf; /\* storage into which fired places are read . \*/ 70 int firing[bounds]; 71 int ictr; /\* counter passed to a function ..... \*/ 72 int ievents; /\* number of non- & displayable nodes ..... \*/ /\* counter for the interrupt mechanism ..... \*/ 73 int iflag; 74 int iotbl[limit]; /\* forms input-to-output relationship ..... \*/ /\* counter for the iterations of the network. \*/ 75 int kpictures; 76 int linktbl [100] [4]; /\* version 1 & 2 screen nodes locations ..... \*/ 77 int nbrolot; /\* counter for number of displayed nodes .... \*/ 78 int nbytes[2]; /\* store count fields for data structures ... \*/ 79 int overflowtb1(100)(2); /\* data structure to store overflow locations \*/ /\* user selected conrac graphics screen ..... \*/ 80 int set? /\* saves flag for later use by trnlite() .... \*/ 81 int sflag(20); /\* a counter for version 3..reset conditions. \*/ A2 int tblctr; 83 int uniqueto1 (100) (4); /\* reset table locations for version 3 ..... \*/ /\* user selected option ..... \*/ 84 int vers: 85 86 B7 /\*\*\* CHARACTERS \*\*\*/ 88 /\* buffer to store name of second file ..... \*/ 89 char fbuf (2017 90 char gbuf(20); /\* buffer to store name of third file..... \*/ 91 char names[limit]; /\* character array for node labels ..... \*/ /\* option variable for display to the screen. \*/ 92 char scrn: 93 94 95 /\* 96 /\*\* FUNCTION MAIN \*\*/ 97 /\*\*\*\*\* 98 99 main() { 100 101 extern rubout(); /\* declare 'rubout' globally ..... \*/ init(); /\* read input file ..... \*/ 102 /\* verify if user wants to see data structure \*/ determine(); 103 104 display(); /\* display input to crt .....\*/ 105 /\* select version of simulation & denisco set \*/ select(); 106 /\* prepare denesco-conrac ...... \*/ prepare(2); /\* draw network nodes on conrac ...... \*/ 107 ()ahonwarb 108 olaces(); /\* verify correct nooes grawn ...... \*/ 109 /\* function displays network transitions .... \*/ trnslink(); imark(); /\* starting status of network packets ..... \*/ 110 signal(2,rubout); /\* sets 'BRK' as interrupt ...... \*/ 111 /\* successive iterations of network flow .... \*/ 115 marking(); /\* closing out graphics facilities ..... \*/ 113 gnfini(); 114 115 ) 116 117 119 /\*\* PROGRAM FUNCTIONS \*\*/ 120 /\*

120

EL MAYO

. 1

```
Page 3
                           Fri Feb 8 05:01:00 1980
transgraph.c
121
155
123 pause(peroid) (
124
        /* function necessary as sleep() not compatible with signal() */
125
        int i.j.k;
126
127
        printf("+++>>interrupt.....");
        for('i=0;i<persid;i++) (
128
            for(j=0;j<400;j++) {
129
                for(k=0;k<1000;k++) {
130
131
                }
132
            }
133
        •
        printf("***>>wait....");
134
135 return;
136 }
137
138
139 rubout() {
140
        /* function enables the 'brk' key as the interrupt signal */
141
        char halt;
142
143
        space(2);
144
        orintf
        ("***>>> received signal...frame number %d...<ret> to continue \n",
145
146
        (if)#a+2));
147
        orintf
        ("***>>> for termination of program...type 'brk' from console \n");
148
149
        while ((halt=detchar())!='\n') {
150
              /* do=nothing loop */
151
        signal(2,rubout);
152
   returnj
153
154 }
155
156
                                               · -,
157 space(returns) {
158
         int i;
159
         for (i=0; i<returns; i++) {
160
             printf("\n");
161
162
             }
163 return;
164 }
165
160
167 init() (
        /* function opens unformatted file & initializes start condition */
168
169
        int a,bufctr,count,fd,i,j;
170
        char chuf(201,c,f;
171
172
        soace(2);
        printf("+++->TRANSGRAPH ILLUSTRATES PETFINET SIMULATION HODELS");
173
174
        (S) space
        printf("***->ENTER THE NAME OF THE FILE TO HE PROCESSED ... BUT \n");
175
        erint f("
                     NOTE THAT THIS FILE MUST BE AN UNFORMATTED FILE NO");
176
        printf("
                     PRODUCED AS A RESULT OF EXECUTING simulator.out \n");
177
        printf("
                     THE LAST LETTER OF WHICH MUST END IN LETTER 'X' \n");
178
      error:scace(2);
179
        printf("***->");
180
```

. .

1

. . . 121

C. And

```
Page 4
                          Fri Feb 8 05:01:00 1980
transgraph.c
181
182
        i=0;
        while((c=getchar()) != '\n') {
183
184
             cbuf[i]=c;
185
             1++;
186
        3
        cbuf4i)='\0';
187
198
189
        bufctr=i;
190
        for(j=0;j=hufctr;j++) (
191
            abuf[j]=fbuf[j]=chuf[j];
192
            if(cbuf[j]=='X') (
193
                fouf[j]='Y';
                qouf []]='Z';
194
195
                abuf[j+1]=fbuf[j+1]='\0';
196
                j=bufctr;
197
            }
198
       )
199
200
        fd = open(cbuf,0);
        if (fd <= 0) {
201
202
            printf("***=>error occurred in opening file...try again");
203
            space(3);
204
            goto error;
205
206
        207
208
        ievents = (nbytes[1] - 1);
209
        al = nbytes[1];
210
211
        a = {nbytes[1]-1]+14;
        if((count = read(fd,file1,a)) 1= a)
212
            printf("error occured in file1 read");
213
214
215
        if((count=read(fg,nbytes,header))1=header)
216
            printf("error occurred in mbytes read");
217
        a2=nbytes[1];
218
        a=(nbytes(1)+1)+16;
219
        if((count=read(fd,file2,a))!=a)
220
            printf("error occurred in file2 read");
221
225
        if((count=read(fd, nbvtes, header))!=header)
553
            printf("error occurred in header read");
224
        a3=nbytes[1];
225
        a=(nbvtes[1]+1)*2;
        if((count=read(fd,ioth1,a))!=a)
559
227
            printf("error occured in iothl read");
22A
229
        if((count=read(fd;nbvtes;header))1=header)
           printf("error occurred in header read");
230
231
        a4=nnytes[1];
232
        a=nhytes[]]+1;
533
        if((count=read(fd;names;a))1=a)
            printf("error occurred in names read");
234
235
       close(fd);
236
237 returni
238 }
239
240
```

1

1

1, 1 • 1 122



1 Mar 1

```
transgraph.c
                 Page 5
                             Fri Feb 8 05:01:00 1980
241 determine() (
242
         int if
243
         char d,dbuf[20];
244
245
         space(2);
         printf("***->FUNCTION 'DETERMINE' ALLOWS THE USER \n");
printf(" TO EXAMINE ALL PRIMARY DATA STRUCTURES");
246
247
248
        over:space(2);
        printf("###=>IF THIS FEATURE IS DESIRED TYPE 1 IF NOT 0,...<RET>");
249
250
         space(2);
         printf("***=>");
251
252
253
         i=0;
254
         while((d=qetchar())!='\n') {
255
             dbuf[i]=d;
529
             i++;
257
         }
258
         dbuf[i]='\0';
259
590
         i=0;
261
         while(dbuf(i)1=1 \times 0) (
595
            d=dbuf[i];
263
             switch(a) (
264
                 case'0':
265
                       scrn='0';
266
                       break;
267
                 case'1':
268
                       scrn='1';
269
                       printf("***->USE CONTROL Q WHEN SCREEN FULL");
270
                       breaki
271
                 default:
272
                      printf("***=>either blank or invalid entry");
273
                       aoto over;
274
                       break;
275
             }
276
             i++;
277
        )
278 return;
279 }
280
281
} ()velasib 595
283
      int if
284
285
      if(scrn=='1') (
266
           space(2);
287
           bol = filel;
288
           crintf("+++-> FILE1 DATA STRUCTURE");
280
           space(2);
290
           crintf("Infeed
                             .
                                   marker
                                              xcord ycord plot length \n");
291
           for (i=C;i<al; i++){
295
               mmintf("2d Nr %d Nr %d
293
                  hpl->ctrll,nrl->namentr, bol->marker,hol->xcord,
294
                  hel=>ycerd,hol=>olot,bol=>length);
295
               h01++;
296
           ł
297
29A
           space(2);
299
          t5efft = 500
300
          print("+++=> FILE2 DATA STRUCTURE");"
```

,

.

t

```
Fri Feb 8 05:01:00 1980
transgraph.c
                                           Page 6
301
                            space(2);
302
                           printf
303
                            ("Infeed
                                                        troptr intro
                                                                                                     outtrn = xxcord yycord trnplot trnlen \n");
                            for (i=0;i<a2; i++){

    printf("%d \t %d \t %d
304
305
306
                                                 no2->ctrl2,bo2->trnptr,bo2->intrn,bp2->outtrn,
307
                                                :(nefnnt<=Sqd,tofqnnt<=Sqd,bnoovy<=Sqd,bnooxy<=Sqd
308
                                     bp2++;
309
                           }
310
311
                           space(2);
                           printf("***-> IOT9L DATA ARRAY");
312
313
                           space(2);
                           314
315
316
                           ł
317
318
                           space(2);
319
                           printf("***-> NAMES DATA ARRAY");
320
                            scace(2);
321
                            for (i=0; i<a4; i++) { ·
322
                                 printf("%c",names[i]);
323
                           3
324
                           space(2);
325
              }
326 return;
327 1
328
329
330 prepare(type) {
331
                    /* function designates set, screen size and color table */
332
                     int notovi
333
334
                     y=0;
335
                     aenisco (set);
336
                     erase();
337
                     screen(0.0,0.0,511.0,511.0);
338
                     setmod(type);
339
                     coltab();
340
                     colort(11);
341
                      for(n=12;n<14;n++) {-
342
                                color(n);
343
                                 for(t=0;+<512;t++) {
344
                                           seamnt(0,v+t,511,y+t);
345
                               }
340
                    )
347 return;
348 1
349
350
351 drawnorte() {
352
                    /* function displays type & location of network nodes */
353
                     char c,*nptr,hold;
354
                     int ashschrihl, count, deentry, h, i, j, k, i, test, x, y, +z;
355
                     float s;
350
                     double sget();
357
358
                    bp1 = filel;
359
                     A=0;
360
                     count = 1;
```

ì

11

۰,

Fri Feb 8 05:01:00 1980 Page 7 transgraph.c 361 365 while((test=(bol->nameotr))!=0) ( 363 color(14); 364 test++; 365 z = &names[test]; c = +z; 366 307 368 if ((b=(bp1->plot))!= 0) { 369 switch (c){ 370 case 'I': 371 x = (bp1->xcord); 372 y = (Fp1->ycord); 373 for(d=0;d<10;d++) { 374 seamnt(x=16,y+2+d,x,y+2+d); 375 > 376 linktb][a][3]=1; 377 break; 378 case '0': 379 x = (bol=>xcord); 380 y = (hol->ycord); 381 for(d=0;d<10;d++) { 382 segmnt(x=16/y=2=d/x/y=2=d); 383 3 384 linktb][a][3]=1; 385 break; 386 case 'R': 387  $x = (pn^{1} \rightarrow xcord);$ v = (hpl->ycord); 388 389 for(d=0;d<31;d++) { 390 segmnt 391 (x-18+d/4,y+15-d,x+18-d/4,y+15-d); 392 ¥ 393 c]r]b]=14; 394 label(x,y,test,clr)b)); 395 breaki case 'S': 396 397 x = (bp1->xcord); 398 y = (hol->ycord); 399 for(d=0;d<31;d++) { segmnt(x-18,y-15+d,x+18,y-15+d); 400 401 } 402 if(vers==3) { 403 notr= &names[test+1]; rola= +notr; 404 405 ciribl=atoi(shold); 406 tahel(x,v,cest,clr1b1); 407 color(14); 408 1 409 else ( c1+101=14; 410 411 label(x+y+test+clrltl); 412 • 413 break; case 'T': 414 x = (vvl->xcord);415 v = (tol=>ycord); 410 for (x=0;x<19;x++) { 417 418 3=+1 419 i=(x-(sqrt(324.-s+s))); 420 j=v+s;

i

.

....

transgraph.c	Page 8 - Fri Feb 8 05:01:00 1980
421	1=(x+(sart(324s*s)));
422	seamnt(i,j,l,j);
423	}
424	for (k=0;k<19;k++) {
425	S=k;
426	1=(x+(sart(524s±s)));
427 .	}=¥ <b>=\$;</b> }=((32//)));
420	f=(k*(\$Grt(J24\$*3))))
410	
431	if(vers==3) {
432	nptr = \$names[test+1];
433	hold = *nptr;
434	clrlbl=atoi(&hold);
435	label(x,y,test,c1r1b1);
436	color(14);
437	<b>)</b>
438	else (
439	CIPIDI=147
440	ladel(x,y,test,ciridi);
441	
442	default:
445	<pre>orintf("name ont valid identifier");</pre>
445	space(2);
446	breaki
447	)
448	
449 li	nktbl [a] [0] =count1
450 li	nktp(a)(1) = x;
451 I i	nkthlal(2) = y;
452 a+	• •
455 }	
455 bol++:	
456	
457 if(ver	s==3) (
458 en	try=0;
459 un	iquetb1 (entry] [0]=1 inktb1 (0] [0] ;
460 un	iquetb1 [entry] [1] =1 inktb1 [0] [1] ;
461 un	auetb1[entry].[2]=1inktb1[0][2];
462 un	iaueth1 [entry] [3] =1 inkth1 [0] [3] J
465 th	Ctr=1;
464 10	(1=U/1<8/144) { if/uniouath1/antou1/11=t1/akth1/i+11/11 = #
465	uniqueth] [entry] [2] == liokto] [i+1] [2] }
467	/* do potning */
408	
469	else (
470	entry++;
471	uniquetol[entry][0]=linktb1[i+1][0];
472	uniqueth1 [entry] [1]=1inktb1 [i+1] [1];
473	uniquetb) [entry] [2]=]inktb] [i+1] [2] ;
474	uniqueth1 (éntry) [3]=1inkth1 [i+1] [3] ;
475	tbictr++;
4/0	1
477 . } 478 L	
470 7 479 shaalai	• <b>Ta</b> :
480 peturn:	· − ♥ /

A.4.2.4.7.

```
Fri Feb 8 05:01:00 1980
                 Page 9
transgraph.C
481 )
492
4A3
484 places() {
485
        int h,i,j,k;
486
         if(strn=='1') {
487
             space(2);
488
             printf("***=>DATA STRUCTURE LINKTEL ");
489
             snace(2);
490
             for(i=0;i<nbrolot;i++) i
491
                  for(h=0;h<4;h++) {
492
                      printf("%d --",linktbl(i][h]);
493
                  5
494
                  snace(1);
495
496
             3
             soace(2);
497
             if(vers==3) {
498
                  printf("***->DATA STRUCTURE UNIQUETBL");
499
                  space(2);
500
                  for(j=0;j<tblctr;j++) {
501
                      for(k=0;k<4;k++) {
    printf("%d --",uniquetb1(j](k));</pre>
502
503
                      )
504
                       space(1);
505
                  }
506
              >
507
              space(2);
508
509
         }
510 return?
511 )
512
513
 514 label(xx+yv+zz+clbl) (
             /* determines node lahel placement in relation to node */
 515
             int fii;
 510
 517
             char af
 518
             color(c1b1);
 519
             if(xx>250) (
if(yy>250) a='1';
 520
 521
                  else a='2';
 522
 523
                 }
             else (
 524
                  if(yy>250) a=1317
 525
                  -1se 3='4';
 526
527
528
                  $
             f=(bol=>lenoth);
 529
             switch (a) (
 530
                  case'l':
 531
                      for(i=fli<2;i++) {
 532
                          charac((xx+(20+(8+i))),vy+15,names[zz]);
 533
                          zz++;
 534
                      .
 535
 536
537
                      breaki
                  case'?':
                       for(i=0;i<2;i++) {</pre>
 538
                         charac((xx+(20+8+1)),vy=28,names[zz]);
 539
                          22++;
  540
```

.1

ţ

1

ì

.1

ŀ,

127

transgraph.c Page 10 Fri Feb 8 05:01:00 1980 541 • 542 preaki 543 case'3': 544 for(i=0;i<2;i++) { 545 charac((xx=(32=(8+i))),yy+15,nates[zz]); 546 zz++; 547 ¥ . 548 break; 549 case'4': 550 for (i=0;i<2;i++) { 551 charac((xx~(32-8\*i)),yy-26,names[zz]); 552 22++; 553 ) 554 break; 555 } 556 returni 557 } 558 559 560 select() { 561 int i/n/ 562 char vyvbuf[20]; 563 564 space(2); 565 printf.("\*\* \*->THERE ARE 3 VERSIONS TO THIS GRAPHICS PACKAGE \n\*); PLEASE SELECT ONE OF THE FOLLOWING VERSIONS: \n"); 566 orintf(" 567 again:space(2); VERSION 1 ... PETRI-NET PACKAGE ...... TYPE 1 \n"); VERSION 2 ... PACKET PEPRESENTATION ... TYPE 2 \n"); VERSION 3 ... MULTIROUTING PACKAGE .... TYPE 3 \n"); 568 printf(" printf(" 569 570 printf(" 571 n = 0; 572 twice:spac=(2); 573 printf("+++->"); 574 n++; 575 i = 0;576 while ((v=getchar()) != '\n') { 577 viuf(i) = vi578 1++7 579 ) 580 vbuf(i) = 101;581 582 i = 0; 583 while(vouf(i) 1= 'NO') ( 584 v = vouf[i];585 if(n==1) { switch(v) { case '1'; 586 587 588 vers = 17549 break; 500 case '2': 591 vers = 2; 592 breakt 593 case '3': 594 vers = 37 595 breaki 596 597 default: orintf 598 ("\*\*\*->incorrect version try again!"); 599 goto again; 600 breakt

4

ť

1.5

· i

and the second sec

Page 11 Fri Feb 8 05:01:00 1980 transgraph.c 601 602 i++; 603 } 604 else ( switch(v) { case '0'; 605 606 set = 0; 607 608 breaki case '1': 609 set = 1; 610 breaki 611 case '2': 612 set = 2; 613 614 break; 615 default: 616 orintf ("\*\*\*->incorrect genisco set try again!"); 617 618 orintf 619 set selection should be 0,1 or 2"); (\* n = 1; 650 621 goto twice; 622 break; } 653 624 1++7 **)** 1 625 959 • 627 if(n==1) ( space(2); 628 ->NOM SELECT THE GENISCO SET YOU WISH \n\*); THE PROGRAM TO BE DISPLAYED TO..... \n\*); IN C3 LAB EITHER SET0, SET1 OR SET2 \n\*); 629 orintf("\*\* 630 printf(" crintf(" 631 635 goto twice; 633 ) 634 return; 635 } 636 637 638 imark() { 639 /\* marks initial state of system by calling appropriate function \*/ 640 int bicolourieiaixiy; 641 bol = filel; 642 643 dfltcolor=3; 644 color(dfltcolor); 645 colour=2; 646 ctroverflow=0; 647 while ((c=(bn1=>nameptr)) 1=0) { 648 if((b=(bo1->plot)) !=0) { 649 switch(vers) - { · 650 case 1: 651 ivers1(); 652 breaki 653 case 2: 654 ivers2(colour); 655 break} 656 case 3: 657 ivers3(colour); 658 breaki 659 ) 660 ł

129

ł

. 1

Ì

```
Fri Feb 8 05:01:00 1980
transgraph.c
                 Page 12
661
             if(dfltcolor!=3) color(3);
662
             bo1++;
663
         3
664
         color(14);
         printa(0,350.,480.,"TIME FRAME = 1");
665
        preread(1);
666
667
         trnlite(1);
668
         displa();
669
        hold();
670
         trnlite(2);
671
         reset(1):
672
         color(13);
673
        ovrflow();
674
        printg(0,350.,480.,"TIME FRAME = 1");
675 return;
676 }
677
678
679 hold() {
680
         int holding;
681
682
        space(2);
        printf("***->THIS IS THE INITIAL STATE OF THE NETWORK \n");
printf(" TYPE <RETURN> TO CONTINUE EXECUTION..... \n");
683
684
685
        while((holding=getchar())1='\n') {
686
            /* do nothing loop */
687
        3
688 return;
689 }
690
691
692 ivers1() {
693
        int exxxyzi
694
        char check;
695
696
        e=(bp1+>marker);
697
        x=(hp1->xcord);
698
        v=(bol~>vcord);
699
         z=(bpl->nameptr);
         if((check=names[z+1])!='I' && (check=names[z+1])!='0') {
700
701
              printa(0,x=3.0,511.=(y=3),"%d",e);
702
        3
703
         else {
704
              if(check=names[z+1]=='I') printa(0,x=14.0,511.=(y+2),"%d",e);
705
              else printa(0,x=14.0,511.=(y=9),"%d",e);
706
        )
707 return;
708 }
709
710
711 ivers2(colour) {
712
        int exxivizi
713
        char check;
714
715
        e=(hol->marker);
716
        x=(tol=>xcord);
717
        v=(bo1->vcord);
         z=(bp1=>namentr);
718
         if((check=names[z+1])!='I' && (check=names[z+1])!='0') {
719
720
             pckt2(x,v,e,colour);
```

130

. .

-1

```
Fri Feb 8 05:01:00 1980
                 Page 13
transgraph.c
721
         3
722
         else (
              if(check=names{z+1}=='I') printq(0,x-14.0,511.-(y+2),"%d",e);
else printq(0,x-14.0,511.-(y-9),"%d",e);
723
724
725
        }
726 return;
727 }
             ••
728
729
730 ivers3(colour) (
731
        char keed, *kptr;
732
         int araarbrbbrcrccrclr[25], irk, nrstack, total, xry;
733
734
        total=0;
735
        n=1;
736
        x=(bp1->xcord);
737
        y=(bpl->ycord);
         a=(bpl=>length);
738
739
        b=(col->nameotr);
740
        c=(bo1->marker);
741
        total=total+c;
742
        if(c>0) {
743
744
             far(i=0;i<c;i++) {
                kotr = %names[b+a=2];
keep = *kptr;
745
746
747
                 clr[n]=atci(&keep);
748
                 n++;
749
             }
750
        }
751
752
        kotr = &names(b+a);
753
        keep = *kptr;
754
         stack=atoi(&keep);
755
756
        if(names(b+(a=1))!='0') (
             it(names[c+(a=1)]==+2*)
757
                                        stack=stack+20;
             else stack = stack + 10;
758
759
        }
760
        for(i=n;i<stack=1;i+t) {
761
762
             001++;
763
             aa=(no1=>length);
764
             bb=(bcl=>nameotr);
             cc=(to1=>marker);
765
766
             total=total+cc;
767
             if(cc>0) (
                 for (k=01k<cc1k++) (
768
769
                     kntr = knames(bb+aa+2);
                     keno = +kotri
770
771
                     clr[n]=atoi(&keep);
172
                     n++;
773
                 }
774
             }
775
        •
        if(names(b+1]1='1' %% names(b+1)1='0') {
776
777
          nckt3
778
           (x,v,total,clr[1],clr[2],clr[3],clr[4],clr[5),clr[6],clr[7],
779
            colour);
780
        )
```

:

ł

i

. 1

. 1

131

32.4

```
transgraph.c
                Page 14
                            Fri Feb 8 05:01:00 1980
781
        else (
782-
             if(names(h+1)=='1') printg(0,x-14.0,511.-(y+2),"%d",total);
             else printg(0,x=14.0,511.=(y=9),"%d",total);
783
784
        }
                                                         785 return;
786 }
787
788
789 preread(flag) {
-7.90
         int bucket[2],count,fd,fa,i,nbrtrns;
         if(flag1=3) {
791
792
         if(flag==1)
                      fd = open (fbuf, 0);
793
794
             if (fd<=0) {
                 printf("***=>error occurred in opening fd file");
795
796
             }
797
             fdfbuf=fd;
798
             fa = open (abuf, 0);
799
             if (fo<=0) {
800
                 printf("***->error occurred in opening fg file");
801
             3
802
             fdgcuf=fa;
803
        if((count=read (fdfhuf, bucket, 2))1=2) {
    printf("***->error occurred in fd bucket read");

804
805
806
807
        if((count=read (fdfbuf, buffer,(ievents*2)))!=(ievents*2)) {
808
            printf("***=>error occurred in buffer read");
809
810
        if((count=read (fdupuf, bucket, 2))1=2) {
            printf("***=>error occurred in fq bucket read");
811
812
513
        if((count=read (fdqbuf, cntr1,2))1=2) {
814
            printf("***=>error occurred in cntrl read");
815
        }
        if(cntr1(0)==0) {
816
817
            space(2);
819
            printf("***=>the last network state has been achieved");
819
            koictures=pictures+1;
820
            soace(2);
821
        ¥
822
        else {
823
            if((count=read (fdqbuf, bucket, 2))!=2) {
824
                   orintf("***=>error occurred in tucket read");
825
826
            nurtros = cotr1[0]+2;
            if((count=read (fdonuf, firing,nhrtrns))!= nbrtrns) {
827
858
                   orintf("***=>error occurred in firing read");
829
            }
830
        }
831
         }
635
         else {
        close(fdfbuf);
833
        close(fdaout);
834
835
         •
836
    returni
837
    838
839
840 stage() (
```

. 4

```
Fri Feb 8 05:01:00 1980
transgraph.c
                Page 15
841
        if (kpictures:=0 && kpictures<pictures+1) (
842
            (S)bearand
843
            if(knictures1=pictures+1) {
844
                 trnlite(1);
845
                 disola();
846
                pause(1);
847
848
                 trnlite(2);
849
            )
850
        •
851 return;
852 }
853
854
855 marking() {
        /* function displays successive iterations of the network */
856
        int colour, draw, i, mark, n, x, y;
857
858
        bol = filel;
859
        n = 2;
860
861
         /* following loop processes ievent # data entries each pass */
802
         for(kpictures=0;kpictures<pictures;kpictures++) {
863
             stage();
864
             if(koictures>0) {
865
                 reset();
866
                 color(13);
867
868
                 ovrflow();
                 prints(0,350.,480., "TIME FRAME = 2d", n);
869
                 n++;
870
871
             }
872
             iflag = kpictures;
873
             draw = (bol->plot);
             dfltcolor=3;
874
             (folosilit) rolos
875
             $5=ruoloo
876
877
             ctroverflow=0;
             for (i=0; i < ievents; i++) {
878
                 if (draw == 1) {
879
                      ictr = it
840
                      switch (vers) (
881
882
                          case 1:
                              vers1();
883
                              breaki
AA4
                          case 2:
885
                              vers2(colour);
886
897
                              nreaki
888
                          case 3:
                              vers3(colour);
889
                              breaki
890
 891
                      }
892
                  )
893
                  i = ictr++;
                  if(d+ltcolor1=3) cclor(3);
894
                 b01++;
 A95
                  draw = (bp1 -> plot);
 896
 897
             )
 898
             dfltcolor=14;
             color(dfltcolor);
 899
 900
             colour=13;
```

ł

1

7

.

1

133

- -

-----

```
transgraph.c
                Page 16
                            Fri Feb 8 05:01:00 1980
            printg(0,350.,480.,"TIME FRAME = %d",n);
901
902
            pause(2);
903
            trolite(3);
904
            bpl = filel;
905
        >
906
        preread(3);
907 return;
908 }
909
910
911 ovrflow() (
912
913
        int i/x/yi
914
915
        i=0;
916
        while(overflowtb1[i][0]!=0) {
917
            x=overflowtb1[i][0];
918
            y=overflowtt1[i][1];
919
            nlock((x-3)+1.,511,-(y+28)*1.,(x+10)*1.,511.-(y+20)*1.);
            overflowtol[i][0]=0;
920
921
            overfloatb1[i][1]=0;
922
            i++;
923
       }
924 return;
925)
926
927
928 vers1() {
929
        int exxxyzzi
930
        char check;
931
932
        color(3);
933
        e=bufferlictr];
934
        x=(bo1->xcord);
935
        v=(bol->ycord);
936
        z=(bp1->nameptr);
        if((check=names[z+1])!='I' && (check=names[z+1])!='0') {
937
938
             printg(0,x-3.0,511.-(y-3),"%d",e);
939
        3
940
        else (
941
             if(check=names[z+1]=='J') printq(0,x-14.0,511.-(y+2),"Xd",e);
             else prints(0,x-14.0,511.-(y-9),"%d",e);
942
943
        3
944 returni
945 }
046
947
948 vers2(colour) (
949
        int markixiyiz;
950
        char check;
951
952
        x=(hol->xcord);
953
        y=(to1->ycord);
954
        z=(col=>nareptr);
955
        mark=buffer[ictr];
        if((check=names[z+1])!='1' %% (check=names[z+1])!='0') {
956
957
         ockt2(x,y,mark,colour);
958
        }
959
        else {
         if((check=names[z+1])=='I') printq(0,x-14.0,511.-(v+2),"%d",mark);
960
```

ŝ

14

1

```
Fri Feb 8 05:01:00 1980
                Page 17
transgraph.c
961
         else printg(0,x-14.0,511.-(y-9),"%d",mark);
962
        }
963 return;
964 }
965
966
967
   vers3(cólour) (
968
        int a.aa.b.bb.mark,marks.clr[25],j.k.n.stack.total.x.y;
969
        char keep, *kotrj
970
971
        total=0;
972
        n=1;
973
        x=(bp1->xcord);
974
        y=(bo1->vcord);
975
        a=(bpl->lenath);
976
        b=(bp1->nameptr);
977
        mark=buffer[ictr];
978
        total=total+mark;
979
        if(mark>0) {
980
            for(j=0;j<mark;j++) {
                kotr = &names[b+a-2];
981
982
                xeeo = *kotri
983
                clr[n]=atoi(&keep);
984
                n++;
985
           `}
986
        )
987
988
        kptr = &names(b+a);
980
        keeo = *kotri
990
        stack=atoi(&keep);
991
992
        if(names(t+(a-1))!='0') {
            if(names[b+(a-])]=='2')
                                      stack=stack+20;
993
994
            else stack = stack + 10;
995
        )
996
997
        for(j=0;j<stack-1;j++) {
998
            bp1++;
999
            ictr++;
1000
             aa=(up1=>length);
1001
             bb=(bol=>nameotr);
             marks=buffer[ictr];
1002
1003
             total=total+marks;
1004
             if(marks>0) {
1005
                 for(k=0;k<marks;k++) {
1006
                     kptr = snames[bb+aa+2];
1007
                     keep = skptrj
1004
                     clr(nl=atoi(%keer);
1009
                     n++;
1010
                 }
1011
             )
1012
         3
         if(names[b+1]:='I' $% names[b+1]:='û') (
1013
1014
           pckt3
1015
           (x,y,total,cir(1),cir(2),cir(3),cir(4),cir(5),cir(6),cir(7),
1016
            colour);
1017
         •
1018
         else
1019
             if(names(h+11=='1') printg(0,x-14.0,511.-(y+2),"%d",total);
             else printg(0,x-14.0,511.-(v-9),"%d",total);
1020
```

×,

. 1,

1.5

135

a charles and the

tran	sgraph.c	Page 18	Fri feb	8 05:01:00 1980
	,			
1021	· · · · · · ·			
1022	returni			
1023	,			
1024				
1025				
1020	OCKIELXAXI	5/ Yax 1 3/ PO1	11/21853/	
1027	au i sabitaa			
1020	Switchibo	1007 1		
1030	beeakt			
1031	case 1:			
1032	block(	(xaxis=3)*1		<pre>varie+3)*1(xarie+3)*1511.=(varie=3)*1.);</pre>
1033	breaki			
1034	case 2:			
1035	block(	(xaxis=3)*1	.,511(v	/axis+3)*1.,(xaxis+3)*1.,511(vaxis=3)*1.);
1036	hlock(	(xaxis=3)*1	.,511(y	/axis=7)*1.,(xaxis+3)*1.,511.=(yaxis=13)*1.);
1037	breaki			
1038	case 3:			
1039	block(	(xaxis=3)*1	.,511(y	/axis+3)*1.,(xaxis+3)*1.,511.=(yaxis=3)*1.);
1040	black(	(xaxis=3)*1	.,511(y	/axis=7)*1.,(xaxis+3)*1.,511.=(yaxis=13)*1.);
1041	block(	(xaxis=3)*1	.,511(y	/axis+13)*1.,(xaxis+3)*1.,511(vaxis+7)*1.);
1042	break;			
1043	case 4:			· · · · · · · · · · · · · · · ·
1044	block(	(xəxis=3)*1	••511(y	/axis+3)*1.,(xaxis+3)*1.,511(yaxis=3)*1.);
1045	block(	(xaxis=5)*1	••511••(9	/axis=7)*1.,(xaxis+5)*1.,>11.=(yaxis=15)*1.);
1046	block(	(xaxis=5)+1	••511(y	/axis+15)*1.,(xaxis+5)*1.,511.=(vaxis+/)*1.);
1047	DIOCK(	(xax15+/)+1	••511(y	/ax1s+5)*l.,(xax1s+15)*1.,511(yax1s-5)*1.);
1040	oreaki			
1047	case Di	(	511 -64	(nuine 2) al (unuine 2) al (511 m(unuine 2) al ):
1050	DIOCKI	(xaxis=3)=1 (~~~i==3)+1	.511 -(v	/dx13+3/=(
1051	block	(x7x15-3)=1 (unuinal)=1	.511 -(y	/dx13=/)=1+/(xdx13+3)=1+/311+=(ydx13=13)=1+//
1053	block(	(**************************************		/3X/3*/3/*/*/(XAK/3*3/*/*//3/10=(/8**3*//*/*// /aujes3)61 .(vavjes13)61511.6(vavjes3)61.)1
1054	ploce(	(xavia=13)*	15110	(vavie+3)*1(vavie+7)*1511(vavie=3)*1)
1055	break:			
1056	case 61			
1057	block(	(xaxis=3)+1		<pre>/axis+3)*1.,(xaxis+3)*1.,511(vaxis-3)*1.);</pre>
1058	black(	(xaxis=3)*1	.,511(v	<pre>/axis=7)*1.,(xaxis+3)*1.,511.=(yaxis=13)*1.);</pre>
1059	block(	(xaxis=3)*1	.,511(y	<pre>/aris+13)*1.,(xaxis+3)*1.,511.=(yaxis+7)*1.);</pre>
1060	block(	(xaxis+7)+1	.,511(y	<pre>/axis+3)*1.,(xaxis+13)*1.,511(yaxis-3)*1.);</pre>
1061	bloc«(	(xaxis=13)*	1.,5110	<pre>varis+3)+1.,(xaris=7)*1.,511.=(yaris=3)*1.);</pre>
1062	block(	(xaxis=13)*	1.,511(-	(yaxis+13)±1.,(xaxis-7)*1.,511(yaxis+7)*1.);
1063	breakt			
1064	case 7:			
1065	block(	(xaxis=3)*1	.,511(+	<pre>/axis+3)*1.,(xaxis+3)*1.,511.=(yaxis=3)*1.);</pre>
1000	nleski	(xaxis=3)+1	511(y	<pre>(avis=7)*1.,(xaxis+5)*1.,511.=(vaxis=13)*1.);</pre>
1067	hlock(	(xaxis=3)=1	.,511(v	<pre>(axis+13)+1., (xaxis+3)+1.,511.=(vaxis+7)+1.);</pre>
1068	D)00*(	(xaxis+7)+1	••511•=(y	(axis+3)+l., (xaxis+13)+l.,511.=(vaxis=3)+l.);
1069	block(	(xaxis=15)*	1.,511.=(	yaxis+5)+1.,(xaxis=7)+1.,511.=(yaxis=5)+1.);
1070	block	(*8*15=15)*	1.,511()	.yaxis+15)+1.,(xaxis=/]*1.,511.=(yax1\$*/]*1.);
1071	O LOCK (	(xax15+/]*]	••511•*(y	'axistlo/=[+/(xaxistlo/=l+/011+=tyaxist/)*l+};
10/2	nreaki			
1075	netault	; 		11
1075	0V=F11	9W18116789V owtolfetoou	- "TIONJ(U.	'] - 1071 3/ (1744 4 4 2
1075	CT COUCH	owiolicifyv rflawii:	TITOWILL.	
1077	nlockf	(xaxie=3)+1		/axis+3)+1.,(xaxis+3)+1.,511.=(vaxis=3)+1.11
1078	block	(xaxis=3)*1		/axis=7)+1.,(xaxis+3)+1.,511.=(vaxis=13)+1.);
1079	blockf	(xaxis=3)+1	.,511(v	axis+13)+1., (xaxis+3)+1.,511.=(vaxis+7)+1.);
1080	block	(	.511.+(4	autostial (unuiositial SII afunuiontial );

Fri Feb 8 05:01:00 1980 transgraph.c Page 19 1081 block((xaxis=13)\*1.,511.-(yaxis+3)\*1.,(xaxis=7)\*1.,511.-(yaxis=3)\*1.); block((xaxis=13)\*1.,511.-(yaxis+13)\*1.,(xaxis=7)\*1.,511.-(vaxis+7)\*1.); 1082 1083 block((xaxis+7)\*1.,511.-(yaxis+13)\*1.,(xaxis+13)\*1.,511.-(yaxis+7)\*1.); 1084 color(class); dfltcolor=class; 1085 1086 printa(0, xaxis=3.0, 511.=(yaxis+22)+3.0, "%d", point=7); 1087 preak; 1088 • 1089 return? 1090 } 1091 1092 1093 1094 pckt3(xaxis, yaxis, tota1, c1, c2, c3, c4, c5, c6, c7, c1ass) 1095 1096 switch (total) - { 1097 case 0: 1098 break 1099 case 1: 1100 color(c1); block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); 1101 1102 **Dreak**: 1103 case 2: 1104 color(c1); 1105 block((xaxis-3)+1.,511.-(yaxis+3)+1.,(xaxis+3)+1.,511.-(yaxis-3)+1.); 1106 color(c2); plock((xsxis=3)\*1.,511.-(yaxis=7)\*1.,(xaxis+3)\*1.,511.-(yaxis=13)\*1.); 1107 1108 break; 1109 case 3: 1110 color(cl); block((xaxis=3)\*1.,511.=(yaxis+3)\*1.,(xaxis+3)\*1.,511.=(yaxis=3)\*1.); 1111 1112 color(c2); block((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1113 1114 color(c3); block((xaxis-3)\*1.,511.-(yaxis+13)\*1.,(xaxis+3)\*1.,511.-(yaxis+7)\*1.); 1115 1116 break; 1117 case 4: 1118 color(cl); block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); 1119 1120 ((Ss) notos plock((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1151 1155 color(c3); plock((xaxis-3)+1.,511.-(yaxis+13)+1.,(xaxis+3)+1.,511.-(yaxis+7)+1.); 1123 1124 color(c4); tlock((xaxis+7)\*1.,511.-(yaxis+3)\*1.,(xaxis+13)\*1.,511.-(yaxis-3)\*1.); 1125 1159 break; case 5: · 1127 1128 color(cl); 1150 nlocx((xaxis=3)\*1.,511.=(yaxis+3)\*1.,(xaxis+3)\*1.,511.=(yaxis=3)\*1.); 1130 color(c2); 1131 plock((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1132 color(c3); 1133 rlock((varis=3)\*1.,511.-(varis+13)\*1.,(varis+3)\*1.,511.-(varis+7)\*1.); 1134 color(c4); 1135 tlock((xaxis+7)+1.,511.-(vaxis+3)+1.,(xaxis+13)+1.,511.-(vaxis-3)+1.); 1136 color(c5); 510ck((xaxis=13)\*1.,511.-(vaxis+3)\*1.,(xaxis=7)\*1.,511.-(vaxis=3)\*1.); 1137 1138 breaki 1139 Case 6: 1140 color(cl);

Ż

۰J

.1

1

'+ (

137

## block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); 1141 1142 color(c2); block((xaxis=3)\*1.,511.=(yaxis=7)\*1.,(xaxis=3)\*1.,511.=(yaxis=13)\*1.); 1143 1144 color(c3); 1145 block((xaxis-3)+1.,511.-(yaxis+13)+1.,(xaxis+3)+1.,511.-(vaxis+7)+1.); 1146 color(c4); block((xaxis+7)\*1.,511.-(yaxis+3)\*1.,(xaxis+13)\*1.,511.-(yaxis-3)\*1.); 1147 1148 color(c5); 1149 block((xaxis=13)\*1.,511.=(yaxis+3)\*1.,(xaxis=7)\*1.,511.=(yaxis=3)\*1.); 1150 color(c6); block((xaxis-13)\*1.,511.-(yaxis+13)\*1.,(xaxis-7)\*1.,511.-(yaxis+7)\*1.); 1151 1152 breakl 1153 case 7: 1154 color(c1); block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); 1155 1150 color(c2); block((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1157 1158 color(c3); block((xaxis=3)\*1.,511.-(yaxis+13)\*1.,(xaxis+3)\*1.,511.-(yaxis+7)\*1.); 1159 1160 color(c4); 1161 block((xaxis+7)\*1.,511.-(yaxis+3)\*1.,(xaxis+13)\*1.,511.-(yaxis-3)\*1.); 1162 color(c5); plock((xaxis-13)\*1.,511.-(yaxis+3)\*1.,(xaxis-7)\*1.,511.-(yaxis-3)\*1.); 1163 1164 color(c6); plock((xaxis=13)+1.,511.-(yaxis+13)+1.,(xaxis=7)+1.,511.-(yaxis+7)+1.); 1105 1166 color(c7); block((xaxis+7)\*1.,511.-(yaxis+13)\*1.,(xaxis+13)\*1.,511.-(yaxis+7)\*1.); 1167 1168 break; 1169 default: 1170 overflowtbl (ctroverflow) [0] =xaxis; 1171 overflowth1[ctroverflow][1]=yaxis; 1172 ctroverflow++; 1173 color(cl); hlock((xaxis=3)+1.,511.=(vaxis+3)+1.,(xaxis+3)+1.,511.=(vaxis=3)+1.); 1174 1175 color(c2); block((xaxis-3)+1.,511.-(yaxis-7)+1.,(xaxis+3)+1.,511.-(yaxis-13)+1.); 1176 1177 color(c3); hlock((xaxis=3)\*1.,511.-(yaxis+13)\*1.,(xaxis+3)\*1.,511.-(yaxis+7)\*1.); 1178 1179 color(c4); 1180 block((xaxis+7)\*1.,511.-(vaxis+3)\*1.,(xaxis+13)\*1.,511.-(yaxis=3)\*1.); 1181 color(c5); block((xaxis-13)+1.,511.-(yaxis+3)\*1.,(xaxis-7)\*1.,511.-(yaxis-3)\*1.); 1185 1183 color(c6); 1184 block((xaxis-13)\*1.,5]1.-(vaxis+13)\*1.,(xaxis-7)\*1.,511.-(yaxis+7)\*1.); 1185 color(c7); 1186 block((xaxis+7)+1.,511.-(yaxis+13)+1.,(xaxis+13)+1.,511.-(yaxis+7)+1.); 1187 color(class); 1184 dfltcolor=class; 1189 printg(0, xaxis=3.0,511.=(yaxis+22)+3.0, "2d", tota1=7); 1190 breaki 1191 ÷ 1192 returna 1193 } 1194 1195 1190 reset() ( 1147 /\* reset function for successive network iterations \*/ 1198 int i,mark,x,y,z; 1199 if(vers==1 ;; vers==?) { 1200

Fri Feb 8 05:01:00 1980 Page 20 transgraph.c

i .

1

. .]

1

. 4

. 1

138

```
Fri Feb 8 05:01:00 1980
transgraph.c
                 Page 21
1201
                for(i=0;i<nbrplot;i++) {</pre>
                 color(14);
1505
1203
                 x=linktbl[i][1];
1204
                 y=linktbl[i][2];
1205
                 z=linktbl[i][3];
1206
                 if(z==0) {
1207
                 block((x-3)*1.,511.-(y-7)*1.,(x+3)*1.,511.-(y-13)*1.);
1208
                 block((x-13)*1.,511.-(y+13)*1.,(x+13)*1.,511.-(y-3)*1.);
1209
                 }
1210
                 else ·
1211
                 block((x-16) ± 1., 511.-(y-2) ± 1., x ± 1., 511.-(y-10) ± 1.);
                 block((x-16)*1.,511.-(y+10)*1.,x*1.,511.-(y+2)*1.);
1515
1213
                 >
1214
                }
1215
         }
1216
         else {
1217
                for(i=0;i<tblctr-1;i++) (
1518
                 color(14);
1219
                 x=uniquetb1{i1[1];
1220
                 v=unipuetb1(i)[2];
1221
                 z=uniquetb1(i)[3];
1222
                 if(z==0) {
1223
                 block((x-3)+1.,511.-(y-7)+1.,(x+3)+1.,511.-(y-13)+1.);
1224
                 block((x-13)*1.,511.-(y+13)*1.,(x+13)*1.,511.-(y-3)*1.);
1225
1226
                 else {
1227
                     block((x-16)*1.,511.+(y-2)*1.,x*1.,511.-(y-10)*1.);
1228
                     block((x-16)*1.,511.-(y+10)*1.,x*1.,511.-(y+2)*1.);
1229
                 }
1230
                }
1231
         •
1232 return;
1233 )
1234
1235
1236 trnlite(tflag) (
1237
           int flag, h, i, inp, j, k, l, m, n, on, outp, ptr, r, s, x, xx, v, yy)
1238
1239
           for(h=0;h<cntr1[0];h++) {
         to2=file2;
1240
1241
          for(j=0;j<(firina(h)-1);j++) {
1242
              hp2++;
1243
1244
          if(on=(bo2->trnolot)==1) {
1245
              x=(bp2=>xxcord);
1240
              y=(bo2+>yvcord);
              if(tflag==1 :: tflag==2) color(11);
else color(15);
1247
1245
1249
              seamnt(x,y=20,x,v+20);
1250
              l=(ho2->trnlen);
1251
              k=(bo2->troptr);
1252
              for(i=0;i<1;i++) {
1253
                  chanac((x-R)(8+i)),y+24,mames(k+1));
1254
                  ĸ++;
1255
              )
1256
1257
               if(tflag==1 || tflag==3) {
1258
              if(tflag==1) color(11);
1259
              else color(14);
1260
              f1ag=0;
```

. 11
Fri Feb 8 05:01:00 1980 Page 22 transgraph.c 1261 m=(bo2->inten); r=iotbl[m]; 1262 for(i=0;i<r;i++) { 1263 1264 ino=iotbl[m+1]; 1265 for(j=0;j<norplot;j++) ( if(inp==linktb1[j1[0]) { 1266 1267 xx=linktbl[j][1]; 1268 vy=linktb1(j)(2); if(flag==0) ( 1269 1270 if(xx<=x) flag=1; 1271 else flag=27 1272 sflag[h]=flag; 1273 3 1274 if(flag==1) intrns1(xx,yy,x,y); 1275 else intrns2(xx,yy,x,y); 1276 j=nbrplot; 1277 > 1278 } 1279 m++; 1280 ) 1281 ) 1282 1283 if(tflag==2 || tflag==3) { 1284 if(tflag==2) color(11); else color(14); 1285 1286 n=(bp2+>outtrn); 1287 s=iotbl[n]; 1288 for(i=0;i<s;i++) { 1289 outo=iotol [n+1]; 1290 far(j=0;j<nhrplot;j++) { 1291 if(outo==linktbl[j]{0]) { 1292 xx=1inkth1[j][1]; 1293 yv=linkth1[j][2]; 1294 if(sflag(h)==1) outrosl(xx,yy,x,y); 1295 else outrns2(xx,yy,x,y); 1296 j=nbrplot; 1297 ) 1298 ł n++; 1299 1300 } 1301 • 1302 ł 1303 } 1304 return? 1305 } 1306 1307 1308 trnslink() { 1309 int flag, i, ind, j, k, l, m, n, on, outp, ptr, r, s, x, xx, v, yv; 1310 1311 ho2=file2; while(str=(hn2=>trnstr)1=0) { 1312 if(on=(ho2=>trnolor)==1) ( 1313 1314 x=(bo2=>xxcord); 1315 v=(to2->vvcord); color(15); 1316 seamat(x,y=20,x,y+20); 1317 1318 l=(no2->trnlen); 1319 k=(bo2->troptr); for(i=0;i<1;i++) { 1320

1

¥

140

CARLOITER



ì

; 1

. 1

141

trans	igraph.c	Page	24	Fri	Feb	8 05:01:00 1980
1381	e	lse in	out=37			
1382	}					
1383						
1384	switc	h (inp	ut) (			
1385	c	ase 0:				• •
1386		seq	nnt(xx:	+15+	¥¥≠x=1.	,y);
1307	٠	r an	tarrow.	(x-1)	, Y ] i	
1300	-	are 11	847			
1390	-	sea	nnt (xx	+12,-	vv+12,	x-12,y-8);
1391		seg	nnt (x=	12.4	-8,x-1	,y-8);
1392		rah	tarrow	(x=1)	,y-8);	
1393		öre	akj			
1394	c	ase 2:				
1395		segi	<b>mntlxx</b>	* 1 2 # 1	yy=12; +8.4-1	X=12/V+7JJ 
1390		seu:	771(11-	( 1 )		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1398		bre	ak]	• •		
1399	c	ase 3:	-			
1400		sear	nnt (xx <sup>.</sup>	-12,	vv=12,:	x-12,y+36);
1401		segr	nnt (x-)	12.4	+16,x=	1,y+16);
1402		lft	seri(x	-12,	v+26);	_
1405		rghi	tarrow	(x-1,	, y + 16 ) i	j -
1404		nrea See /i•	3 4 /			
1406		503E		•12.	vv+12.	(-12, v-36);
1407		Sear	nnt (x=)	12.9	-16,x-	1,y-16);
1408		lft	semi(x'	-12,	(65 <del>+</del>	
1409		rghi	tarrow	(x-1)	,y-16);	<b>;</b>
1410		Dre	aki			
1411	}					
1412	returni					
1414	,					
1415						
1416	rghtarrow	(x,v)	(			
1417						
1418	Segmin	t(x=4,	/-4, x,	y);		
1419	Segmo	tlx=4,	v+4,x,	y];		
1420	recurni -					
1422	,			•		
1423						
1424	lftsemilx	,v) (				
1425	int i	, j , k ;				
1420	doubl	e sart	();			
1427	totir	=0:-<1	1 2 6 + + 1 -	(		
1429	i (), (1	=(v=(q)	ant (19)	)k:	**)));	
1430	j	= ( + ( s	art (10)	)k	**)));	
1431	d	ot ( = (	(2.),	511.4	-i);	
1432	1	ot (x=()	k/2.),	511.	-j);	
1433	i	f(x>8)				•••
1454		001	(*=(k/) f==f=/	2 + 1 + 1 2 - 1 - 1	511 <b>-</b> 1'	- <i>cji</i> •1):
1437		100	(/	2,1.9	511.+1	+1);
1437		dot	(x={k/	, , , , , , , , , , , , , , , , , , ,	5111	+2);
1438		cot	(x=(k/	2.),	511j·	-2);
1439		101	(x-(x/i	5.) *	511 <b></b> j	-1);
1440		dot	(x=(k/i	2.),5	511 <b></b> j	+1);

Page 25 Fri Feb 8 05:01:00 1980 transgraph.c 1441 dot(x-(k/2.),511.-j+2); 1442 ) -1443 ) 1444 returni 1445 } 1446 1447 1448 outrosi(xx,yy,x,y) { 1449 int output; 1450 1451  $if(xx>(x+12)) {$ if(yy<=y) { 1452 if(yy==y) output=0; 1453 1454 else output=1; 1455 ł 1456 else output=2; 1457 ł 1458 else ( 1459 if(yy<=y) output=4; 1460 else output=3; 1461 ł 1462 1463 switch (output) { 1464 case 0: 1465 segmnt(x+1,y,xx=20,yy); rohtarrow(xx=20,yy); 1466 1467 break; 1468 case 1: 1469 segmnt(x+12,y-8,xx-28,yy+8); 1470 seamnt(xx=28, yy+8, xx=20, yy+8); rantarrow(xx=20,yy+8); 1471 1472 segmnt(x+1,y-8,x+12,y-8); 1473 breaki 1474 case 2: 1475 segmnt(x+12,y+8,xx-28,yy-8); 1476 sepmot(xx=28,yy=8,xx=20,yy=8); 1477 rghtarrow(xx=20,yy=8); 1478 seamnt(x+1,y+8,x+12,y+8); 1479 break; 1480 case 3: 1481 segmnt(x+1,y+16,x+12,y+16); 1482 seamnt(x+12,y+36,xx+28,yy=8); segmnt(xx+28,yy=8,xx+20,yy=8); 1483 1484 lftarrow(xx+20,yy=A); 1445 rahtsemi(x+12,y+26); 1486 breaki 1487 case 4: 1488 segmnt(x+1,y=16,x+12,y=16); segmnt(x+12,y-30,xx+20,yy+8); segmnt(xx+20,yy+8,xx+20,yy+8); 1489 1490 1491 lftarrow(xx+20+yv+8); 1492 rahtsemi(x+12,y-26);1493 break; 1494 ) 1495 return? 1496 ) 1497 1498 1499 rghtsemi(x<sub>f</sub>y) - { 1500 int i,j,k;

. . . . .

1

trans	graph.c	Page	26	P P I	Fab	8 05:01:00	1400
1501	douple	sart	();				
1502						•	
1503	for (k:	:U;K<)	L1;x++)				
1504	i :	= (y=)	(sart()	00	-k*k))	);	
1505	; ;	: (y+)	(sart(1	00	-k*k))	);	
1506	dot	(x+()	(/2.),5	11	-i);		
1507	det	.(x+()	(/2.),5	i11.·	-j);		
1508	if	(k>8)	(		-		
1509	-	dot	(x+(k/2	.),9	511i	-5);	
1510		dot	(x+(k/2		511 <b></b> i	-1);	
1511		dot	(x+(k/2		511i	+1);	
1512		dot	(x+(k/2	2.) .!	5111	;+2);	-
1513		Jot	(x+(k/2	2.) /	511]	;-2);	
1514		dot	(x+(k/2	2.) .!	511;	-1);	
1515		dot	(x+(k/2	?•),'	511;	(+1);	
1516		dot	(x+(k/a	2.),	5113	;+2);	
1517	}						
1518	}						
1519	returni						
1520	}						
1521	•		•				
1522							
1523	lftarrow(x	· Y )	(				
1524							
1525	seannt	(x+4,	y-4, x,	();			
1526	segnnt	(x+4,	y+4, x,	,);			
1527	returni						
1528	}						
1529	•						
1530							
1531	intros2(xx	, v v , ×	,v) (				
1532	int in	outi					
1533							
1534	if(xx>	(x+12	)) (				
1535	i f	( v y <=	v) (				
1536		iff	vy==y)	ino	ut=0;		
1537		els	e inou	1=1;			
1538	}						
1539	el	se in	nut=27				
1540	}						
1541	else {		•				
1542	í f	(yy<=	y) inp	ut=4	;		
1543	e1	se in	out=3;				
1544	}						
1545							
1546	switch	(ino	ut) (				
1547	CA	se 0:					
1548		sen	innt (xx	-16,	¥¥+**	1,y);	
1549		- 1ft	arrow(	x+1,	y);		
1550		hre	aki				
1551	C 4	se 1:					
1552		sea	mnt (xx	-12,	yy+12	;(8-y,S1+x,	
1553		sea	imnt (x+	15.4	-A, x+	1,y=8);	
1554		1 f t	)worne:	x+1,	v=₽);		
1555		bre	i a k J				
1550	C 4	se 2:	:				
1557		ser	innt (xx	-12,	yy-12	,x+12,y+8);	
1558		sen	imnt (x+	15.4	+#+#+	1,y+8);	
1559		1 f t	arrow(	x+1,	y+8);		
1560		bre	eak\$				

¥



transgraph.c Page 27 Fri Feb 8 05:01:00 1980 case 3: 1561 1562 seamnt(xx+12,yv=12,x+12,y+36); 1563 seamnt(x+12,y+16,x+1,y+16); rghtsemi(x+12,y+26); 1564 lftarrow(x+1,y+16); 1565 1560 break; 1567 case 4: 1568 segmnt(xx+12,yy+12,x+12,y=36); seamnt(x+12,v=16,x+1,y=16); 1569 1570 rahtsemi(x+12,y-26); 1571 lftsrrow(x+1,y=16); break; 1572 1573 } 1574 returni 1575 } 1576 1577 1578 outrns2(xx,yy,x,y) { 1579 int output; 1580 1581 if(xx<(x-12)) ( 1582 if(yy <= y){ 1583 if(yv==v) output=0; 1584 else output=1; 1585 } 1586 else output=2; 1587 3 1588 else ( 1589 if(yy<=y) output=4; 1590 else output=3; 1591 ) 1592 1593 switch (output) ( 1594 case 0: 1595 seamnt(x=1,y,xx+20,yy); 1596 Iftarrow(xx+20,yy); 1597 breaki 1598 case 1: 1599 segmnt(x=12,y=8,xx+28,yy+8); 1600 seamnt(xx+28,yy+8,xx+20,yy+8); 1601 lftarrow(xx+20,yy+8); segmnt(x-1,y-8,x-12,y-8); 1602 1603 break; 1604 case 2: 1605 seamnt(x-12,y+8,xx+28,yy-8); 1606 segmnt(xx+28,yy-8,xx+20,yy-8); 1ftarrow(xx+20,yy=8); 1607 1608 segmnt(x=1,y+8,x=12,y+8); 1609 break; 1610 case 3: seamnt(x-1,y+16,x-12,y+16); 1611 1612 seamnt(x-12,y+36,xx-28,yy-8); seamnt(xx-28,yy-8,xx-20,yy-8); 1613 rghtarroy(xx-20,yy-8); 1614 lftsemi(x-12,y+26); 1615 1616 break; 1617 case 4: 1618 segmnt(x-1,y-16,x-12,y-16); segmnt(x-12,y-36,xx-28,yy+8); 1619 1620 segmnt(xx-28,yy+8,xx-20,yy+8);

l

1

4

1

114

4

' ¥

1

+ 11

ŝ

145

transgraph.c	Page 28	fri Feb	8 05:01:00	) 1980	
1621	rghtarrow(	xx-20,yy4	8);		
1622	lftsemi(x-	12, y-26);	1		
1623	break;	-			
1624					•
1625 return;					
1626 }		_			
1627					
1628					
1629 coltab()	{				
1630 int i;					
1631					
1632 i=11*1	6;				
1633 lodcol	(1++,15,15,1	5); /*	color 0	*7	
1634 lodcol	(i++,0,10,0)	; /*	color 1	*/	
1635 lodcol	(i++,15,0,0)	7 /*	color 2	*/	
1636 lodcol	(i++, 15, 15, 0	); /*	color 3	*/	
1637 loacol	(i++,12,0,12	); /*	color 4	*/	
1638 loncol	(i++,5,5,12)	; /*	color 5	*/	
1639 loncol	(1++,6,6,5);	/*	color ó	*/	
1640 loncol	(1++,5,3,3);	/*	color 7	*/	
1641 10dcol	(i++,10,0,10	]; /*	color 8	*/	
1642 lodcol	(1++,12,5,5)	; /*	color 9	*/	
1645 lodcol	(1++,5,5,5,5);	/*	color 10	*/	
1644 lodcol	(1++,12,12,0	); /*	color 11	*/	
1645 10dcol	(1++,8,7,5);	/*	color 12	*/	
1646 [odeo]	(1++,5,4,2);	/*	COLOT 13	*/	
1647 100001	(1++,0,0,6);	/*	COTOP 14	*/	
1645 101001	(1++,0,0,0);	/*	COLOL 12	*/	
1649 returni					
1050 7		• -			
1453					
1036					/
1033 /	********	********			
1655 /	******** F			GRAPH.C	********
1656 /				4	********
1657 /	*********		*********	*******	*******
1658					
1030					

146

F.R.Y

linkgraph.c Page 1 Fri Feb 8 07:09:45 1980 1 # 2 3 /\*\*\*\*\*\*\*\*\*\*\*\* a 5 /\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\* PROGRAM LINKGRAPH.C 6 \*\*\*\*\*\*\*\*/ 7 /\*\*\*\*\*\*\* \*\*\*\*\*\*\*/ 8 /\*\*\*\*\*\*\* STEPHEN C. JENNINGS JC91 USMC \*\*\*\*\*\*\*\*/ ٥ /\*\*\*\*\*\* ROPERT J. HARTEL CS91 USA \*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\* 10 \*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\* 11 MRITTEN FALL QUARTER 1979 \*\*\*\*\*\*\*\*/ 12 /\*\*\*\*\*\*\* NAVAL POSTGRADUATE SCHOOL \*\*\*\*\*\*\* 13 /\*\*\*\*\*\* MONTEREY, CALIFORNIA \*\*\*\*\*\*\*\*/ 1.4 /\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*/ 15 16 /\* 17 18 20 /\*\*\*\* EXTERNAL DECLARATIONS \*\*\*\*/ 22 23 24 /\*\*\*\* LITERALS \*\*\*\*/ 25 26 #define header 4 /\* contains control # as to bytes read in ... \*/ 27 Noetine 10 /\* the standard input read buffer length ++++ \*/ std 28 #Jefine 100 fired /\* the max # of transitions fired in 1 frame +/ 29 #define frames 150 /\* indicates the total # of network states .. \*/ 30 #define /\* bounds on max # of nodes or transitions .. \*/ 360 . bounds 31 #define upper 2000 /\* defines iotbl max length ..... \*/ 3600 32 #define limit /\* defines names max length ..... +/ 33 34 35 /\*\*\*\* STRUCTURES \*\*\*\*/ 36 37 struct ( /\* data structure information on net nodes .. \*/ 38 39 int ctrll; /\* store control char not used in program ... \*/ 40 /\* index to names array ..... \*/ int nameptr; 41 int marker; /\* initial marker state of the network ..... \*/ /\* x condinate of place ..... \*/ 42 int xcord? /\* v cordinate of place ..... \*/ /\* whether or not place is to be plotted .... \*/ 43 int ycord; 44 int plot; 45 /\* length of name associated with place ..... \*/ int length; 40 47 }file1 [bounds], \*hp1; /\* pointer into data structure ...... \*/ 48 49 50 struct ( /\* data structure information on transitions. \*/ 51 52 int ctrl2; /\* store control char not used in crogram ... \*/ 53 /\* index to names array ..... 4/ int trnotr; 51 int intrn? /\* pointer to inputs for a transition ...... \*/ 55 int outtrn; /\* cointer to outputs for a transition ..... \*/ /\* x condinate of transition ..... \*/
/\* y condinate of transition ...... \*/ 56 int excord; 57 int vycord; SA int trnplot; /\* whether or not transition is to be plotted \*/ 59 int trolen) /\* length of name associated with transition \*/ 60

lin	kgraph.c	Page 2	Fri Feb	8 07:09:45 1980
61	<pre>}file2 [bo</pre>	unds], *bi	•2; /*	pointer into data structure */
63				
64	/**** INTE	GERS ****/		
65	_			
06	int al,a2,	a3, 34;	/*	global storage for each data structure */
67	int buffer	{counds};	/*	buffer into which each frame is read #/
58	int cntrl[		/*	variable containing # transitions fired */
70	int ctrove	rtiow;	/*	Reed track of nodes overflow status */
71	int drited	1017	/*	file descriptor for RUN=Y files
72	int fdobuf	;	/+	file descriptor for RUNZ files
73	int firing	{bounds];	/+	storage into which fired places are read . */
74	int ictri		/*	counter passed to a function*/
75	int ievent	s;	/*	number of non- 3 displayable nodes */
76	int iflag;	_	/•	counter for the interrupt mechanism */
77	int iotoll	upperli	/*	forms input-to-output relationship*/
70	int kthtra	Mei 1 (bouodol ()	/▼ «1• /•	counter for the iterations of the network. */
40	int linkto	+ 1 LOOUNDSI L'	· / •	counter for number of displayed podes \$/
91	int obvies	(2):		store count fields for data structures */
82	int overfl	owthlificed	al (2); /+	data structure to store overflow locations */
83	int set?		/*	user selected conrac graphics screen */
84	int tabcou	nt.)	/*	counter for link revert */
95	int tblctr	;	/*	a counter for version 3reset conditions. */
86	int unique	tbl[fired]	[4]; /+	reset table locations for version 3 */
87	int versi		/*	user selected option */
88	int xinsto	re[fired]}	. /*	coordinates retained for link revert */
00	int xoutst	orelfired) on (fired):	· /*	coordinates retained for link revert */
90	int vinsto	retrired); acelficed]	: /*	coordinates retained for link revert */
92	The Voorse	0101111003		
93				
94	/**** CHAR	ACTERS ***	*/	
95				
96	char coufl	stdl7	/+	ruffer store of file to be executed */
97	char dbufl	stoli	/*	buffer store determining data scan*
98	char thut!	stdj;	/*	buffer to store name of second file
100	char gourt	\$t0// /limi+1:	/*	character array for onde labels
101	char scrot		/+	option variable for display to the screen. */
102	char thufl	stdl;	/*	puffer to store timing variable */
103	char vbufl	stdl;	/+	buffer to store version selected */
104	char timin	a;	/*	variable to set program execution timing - */
105				
105				
107	/*******	***********	*******	**/
100	/ = = = = = = = = = = = = = = = = = = =		N LIV - ##	**/
110	,			
111				
112	main() {			
113	· · · ·			
114	extern	rubout();	/* dec	lare 'rubout' clobally */
115	init()	;	/+ rea	d input file*/
110	detern	ine();	/* ver	ify it user wants to see data structure */
117	disola	9(]] ();	/* (15	DIAY INDUC CO CFC
117	select	(7)	/* 501	CCC VERBION OF SIMULATION & GENISCO SEC #/
120	prevar desuas	e(6/7 de():	/* dea	ware deneator-tonn of thissississississis */

ş

Я

```
Page 3
                       Fri Feb 8 07:09:45 1980
linkgraph.c
121
       nlaces();
                         /* verify correct nodes drawn ...... */
122
       link();
                         /* function depicts network connectivity .... #/
123
       imark();
                         /= starting status of network packets ..... */
       124
125
126
       gnfini();
                         /* closing out graphics facilities ...... */
127
128 }
129
130
132 /**** PROGRAM FUNCTIONS
                               ****/
134
135
136 pause(peroid) {
137
       /* function necessary as sleep() not compatible with signal() */
138
       int i,j,k;
139
140
       printf("***>>interrupt.....");
       for(i=0;i<oeroid;i++) {
141
142
           for(j=0;j<400;j++) {
143
              for(k=0;k<1000;k++) {
144
              }
145
          }
       )
146
       printf("+++>>wait.....");
147
148 return;
149 }
150
151
152 rubout() (
153
       /* function enables the 'brk' key as the interrupt signal */
154
       char halt;
155
156
       space(2);
157
       printf
       ("***>>> received signal...frame number %d...<ret> to continue \n",
158
159
       (iflag+2));
160
       printf
       ("+++>>> for termination of program...type 'brk' from console \n");
while ((halt=getchar())!='\n') {
161
162
163
             /* do=nothing loop */
164
165
       sional(2, nubout);
166 return;
167 F
108
169
170 space(returns) {
171
       int i;
172
173
        for (1=0; i<returns; i++) (
174
           printf("\n");
175
            •
176 return;
177 }
178
179
180 init() (
```

I

1

J

- 4

;

1

10

149

```
linkgraph.c
                          Fri Feb 8 07:09:45 1980
                Page 4
         /* function opens unformatted file & initializes start condition */
181
182
        int a, bufctr, count, fd, i, j;
183
        char c,f7
184
185
        space(2);
        printf("***->LINKGRAPH ILLUSTRATES NETWORK SIMULATION MODELS");
186
187
        space(2);
185
        printf("***->ENTER THE NAME OF THE FILE TO BE PROCESSED .. BUT No");
189
        printf("
                      HOTE THAT THIS FILE MUST BE AN UNFORMATTED FILE NO");
        orintf(*
                      PRODUCED AS A RESULT OF EXECUTING simulator.out \n");
190
191
         printf("
                      THE LAST LETTER OF WHICH MUST END IN LETTER 'X' \n");
 192
       error:soace(2);
 193
         orintf("+++->");
 194
 195
         i=0;
 196
         while((c=getchar()) != '\n') {
197
              couf(i)=c;
198
              i++;
199
         3
200
         cbuf[i]='\0';
201
         bufctr=i;
202
203
         for(j=0;j<bufctr;j++) {
204
             anuf[j]=fbuf[j]=cbuf[j];
205
             if(cbuf[j]=='X') (
200
                 fouf[j]='Y';
207
                 abuf[j]='Z';
208
                 abuf[j+1]=touf[j+1]='\0';
209
                 j=bufctr;
015
             )
115
        )
515
213
        fd = open(cbuf,0);
214
         if (fd <= 0) (
515
            printf
216
             ("***->error occurred in opening file.....try again");
217
             snace(3);
815
             goto error;
219
220
155
        if((count = read(fd;nbytes;header)) != header)
555
            printf("error occurred in nbytes read");
        ievents = (nbytes[1] - 1);
553
224
        al = nbvtes[1];
225
        a = (nbvtes[1]-1)+14;
220
        if((count = read(td,file1,a)) != a)
227
            printf("error occured in file1 read");
228
559
        if((count=read(fd, nbvtes, header))1=header)
230
            printf("error occurred in nbvtes read");
531
        a2=ntytes[1];
535
        a=(nbvtes(1)-1)+16;
233
        if((count=read(fd,file2,a))1=a)
234
            nrintf("error occurred in file2 read");
235
539
        if(lcount=read(fd,nbvtes,header))1=header)
237
            printf("error occurred in header read");
238
        a3=nbvtes[]];
538
        a=(nbytes[1]+1)+2;
        if((count=read(fd,iotbl,a))1=a)
240
```

. 1

```
Page 5
                        Fri Feb 8 07:09:45 1980
linkgraph.c
241
            printf("error occured in iotbl read");
242
243
        if((count=read(fd, noytes, header))!=header)
            nrintf("error occurred in header read");
244
245
        a4=nbytes[1];
240
        a=nbytes[1]+1;
247
        if((count=read(fd,names,a))!=a)
            printf("error occurred in names read");
248
249
250
        close(fd);
251 return;
1 525
253
254
255 determine() {
256
        int if
        char di
257
258
        space(2);
259
        printf("***->FUNCTION 'DETERMINE' ALLOWS THE USER \n");
260
        printf(" TO EXAMINE ALL PRIMARY DATA STRUCTURES");
261
       over:soace(2);
595
593
        printf
        ("***->IF THIS FEATURE IS DESIRED TYPE 1 IF NOT 0, THEN <RET>");
264
265
        space(2);
        print("***=>");
266
267
268
        1=0;
269
        while((d=setchar()):='\n') {
270
            dbuf[i]=d;
271
            i++;
272
        }
        dbuf[i]='\0';
273
274
275
        i=0;
276
        while(dbuf(i):='\0') {
            d=abuf[i];
277
278
            switch(d) {
279
                 case'0':
280
                      scrn='0';
281
                      break;
                 case'1':
282
                      scrn='1';
283
284
                      printf("***->USE CONTROL Q WHEN SCREEN FULL");
285
                      Freak?
286
                 nefault:
                      printf("***->either blank or invalid entry");
287
288
                      anto over:
289
                      breaki
290
            }
291
            i++;
292
        }
293 returns
294 }
295
296
297 display() {
298
       int if
299
       if(scrn=='1') (
300
```

11

ş

.

- 1

1

. 11

151

```
linkgraph.c
               Page 6
                         Fri Feb 8 07:09:45 1980
301
          sosce(2);
302
          bol = filel;
          printf("***-> FILE1 DATA STRUCTURE");
303
304
          space(2);
305
          printf
          ("Infeed
306
                    #
                          marker
                                   . xcord
                                           ycord
                                                     plot
                                                            length \n");
          for (i=0;i<a1; i++){
307
308
              printf("%d \t %d \t
309
                 bol->ctrl1,hol->nameotr, bol->marker,bpl->xcord,
310
                 bpl->ycord,pol->plot,ppl->length);
311
              bp1++;
312
          ÷
313
314
          space(2);
315
          52 = file2;
          printf("***-> FILE2 DATA STRUCTURE");
316
317
          space(2);
          printf
("Infeed
318
319
                     troptr intro
                                    outtrn *xcord yycord trnplot trnlen \n");
320
          for (i=0;i<a2; i++){
              printf("%3 \t %d \t %3 \t %d \t
321
322
                  .bo2->ctrl2,bo2->trnotr,bo2->intrn,bo2->outtrn,
323
                  ho2->xxcord,bo2->yycord,ho2->trnplot,bp2->trnlen);
324
              602++;
325
          )
359
327
          space(2);
328
          printf("***-> IGTSL DATA ARRAY");
329
          space(2);
330
          for (i=0; i<a3; i++) {
331
332
              printf("%d ", iotbl[i]);
          3
333
334
          space(2);
335
          printf("***-> NAMES DATA ARPAY");
          space(2);
33ô
          for (i=0; i<a4+1; i++) {
337
338
            printf("%c",names[i]);
339
          - }
340
          space(2); ·
341
      )
342 return;
343 }
344
345
348
        int notovi
349
350
        y=0;
351
        aenisco (set);
352
        erase();
353
        screen(0.0,0.0,511.0,511.0);
354
        setmod(tyne);
355
        coltan();
350
        colort(11);
357
        for(n=12;n<14;n++) {
358
            color(n);
            for(t=0;t<512;t++) (
359
360
                  seamnt(0,v+t,511,y+t);
```

P

ij

•

1

.

152

- Carthage

```
Fri Feb 8 07:09:45 1980
               Page 7
linkgraph.c
361
            }
       }
362
363 return;
364 )
365
366
367 drawnode() {
368
        /* function displays type % location of network nodes */
369
        char c, *notr, hold;
        int a,b,clrlpl,count,d,entry,h,i,j,k,l,node,test,x,y,*23
370
371
        float s;
372
        double sqrt();
373
374
        bol = filel;
375
        a=0;
376
        count = 17
377
        while((test=(bp1+>nameptr))1=0) {
378
379
            color(14);
380
            test++;
381
            z = Snames[test];
            c = +z;
382
383
384
             if ((b=(bp1->plot))!= 0) (
                 switch (c){
case 'I':
385
386
387
                         node=1;
388
                          x = (bol - > xcord);
389
                          y = (bol->ycord);
                          for(d=0;d<10;d++) {
390
                               segmnt(x=16,y+2+d,x,y+2+d);
391
392
                          }
                          if(vers==3)
393
                                       (
394
                               nptr= &names[test+1];
395
                               hola= *netr;
396
                               clrlbl=atoi(&hold);
397
                               label(x,v,test,clrlbl,node);
                               color(14);
398
399
                          }
400
                       .
                         else (
                               c1r1b1=14;
401
                               label(x;y;test;clrlbl;node);
402
403
                          linktbl(a)(3)=1;
404
                     break;
case 101:
405
400
407
                         node=2;
408
                          x = (ool \rightarrow xcord);
                          v = (t:o1=>ycord);
409
                          for(==0;d<10;d++) {
410
411
                               segmnt(x=10,y=2=d,x,y=2=d);
412
                          if(vers==3) (
413
414
                               notr= Knamesltest+117
415
                               hold= #notr;
416
                               ciribl=atoi(&hold);
                               lahel(v,v,test,clrlbl,node);
417
                               color(14);
418
419
                          }
420
                         else (
```

ų

153

linkgraph.c	Page	8 Fri	Feb	8	07:	09:4	15 1980	
421			cirit	= 1 0	14;			
421			lape	1 ( x		test	vc1-16	l,node);
420		•		•	• • •			
423		, 1 i m li	+ 61 ( = )	1 13	1 = 1	:		
424		1104	τυι ια: 	, ()	, - 1	,		
425		rrea	K P					
426		case 'R'	:	•				
427		node	=0;					
428		¥ =	(bo1-)	>×c	ord	);		
429		y =	(bc1=)	>yc	ord	);	_	
430		for(	d=0;a	<31	;d+	+)	(	
431		seqm	nt					•
432		(x=1	8+d/4	, y +	15-	d,×	+18-0/4	/y+15-d);
433		}						
414		ciri	61=14	;				
425		labe	1 (x . v	, te	st,	clr	bl/nod	e);
433		brea	ik 3					
4 3 0			•					
437			=0:					
4 30			(hele	2.00		1:		
439		x =	(001-	200	010	1:		
440		v -	1001-	221	1 10		1	
441		rari	3=010	~ > 1	101	-16	1 	
442		se.	genti	x = 1	0 / V	-15	-0, 10	//-1/0//
443		)						
444		1 f ( v	ersaa	5)	1			••
445			not	r =	8na	mes	itest+1	] #
446			hol	₫=	*00	tri		
447			c]r	101	=at	0i(	Shold);	
448			l ab	e1(	× , y	',te	st,clrl	bl,nøde);
449			c 0 1	or (	14)	;		
450		}						
451		else	• (					
463		•	cle	161	=14	17		
436			lab	e Î (	x . v	,te	st,clr1	bl,node);
4 J J # E #		•						
434		, bce						
437			,					
470		Case 1	-0:					
457		HOUL				() : ()		
455		× -	(001-			• • •		
459		<b>v</b> -	(001-				1	
460		' 105	(			,	•	
461			\$=k;					• • •
462			1=(x=	- ( 3 (	וזיינו	324	5 - 5 / 2	,,,
463			j=v+9	5 i				
464			)=(x+		urt (	(324	5 - 5 / .	, , ,
465			Seamr	nt (1	<b>i , j</b> ,		);	
466		}						
467		for	(x=0;	; k < ;	19;1	(++)	(	
468			s=kj					
469			_i=(x•	• ( s	ant	(324	4*5)	));
470			j=v=9	57				
471			1=(**	+(s	art	(324	s*s)	));
472			seant	n+ (	1,1	, 1 , j	);	
172		3	/ ·	-		-		
413 490				= 3)	(			
4/4			nati	= م	£n.		(test+	1];
4/3			6.000	. – a ≖	***	otei		
4/0			ام دم ام دم	1,51		-i (8	hold):	
477			1	-14			1.0.0	hlanoge):
478			1.80	e i l 	= / ¥ 1 // 1	हर <b>€</b> 3 4		
479			C01	0 7 (	141	•		
480		}						

-

1 i

ŝ

ł

.....

154

Ľ,

Fri Feb 8 07:09:45 1980 Page 9 linkgraph.c else ( 481 c1r1r1=14; 482 lapel(x,v,test,clrlbl,node); 483 484 3 breaki 485 default: 486 printf 487 ("name not valid identifier"); 488 space(2); 489 breaki 490 > 491 492 linktbl[a][0]=count; 493 linkthl[a][1] = x; 494 linktbl[a] (2) = yi495 a++; 496 497 3 498 count++; 499 001++; 500 if(vers==3) { 501 entry=0; 502 uniquetol (entry) (0] =1 inktol (0) [0] ; 503 uniquetbl [entry] [1]=linktb] [0] [1]; 504 uniquetpl lentry] [2]=1inktb1[0][2]; 505 uniquetb1 [entry] [3]=1 inkto1 [0] [3]; 506 507 rblctr=1; for(i=0;i<a;i++) { 508 it(uniquetbl[entry][]]==linktbl[i+1][1] 88 509 uniquetb1 [entry] (2] ==1 inktb1 [i+1] (2) ) ( 510 /\* do nothing \*/ 511 512 513 ) else ( entev++; 514 uniquetbl (entry) [0]=linktbl[i+1][0]; 515 uniquetbl (entry) [1] = linktbl [i+1] [1]; 516 uniquetol (entry) [2]=linktbl[i+1][2]; 517 uniqueth1 [entry] [3]=1inktb1 [i+1] [3] ; 514 tplctr++; 519 520 Ł 521 } ١ 522 rorclot=a; 523 524 returni 525 ) 526 527 528 places() { 529 int haiajaki 530 if(scrn=='l') { 531 space(2); 532 DEIDT ( "+++->)ATA STRUCTURE LINKTHL "); 533 \$0acn(2); 534 for(i=0;i<nbrolot;i++) { 535 536 537 538 . soace(1); 539 540 )

155

SAMA

7.1

. 1

ł

Page 10 Fri Feb 8 07:09:45 1980 linkgraph.c 541 soace(2); if (vers==3) { printf("\*\*\*->DATA\_STRUCTURE\_UNIQUETEL"); 542 543 544 scace(2); for(j=0;j<tbletr;j++) { 545 for(k=0;k<4;k++) {
 orintf("%d ~-",uniquetb1[j1(k]);
 546 547 548 ł space(1); 549 550 ) 551 } 552 space(2); 553 } 554 return; 555 ) 556 557 558 label(xx,yv,zz,clol,olace) ( 559 /\* getermines node label placement in relation to node \*/ 560 int fii; . 561 562 color(clb1); 563 564 switch (place) { 565 case 0 : if(xx>250) ( 566 567 for(i=0;i<2;i++) { cnarac((xx+(22+(8+i))),vy,names[zz]); 568 569 zz++; } 570 571 } else { 572 573 for(i=0;i<2;i++) { charac((xx=(34=(8\*i))),yy,names[zz]); 574 575 zz++; 576 ) 577 } 578 breaki 579 case 1 : if(xx>250) ( 580 for(i=0;i<2;i++) { 581 582 charac((xx+4+(6\*i)),yy+2,names[zz]); 583 22++; 584 ł SAS 3 586 else { SA7 for(i=0:i<2;i++) { cnarac((xx=(32=(d+i))),vv+2,names[zz]); 588 589 22++; 590 } 591 ) 592 hreat 593 case 2 : 594 if(xx>250) ( 595 for(i=0;i<2;i++) { charac((xx+4+(A+i)),vv=10,names(zz)); 540 597 22++; 598 } 599 ) else ( 600

l

1

Ý

4

- 1

1

\$ 11

156

-

Page 11 Fri Feb 8 07:09:45 1980 linkgraph.c 601 tor(i=0;i<2;i++) { 602 charac((xx+(32-(8+i))),yy-10,names(zz)); 603 77++; } 604 605 > 606 break; 607 ) 608 return; 609 } 610 611 612 link() { 613 /\* function links nodes by information stored in iotbl \*/ 614 int arc, in (40), out (40), plot in (40), plotout (40); int content,ktr,i,k,m,maxinctr,maxoutctr,1,p,inx,iny,outx,outy; 615 616 617 color(14); 618 content=iatb1[1]; 619 ktr=1; 620 while(content1=0) { 621 maxinctr=intbl[ktr]; **62**2 for(i=0;i<content;i++) {</pre> 623 ktr++; in[i]=iotb1[ktr]; 624 625 ¥ 959 ktr++; content=iotel [ktr]; 627 maxoutctr=iotbl[ktr]; 628 629 for(i=0;i<content;i++) {</pre> 630 ktr++; 631 out[i]=iotbl[ktr]; 632 > for(n=0;p<maxinctr;n++) { 633 634 clotinic1=0; 635 for(l=u;l<norelat;l++) ( if(in(o]==1inktb1[1][01) { 636 637 nlotinlo]=1; 638 l=nbrplot; 639 ¥ . 640 ¥ > 641 642 643 for(p=0;n<maxeutctr;p++) { 644 elotout [n]=0; 645 +or(1=0;1<nbrolotf1++) ( 646 if(out(o)==1)nktb1(1)(01) { 647 clotout [o]=1; 648 i=norolati 649 ł 650 } 651 ) 652 653 for(k=);<<=:xinctr;k++) { 654 for(i=0):<=acutetr;i++) { 655 if(plotin(k)==1 &% plotout(i)==1) { 656 657 a=in[k]; hanut [i]; 658 for(1=0;1<nhrp1ot;1++) ( 659 if(]inktb][]][0]==a) { 660 inx=linktbl[]][]];

link	graph.c	Page	12	F	r i	Fel	ь	8	01	1:	09	:	45	1	9	e 0
661							iny	= 1	ir	n k	tb	1	(1	3	12	];
662							1=0	hr	01	l o	t i					
663					}											
664				•												
54C				for	(	=0:	m < ∩	br	•0	le	t i	<b>m</b> :	++	)	•	
607						4 ( )	ink	**	51	1 7	1	0	) =	31	s)	(
000					•	• • •			. 1	in		'n	11	mi)	ſ	117
667									- 1 -		2,		ìr		i i	21:
668																- / /
669					•		m <b>⇒</b> ⊓	CI	ч <b>р</b>		•					
670					,											
671				· · ·						<b>.</b>	• •		<b>.</b>			•
672				110	es	(17	X # 1	וח	/ • •	Ju	τ,		04	Ľ	, ,	,
673			•													
674		}														
675	}															
676	ktr++;	;														
677	conter	nt=iot:	5 T ( 4	tr] 7												
678	}															
679	returni															
680	}				•											
681																
682																
683	lines(x1,)	1,x2,	(Sv	(												
684	int l	nkcase	, x , y	;												
685																
686	ifixle	<=x2)	4													
687	1	f(v]<=	v2)	{												
688	•	i+(	x1==	×2)	ln	k c a	se	= 3	;							
689		els	e {													
690		• • •	- · ·	f(y)	==	y2)	1.	٦k	c a	<b>s</b> e	2	0;				
691			e	lse	۱n	kca	se	= 4	;							
692		)														
693	•	•														
694		140 (														
695	¢.	1 110	x 1 = =	x2)	1.0	kca	se	= 1	;							
A94		els	e lo	kcas	se=	52										
407	,		•													
6077	· · ·															
600		4														
700	6130		v 2 )·	4												
700		; + (	.1==	221	10	kea	se	= 2	;							
702			- 1r			7:										
702	1		• • •													
703	'	100 10		e En	:											
704	, F	136 11			•											
705	7	-		1												
100	541(5	A //														
101	¢	138 V.		- 1 -	<u>،</u> ،	1		-7	0.		21	:				
708		5,00	2 - 20								. ,					
704		¥ 2 1	2-21													
710		v = v	e •			_ //		۰.								
711		<b>8</b> <del>6</del> 6	17 1 ( )		, • •		* # Y	ς,								
712		ser	571 A C	(x+a	• • •	• 4 ,	XØV	<i>,</i> ,	,							
713		rre	13K7													
714	c	ase 1				- 20				. >	<u>م</u>	•				
715		8.61	1 nnt	(¥],	٧l	-20	, x C	• 1	. 2 .	~ C	vJ	•				
716		y = 1	121													
717		¥ 2 1	12+2	01				ς.								
718		5 ± 0	TULE	(x=4	• * '	•4•	<b>X / Y</b>									
719		\$ e (	n n t	( <b>x</b> +4	• Y '	+4,	x , y	1	,							
720		hri	eakj													

ţ

158

0.44

Fri Feb 8 07:09:45 1980 linkgraph.c Page 13 721 case 2: segmont(x1=20,v1,x2+20,v2); 722 723 #=#5+50; 724 v=v51 725 seamnt(x+4,y=4,x,y); 726 seamntlx+4,y+4,x,y}; 727 breaks 728 case 3: 729 segmnt(x1,y1+20,x2,y2-20); 730 x=x5; 731 v=y2-20; 732 segmnt(x=4,y=4,x,y); seamnt(x+4,y=4,x,y); 733 734 break; 735 case 41 736 seamnt(x1+24,y1+24,x2-24,y2-24); 737 x=x2-1et 738 v=y2-16; 739 segmnt(x,y=6,x,y); 740 segmnt(x=6,y,x,y); 741 seamnt(x2-24,y2-24,x,y); 742 break; case 5: 743 744 seamnt(x1+24, y1-24, x2-24, y2+24); 745 x=x2-16; 746 v=v2+10; 747 segrat(x,y+b,x,v); 748 segmnt(x+6,v,x,y); 749 segmnt(x2-24,y2+24,x,y); 750 break; 751 case 6: 752 seannt(x)-24, y1-24, x2+24, y2+24); 753 x=x2+16; 754 y=y2+16; 755 seamnt(x,y+6,x,y); 756 seamnt(x+o,y,x,y); 757 searat(x2+24,v2+24,x,v); 758 treaki 759 case 7: 760 searnt(x1-24,v1+24,x2+24,v2-24); 761 x=x2+10; 762 v=y2-16; 783 4e3mnt(x,y=b,x,y); 764 searat(x+o,y,x,y); 765 =====+(x2+24,v2-24,x,v); 760 break? 757 3 768 neturna 769 } 770 771 772 select() { 773 int int 774 char vi 775 776 scace(2); Drintf("\*\*\*\*>THERE ARE & VEPSIONS TO THIS GRAPHICS PACKAGE \n"); Drintf(" PLEASE SELECT ONE OF THE FULLOWING VERSIONS: \n"); 777 778 779 acaintsoace(2); printf(" VERSION 1 ... PETRI-NET PACKAGE ...... TYPE 1 \n"); 780 159

State State And State State State

-

-----

1

\$

1

1

i.

- 4

```
Fri Feb & 07:09:45 1980
                Page 14
linkgraph.c
                         VERSION 2 ... PACKET REPRESENTATION ... TYPE 2 \n");
781
         orintf(*
782
         printf(*
                         VERSION 3 ... MULTIROUTING PACKAGE .... TYPE 3 \n");
783
         n = 0;
       twice:space(2);
    printf("***=>");
784
785
786
        n++3
787
         i = 0;
         while ((v=qetchar()) l= ! \n') (
788
789
             vouf[i] = v;
790
             1++7
791
        }
         vbuf(i] = '\0';
792
793
         i = 0;
794
795
         while(vbuf(i) 1= '\0') {
796
             v = vouf[i];
797
             if(n==1) {
                    switch(v) {
   case '1';
798
799
                        vers = 1;
800
801
                       break;
802
                      case '2';
                        vers = 2;
803
804
                        break;
805
                      case '3':
806
                        vers = 3;
807
                        break;
808
                      default:
809
                       printf
                        ("***~>incorrect version try again1");
810
811
                        goto again;
812
                       breaka
813
                    }
                    i + + :
814
815
             )
816
             else (
817
                    switch(v) {
                      case '0':
set = 0;
818
819
820
                        breakl
                      case '1':
set = 1;
821
822
823
                        break;
                      case '2':
set = 2;
824
825
859
                       nreaki
827
                      default:
828
                       orintf
829
                        ("+++=>incorrect genisco set try again");
                       printf
830
                        ("
                               set selection should be 0,1,or2");
831
832
                       n = 1;
833
                       acto taicel
834
                       breakt
835
                    }
836
                    i++;
837
             3
838
         ł
839
         if(n==1) (
840
              soace(2);
```

Į

.

ŧ

- 1

- 1

Page 15 Fri Feo 8 07:09:45 1980 linkgraph.c 841 orintf("\*\*\*->NON SELECT THE GENISCO SET YOU WISH \n"); printf(" THE PROGRAM TO BE DISPLAYED TO ..... \n"); 842 orintf(" IN C3 LAB EITHER SETO, SET1 OR SET2 Nn"); 843 844 goto twice; 845 } 846 return; 847 ) 848 849 850 imark() ( 851 /\* marks initial state of system by calling appropriate function \*/ 852 int prolourrergravy; 853 85.4 bp1 = filel; 855 dfltcolor=3; 856 color(sfltcslor); 857 colour=2; 858 ctroverflow=0; while ((g=(ppl->nameotr)) :=0) { 859 if((b=(bc1->plot)) 1=0) { 860 861 switch(vers) 862 case 1: 863 ivers1(); 864 hreak; 865 case 2: 866 ivers2(colour); 867 break; 868 case 3: 869 ivers3(colour); 870 break; 871 } 872 3 if(dfltcolor1=3) color(3); 873 874 001++; 875 > 876 color(14); 877 printg(0,350.,480.,"TIME FRAME = 1"); 878 preread(1); 879 hilite (); 880 displa(); 881 hold(); 882 reset(1); 883 color(13); 884 ovrflow(); orinra(0,350.,480.,"TIME FRAME = 1"); 885 886 return; 887 ) 888 889 890 hold() ( 801 int if 892 time:susce(2); crintf("+++->THIS IS THE INITIAL STATE OF THE NETWORK NO"); 893 894 printf(" ENTER THE TINING MODE FOR EXECUTION .... NO"); 10' FOR DU DELAYS ..'1' FOR FRAME PAUSES NO"); 805 printf(" 896 printf("+++=>"); 897 898 i = 0; 899 while((timins=getchar()) != '\n') ( 900 thuf[i] = timing;

Fri Feb 8 07:09:45 1980 linkgraph.c Page 16 901 1++; 902 903  $tbuf(i) = 1 \cdot 0^{1} i$ 904 905 i = 0;while(thuf(i) 1= 'X0') { 906 907 timino = truf[i]; 908 i++; 909 • if(timing1='0' 48 timing1='1') { 910 orintf("\*\*\*=>incorrect version try again!"); 911 912 goto time; 913 } 914 return; 915 } 916 917 918 ivers1() ( 919 int erxivizi 920 char check; 921 922 e=(tpl=>marker); 923 x=(hol->xcord); 924 y=(tol=>vcord); 925 z=(tp1+>namentr); if((check=names(z+1))!='I' 88 (check=names(z+1))!='0') { 926 927 orinta(0,x-3.0,511.-(y-3),"%d",e); 928 } 929 else { if(check=names[z+1]=='I') printg(0,x~14.0,511.-(v+2),"%d",e); 930 else printg(0,x=14.0,511.=(y=9),"Zd",e); 931 932 3 933 return; 934 } 935 936 937 ivers2(colour) { 938 int erxivizi 939 char check; 940 e=(nol->+arker); 941 942 x={bo1=>xcord); 943 v=(no1->ycord); 944 z=(bo1->nameptr); 945 if((check=names[z+1])!='I' && (check=names[z+1])!='0') { 940 ocxt2(x,y,e,colour); 947 3 940 else ( . if(creck=namus(z+1)=='1') printg(0,x=14.0,511.=(y+2),"%d",e); else printp(0,x=14.0,511.=(y=9),"%d",e); 949 950 951 } 952 returni 953 } 954 955 956 ivers3(colour) { 957 char keep,+kotri int a,aa,b,bo,c,cc,clr[fired],i,k,n,stack,total,x,y; 958 959 960 total=0;

I

1

1

y

4

- 1

1

ł.

. 11

162

El Matrie

```
linkgraph.c
               Page 17
                          Fri Feb 8 07:09:45 1980
961
        n=1;
962
        x=(bol=>xcord);
963
        y=(pol->ycord);
964
        a=(bol->lenath);
965
        b=(bo1->nameptr);
        c=(bol->marker);
966
967
        total=total+c;
968
969
        if(c>0) {
970
            for(i=0;i<c;i++) {
971
                kotr = &names[b+a=2];
972
                keep = *kotr;
                clrlnl=atoi(&keep);
973
974
                n++;
975
            }
976
        )
977
978
        kptr = &names[b+a];
979
        keep = *kotr;
980
        stack=atoi(8keep);
981
982
        if(names(r+(a-1))!='0') {
            983
984
            else stack = stack + 10;
985
        ł
986
987
        for(i=0;i<stack=1;i++) {
958
           b01++;
989
            aa=(bo1=>length);
990
            bo=(bo1=>namentr);
991
            cc=(bpl=>marker);
992
            total=total+cc;
993
            if(cc>0)
                      1
994
                for(k=0;k<cc;k++) {
995
                    kptr = 3names[bb+aa+2];
996
                    keep = *kptr;
997
                    cirln1=atoi(&keep);
998
                    n++;
000
                }
1000
             F
1001
         )
         if(names[0+1]!='I' && names(b+1]!='0') {
1002
           pckt3
1003
1004
           (x,v,total,clr[1],clr[2],clr[3],clr[4],clr[5],clr[6],clr[7],
1005
           colourli
1006
         3
1007
         else ( '
1008
             if(names[n+1]=='I') printo(0,x=14.0,511.=(y+2),"%d",total);
1009
             else printa(0,x-14.0,511.-(y-4),"%d",total);
1010
        }
1011 return;
1012 )
1013
1014
1015 preread(flas) {
         int oucket[2],count,fd,fa,i,nortrns;
1016
          it(flag1=3) {
1017
1018
         if(flag==1) {
1019
            fd = open (fbuf, 0);
             if (fd<=0) {
1020
```

ļ

.

ian∎ Artis

. 1

١,

```
Fri Feb 8 07:09:45 1980
               Page 18
linkgraph.c
                  printf("***->error occurred in opening fd file");
1021
1022
             }
             fafbuf=fd;
1023
              fg = open (abuf, 0);
1024
1025
              if (fa <= 0) {
1026
                  printf("***->error occurred in opening fg file");
1027
             1
1028
             fdgbuf=fg;
1029
         3
1030
         if((count=read (fdfbuf, bucket, 2))!=2) {
1031
             printf("***->error occurred in fd bucket read");
1032
         3
1033
         if((count=read (fdfnuf, buffer,(ievents*2)))!=(ievents*2)) {
             printf("***->error occurred in huffer read");
1034
1035
         3
1036
         if((count=read (fdobuf, bucket, 2))1=2) {
1037
             printf("***->error occurred in fa bucket read");
1038
         }
1039
         if((count=read (fdopuf, cntr1,2))1=2) {
1040
             printf("***->error occurred in cntrl read");
1041
         }
1042
         if(cntr1[0]==0) (
1043
             space(2);
             printf("***=>the last network state has been achieved");
1044
1045
             kthframe=frames+1;
1045
             space(2);
1047
         3
1048
         else {
1049
             if((count=read (fdqbuf, bucket, 2))!=2) {
                    printf("***=>error occurred in bucket read");
1050
1051
             3
1052
             nbetros = cotrl[0] *2;
             if((count=read (fdacuf, firing, nbrtrns))!= nbrtrns) (
1053
1054
                    printf("***->error occurred in firing read");
1055
             }
1056
         }
1057
          ł
1055
          else (
         close(fdfbuf);
1059
1060
         close(fdgbuf);
1061
          - }
1062 return?
1063 1
1064
1065
1066 hilite() (
1067
         int giiin[25], intol, j, k, l, m, maxinctr, maxoutctr, n, out[25],
             outtpl,p,plotin[25],plotout[25],inx,iny,cutx,outv;
1066
1069
1070
         tabcount=0;
1071
         tor(i=0;i<cntr1(01;i++) {
1072
             to2=file2;
              for(j=0;j<firing(i)=1;i++) {
1073
1074
                  002++;
1075
              •
1070
             inttl=(rc2->intrn);
1077
             outtol=(ne2=>outtrn);
             maxinctr=iotbl[intb]];
1078
1079
              for(k=0;k<maxinctr;k++) (
1080
                 intbl++;
```

.....

164

. 1

1.

linkgraph.c	Page 19	Fri Feb	8 07:09:45 1980	
1081	in[k]=ic	tbllintbl	);	
1082	}	•••••••		
1083	maxoutctr=ic	tol louttb	117	
1084	for(1=0;1 <ma< td=""><td>xoutetril</td><td> ++<b>)</b> {</td><td></td></ma<>	xoutetril	++ <b>)</b> {	
1085	outth)++	.,		
1086	out[1]=i	otolloutt	; b117	
1087	}			
1088	for(o=0;o <ma< td=""><td>xinctr;p+</td><td>)+) {</td><td></td></ma<>	xinctr;p+	)+) {	
1089	plotinic	)=0;		
1090	for(1=0;	l <nbrolot< td=""><td>[]]++] {</td><td></td></nbrolot<>	[]]++] {	
1091	11(1	n[0]==11r	ktollij(0]) (	
1092			=1;	
109/	,	I = norpiot	. •	
1095	<b>`</b>			
1096	· ·			
1097	•			
1098	for(s=0;p <ma< td=""><td>xoutctrip</td><td>s++) {</td><td></td></ma<>	xoutctrip	s++) {	
1099	olotout	p]=0;		
1100	tor()=0;	lenbrolat	(;;)++) {	
1101	ific	out [b] == l i	inktb1(1][0]) {	
1102		olotoutic	) = 1 ;	
1103		l=abrplot	. ,	
1104	}			
1105	, )			
1106				
1107	1000LK=000KNH8		** ****	
1100	*Gr(g-0) if(c	lotio/kla	(1) $(1)$	<b>1</b> 4
1110		for (==0); -	<pre>n<nbroint;m++) pre="" {<=""></nbroint;m++)></pre>	<i>,</i> ,
1111		if()i	nktpl [m] [0] == in[k] )	(
1112		i	nx=linktbl[m][1];	
1113		i	ny=linktbl[m][2];	
1114		T	=nbrplot,	
1115		)		
1116		}		
1117		torin=Uin	SABROLOTIA++J (	、 <i>,</i>
1110		17(11	[nktD][n][V]==OUt[g]  t==]{=}{1}{1}{1}{1}{1}{1}{1}{1}{1}{1}{1}{1}{1}	
1120		0	utu=1 initial (2):	
1121				
1122		}		
1123		•		
1124		color(11)	1;	
1125		lingslinx	(,inv,outx,outy);	
1150		Ainstonal	[rancount]=inx;	
1127		vinstorel	[tancount]=iny;	
1128		YOUTSTORE	(tahcount)=outx;	
1124		vourstane	<pre>/tarcountl=outvi </pre>	
1150	,	tarcount+	•••	
1132	· · · ·			
1133	· ·			
1134 }				
1135 returna				
1130 +				
1137				
1138	、 <i>(</i>			
1159 Inkrvt(	] { 			

165

- CLAM

```
Fri Feb 8 07:09:45 1980
               Page 20
linkgraph.c
1141
1142
         color(14);
1143
         for(i=v;i<tabcount;i++) {
1144
             inx= xinstore[i];
1145
             iny= yinstore[i];
             outx= xoutstore[i];
1146
1147
             outy= youtstore[i];
1148
             lines(inx, inv, out x, outy);
1149
         }
1150 returna
1151 )
1152
1153
1154 stage() {
1155
1156
         if (kthframe!=0 %% kthframe<frames+1) {
1157
             oreread(2);
1158
              if(kthframe:=frames+1) {
1159
                  Hilite();
1160
                  if(timina=='1') (
1161
                      disola();
1162
                      pause(1);
1163
                  ł
1164
             }
1165
         •
1166 return;
1167 }
1168
1169
1170 marking() {
1171
         /* function displays successive iterations of the network */
1172
         int colour, draw, i, mark, n, x, y;
1173
1174
         bp1 = file1;
1175
         n = 2;
1176
1177
         /* following loop processes ievent # data segments each pass */
1178
         for(kthframe=0;kthframe<frames;kthframe++) {
1179
             stage();
1180
              if(kthframe>0) (
1181
                  reset();
1182
                  color(13);
1183
                  avetlow();
1184
                  printal ?, 350 ., 490 ., "TIME FRAME = %d", n);
1185
                  n++;
1186
             ł
1187
             iflan = +rnframe;
1186
             draw = (rel=telst);
1189
              dfltcolor=31
1190
             color (tilicolor);
1191
             colour=?;
1192
             ctroverflow=0;
1193
              for (i=0; i < ievents; i++) {
1194
                  if (draw == 1) (
1195
                      ictr = i;
1190
                      switch (vers)
                                     - (
1197
                          case 11
1198
                               vers1();
1199
                              preakl
1200
                          case 21
```

the best matter at 5 hours

166

. .

. .

1

Į,

;

```
Fri Feb 8 07:09:45 1980
               Page 21
linkgraph.c
                              vers2(colour);
1201
                              breakt
1505
                          case 3:
1203
                              vers3(colour);
1204
                              break;
1205
                      }
1206
1207
                  )
1208
                  i = ictr++;
                  if(dfltcolor1=3) color(3);
1209
                 501++;
1210
                  draw = (pol -> plot);
1211
1212
              }
              dfltcolor=14;
1213
              color(afltcolor);
1214
              colour=13;
1215
              printa(0,350.,480.,"TIME FRAME = %d",n);
1216
              if(timina=='1') oause(2);
1217
              Inkrvt();
1218
              bp1 = filel;
1219
1220
         }
1221
         preread(3);
1222 return?
1223 }
1224
1225
1226 ovrflow() (
1227
          int irxry?
1228
1229
1230
          i=0;
          while(overflowtbl(il(0)1=0) {
1231
              x=overflowtb1[i][0];
1232
              v=overflowtb1[i][1];
1233
              i+(x>250) {
1234
              plock((x+3#)*1.,511.-(y+10)*1.,(x+54)*1.,511.-y*1.);
1235
1539
              else (
1237
              block((x-50)*1.,511.-(y+10)*1.,(x-35)*1.,511.-y*1.);
1238
1239
              overflowtb1[i][0]=0;
1240
              overflowth1[i][1]=0;
1241
              i++;
1242
1243
          •
1244 returni
 1245 }
 1246
 1247
 1248 vers1() {
 1249
          int errivizi
 1250
          char checki
 1251
 1252
          e=buffer[ictr];
          x=(001->vcori);
 1253
          y=(nol->vcord);
 1254
          z=(nol->namestr);
 1255
          if((check=names[z+11)1='1' && (check=names[z+11)1='0') {
 1250
               prints(0,x-3.0,511.-(y-3),"%d",e);
 1257
 1258
          1
 1259
          else (
                if[check=names[z+1]=='I') printq(0,x-14.0,511.-(y+2),"%d",e);
 1260
```

í

ļ

i

÷

.;

The state of the state of the

```
Page 22
                        Fri Feb 8 07:09:45 1980
linkgraph.c
1261
              else printa(0,x-14.0,511.-(y-9),"%d",e);
        )
1262
1263 returni
1264 )
1265
1266
1267 vers2(colour) (
1268
         int mark,x,y,z;
1269
         char check;
1270
1271
         x=(bol->xcord);
1272
         y=(bp1->ycord);
1273
         z=(bp1->nameptr);
1274
         mark=buffer[ictr];
1275
         if((check=names(z+1))!='I' && (check=names(z+1))!='0') {
1276
         pckt2(x,y,mark,colour);
1277
         3
1278
         else (
         if((check=names[z+1])=='I') printg(0,x=14.0,511.-(y+2),"%d",mark);
1279
1280
          else printa(0,x-14.0,511.-(y-9),"Zd",mark);
1281
         >
1282 return;
1283 }
1284
1285
1286 vers3(colour) {
1287
         int araarb, nb, mark, marks, clr[fired], j, k, n, stack, total, x, y}
1285
         char keeo,*kptrj
1289
1290
         total=0;
1291
         n=1;
1292
         x=(bpl+>xcord);
1293
         y=(bp1+>ycord);
1294
         a=(bp1->length);
1295
         b=(bp1=>nameptr);
1296
         mark=buffer[ictr];
1297
         total=total+mark;
1298
         if(mark>0) {
1299
             for(j=0;j<mark;j++) {
1300
                 kotr = Rnames[b+a-2];
1301
                 keep = #kptr:
1302
                 clr[n]=atoi(&keep);
1303
                 n++3
1304
             )
1305
         ł
1306
1307
         kctr = &names[b+a];
         keep = *kotrj
1308
1309
         stack=atoi(&keep);
1310
1311
         if(names(b+(a-1))!=*0*) {
             if(names(b+(a-1))=='1') stack=stack+10;
1312
1313
             else (
1314
                   else stack = stack + 30;
1315
             3
1316
1317
         )
1318
         for(j=0;j<stack-1;j++) {
1319
             bp1++;
             ictr++;
1320
```

i

Ż

Ð

÷

168

29

```
Page 23 Fri Feb 8 07:09:45 1980
linkgraph.c
1321
             aat(bol=>length);
             bb=(bp1=>nameptr);
1322
1323
             marks=ouffer[ictr];
1324
             total=total+marks;
1325
             if(marks>0)
                           ł
                 for(x=0;k<marks;k++) (
1326
                      kptr = &names(bb+aa-2);
1327
1328
                      keep = *kptr;
1329
                     cir(n)=atoi(%keep);
                     n++1
1330
1331
                 }
1332
             ,
1333
         3
         if(names(b+1):='I' && names(b+1):='0') {
1334
1335
           pekt 3
           (x,y,total,clr[1],clr[2],clr[3],clr[4],clr[5],clr[6],clr[7],
1336
1337
           colour);
         з
1338
1339
         else {
1340
           color(3);
           if(names[b+1]=='I') printa(0,x=14.0,511.=(y+2),"%d",total);
1341
           else printa(0,x-14.0,511.-(y-9),"%d",total);
1342
1343
         з
1344 returni
1345 1
1346
1347
1348 pekt2(xaxis,yaxis,point,class) {
1349
      switch(point) {
1350
1351
        case 0:
1352
         breaks
1353
        case 1:
         block((xaxis=3)*1.,511.-(vaxis+3)*1.,(xaxis+3)*1.,511.-(vaxis=3)*1.);
1354
1355
         breaki
1356
        case 2:
         block((xaxis=3)*1.,511.-(yaxis+3)*1.,(xaxis+3)*1.,511.-(yaxis=3)*1.);
1357
         block((xaxis=3)*1.,511.=(yaxis=7)*1.,(xaxis+3)*1.,511.=(yaxis=13)*1.);
1358
1359
         breaki
1360
        case 3:
         Dlock((xaxis-3)+1.,511.-(yaxis+3)+1.,(xaxis+3)+1.,511.-(yaxis-3)+1.);
1361
         block((xaxis=3)+1.,511.=(yaxis=7)+1.,(xaxis+3)+1.,511.=(yaxis=13)+1.);
1362
         block((xaxis-3)+1.,511.-(vaxis+13)+1.,(xaxis+3)+1.,511.-(vaxis+7)+1.);
1363
1364
         breaki
1365
        case 41
         nlock((xaxis=3)*1.,511.-(vaxis+3)*1.,(xaxis+3)*1.,511.-(vaxis=3)*1.);
1366
         block((xaxis-3)+1.,511.-(vaxis-7)+1.,(xaxis+3)+1.,511.-(vaxis-13)+1.);
1367
         plocx((xaxis=3)+1.,511.=(vaxis+13)+1.,(xaxis+3)+1.,511.=(yaxis+7)+1.);
136<sup>H</sup>
         block((xaxis+7)+1.,511.-(vaxis+3)+1.,(xaxis+13)+1.,511.-(yaxis=3)+1.);
1369
1370
         breaki
1371
        case 5:
         block((xaris=3)#1.,511.-(yaris+3)*1.,(xaris+3)*1.,511.-(yaris=3)*1.);
1372
         block((xaxis=3)+1.,511.=(vaxis=7)+1.,(xaxis+3)+1.,511.=(vaxis=13)+1.);
1373
         hlock((xaxis-3)+1.,511.-(vaxis+13)+1.,(xaxis+3)+1.,511.-(yaxis+7)+1.);
1374
1375
         block((xaxis+7)*1.,511.-(yaxis+3)*1.,(xaxis+13)*1.,511.-(yaxis=3)*1.);
1376
         plock((xaxis=13)+1.,511.-(yaxis+3)+1.,(xaxis=7)+1.,511.-(yaxis=3)+1.);
1377
         breaki
1378
        case n:
1379
         block((xaxis-3)*1.,511.-(yaxis+3)*1.,(xaxis+3)*1.,511.-(yaxis-3)*1.);
         block((xaxis=3)*1.,511.-(yaxis=7)*1.,(xaxis+3)*1.,511.-(yaxis=13)*1.);
1380
```

مان الاستان المحمد المان المانية من المان المانية. مانية مانية المحمد المانية المحمد المانية المانية المانية المانية المانية المانية المانية المانية المانية المان

1

1

Fri Feb 8 07:09:45 1980 Page 24 linkgraph.c block((xaxis-3)\*1.,511.-(yaxis+13)\*1.,(xaxis+3)\*1.,511.-(yaxis+7)\*1.); 1381 block((xaxis+7)\*1.,511.-(vaxis+3)\*1.,(xaxis+13)\*1.,511.-(vaxis-3)\*1.); 1382 1383 block((xaxis-13)+1.,511.-(vaxis+3)+1.,(xaxis-7)+1.,511.-(yaxis-3)+1.); 1384 plock((xaxis-13)\*!.,511.-(yaxis+13)\*1.,(xaxis-7)\*1.,511.-(yaxis+7)\*1.); 1385 **Dreak**; 1386 case 7: 1387 block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); Dlock((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1388 block((xaxis-3)+1.,511.-(yaxis+13)+1.,(xaxis+3)+1.,511.-(yaxis+7)+1.); 1389 1390 block((xaxis+7)+1.,511.-(yaxis+3)+1.,(xaxis+13)+1.,511.-(yaxis-3)+1.); block((xaxis=13)\*1.,511.-(vaxis+3)\*1.,(xaxis+7)\*1.,511.-(vaxis=3)\*1.); 1391 1392 block((xaxis=13)\*1.,511.-(yaxis+13)\*1.,(xaxis=7)\*1.,511.-(yaxis+7)\*1.); 1393 block((xaxis+7)+1.,511.-(yaxis+13)+1.,(xaxis+13)+1.,511.-(yaxis+7)+1.); 1394 **Dreak**; 1395 default: 1396 overflowtrl (ctroverflow) [0]=xaxis; 1397 overflowtbl (ctroverflow) [1] = yaxis; 1398 ctroverflow++; 1399 block((xaxis-3)+1.,511.-(yaxis+3)+1.,(xaxis+3)+1.,511.-(yaxis-3)+1.); 1400 block((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1401 block((xavis-3)\*1.,511.-(yaxis+13)+1.,(xaxis+3)+1.,511.-(yakis+7)+1.); 1402 block((xaxis+7)\*1.,511.-(vaxis+3)\*1.,(xaxis+13)\*1.,511.-(yaxis-3)\*1.)" block((xaxis=13)\*1.,511.-(yaxis+3)\*1.,(xaxis=7)\*1.,511.-(yaxis=3)\*1.); 1403 block((xaxis=13)\*1.,511.=(yaxis+13)\*1.,(xaxis=7)\*1.,511.=(yaxis+7)\*1.); 1404 block((xaxis+7)\*1.,511.-(yaxis+13)\*1.,(xaxis+13)\*1.,511.-(yaxis+7)\*1.); 1405 1406 color(class); 1407 dfltcolor∓class 1408 if(xaxis>250) { 1409 printg(0, xaxis+38.0, 511.-yaxis, "%d", point-7); 1410 - } 1411 else ( printg(0, xaxis=50.0, 511.-yaxis, "%d", point=7); 1412 1413 1414 break; 1415 ۱, 1416 return# 1417 > 1418 1419 1420 1421 ockt3(xaxis,yaxis,tota],c1,c2,c3,c4,c5,c6,c7,c1ass) { 1422 1423 switch (total) ( case 0: 1424 1425 breaki 1426 case 1: 1427 color(c1); block((xaxis-3)+1.,511.-(yaxis+3)+1.,(xaxis+3)+1.,511.-(yaxis-3)+1.); 1428 1429 break; 1430 case 2: 1431 color(c1); block((xaxis=3)+1.,511.=(yaxis+3)+1.,(xaxis+3)+1.,511.=(yaxis=3)+1.); 1432 1433 color(c2); block((xaxis=3)\*1.,511.-(yaxis=7)\*1.,(xaxis+3)\*1.,511.-(yaxis=13)\*1.); 1434 1435 breakf • 3 1430 case 3: 1437 color(c1); 1438 block((xaxis-3)+1.,511.-(yaxis+3)+1.,(xaxis+3)+1.,511.-(yaxis-3)+1.); 1439 color(c2); block((xaxis-3)+1.,511.-(yaxis-7)+1.,(xaxis+3)+1.,511.-(yaxis-13)+1.); 1440

٩

i

Ð

1

170

linka	raph.c	Page	25	Fri	Feb	8	07:0	9:45	1980
1441	color	(c3);							
1442	block	((xax	15-5)*	15	11(	уах	1 S + I	5)*1	.,(xax1\$+5)*1.,511.=(yax1\$+/)*1.j;
1445	Dreak	i							
1444	case 4.	(61):							
1446	block	((xax	is=3)*	15	11(	vax	is+3	) + 1 -	(xaxis+3)*1.,511(vaxis=3)*1.);
1447	calor	(c2);			••••	, .			
1448	block	((xax	is=3)*	1.,5	11(	yax	is-7	)+1.	/(xaxis+3)*1.,511(yaxis-13)*1.);
1449	color	(c3);							
1450	block	((xax	is=3)*	1.,5	11(	уах	is+1	3) * 1	.,(xaxis+3)*1.,511(yaxis+7)*1.);
1451	color	(c4);		. E	•• -/		7		
1452	DIOCK	itxax	157/)*	1.,,	11(	yax	1573		/(xaxis+13)+1+/311+=(yaxis=3)+1+//
1454	case 5:	,							
1455	color	(c1);							
1456	block	((xax	is=3)*	1.,5	11(	yax	is+3	) *1.	/(xaxis+3)*1.,511(yaxis-3)*1.);
1457	color	(c2);							
1458	plock	((xax)	is=3)*	1.,5	11(	vax	is-7	)+1.	/(xaxis+3)*1.,511(yaxis-13)*1.);
1459	color	(c3);			,				
1400	01004	((232)	15-3/*	1 • 7.7	11(	yax	1571	2141	**(xax1\$*3)*1**31[**(yax1\$*/)*1*)*
1462	Dlock	((****	( <b>*</b> +7)*	15	11.=(	vax	i e + 3		(xaxis+13)+1.,511(vaxis-3)+1.)*
1463	color	(c5);			••••	,			
1464	block	((xax	is-13)	*1.,!	511	(ya	xis+	3) * 1	.,(xaxis-7)*1.,511.+(yaxis-3)*1.);
1465	break	;							
1460	case 6:								
1467	color	(c1);	• • • • •						
1460	DIOCK	((x3x)	18-31*	1	11(	yax	15+3	1#1.	/(xax15+3)*1•/311•=(Yax15=3)*1•J7
1407	color block	((****		15	11.+(		i e = 7	)+1.	.(vavie+3)*1511.=(vavie=13)*1.);
1471	color	(c3);			、	,			
1472	block	((x3x)	is=3)+	1.,5	11(	yax	is+1	3)+1	.,(xaxis+3)*1.,511(yaxis+7)*1.);
1473	color	(c4);							
1474	block	((xax)	is+7)*	1.,5	11(	yax	is+3	) * 1 .	,(xaxis+13)*1.,511(yaxis-3)*1.);
1475	color	(c5);					• •		
14/0	DIOCK	((xax)	(\$=15)	*1.7	511	(ya	x15+	2) = 1	•/(xax1\$=/j*l•/31]•=(Yax1\$=3)*l•j;
1477	color block	((		•1	511	1	<b>vie</b> 4	131+	1 (vavie=7) ±1511. = (vavie+7) ±1.):
1479	break	;				.,.	~ • • • •		
1480	case 7:	•							
1481	color	(c1);							
1482	block	((xax)	is-3)*	1.,5	11(	yax	is+3	)*1.	/(xaxis+3)*1.,511(vaxis-3)*1.);
1483	color	((2))			,		, <b>.</b>		
1484	DIOCK	((xax)		1.13	11(	yəx	15-/	1=1+	/(xaxis+))×1.,711.=(yaxis=1))×1.)/
1485	block	([]]		15	11(		i a + 1	31+1	(vavie+3)+1511.=(vavie+7)+1.);
1487	color	(c4);							
1488	block	((xax)	s+7) +	15	11(	yax	is+3	) = 1 .	,(xaxis+13)+1.,511.=(yaxis=3)+1.);
1489	color	(c5);							
1490	block	((xar)	is-13)	*1.,	511	(ya:	xis+	3)+1	./(xaxis-7)*1./511(yaxis-3)*1.);
1491	color	(66)7				(		121-	1 . (
1476	DIOCK	((X8X) /-7)+	8-13)	-1-27		(ya:	×15+	1314	1 • # L # @ X 1 S = / J # 1 • / 21 1 • = L Y @ X 1 S = / J # 1 • J #
1896	5010F	12138		15	11(	~ * *	<b>i g + 1</b>	31+1	(xaxis+13)+1511.=(varis+7)+1.12
1495	break	3			(				
1496	default	:							
1497	overf	lowtb	lctro	verfi	low] [	0]=	xaxi	\$7	
1498	overf	lowtb	lctro	verf	0#)[	11=	yaxi	<b>\$</b> 7	
1499	ctrov	erfion	***3						

Fri Feb 8 07:09:45 1980 Page 26 linkgraph.c block((xaxis-3)\*1.,511.-(yaxis+3)\*1.,(xaxis+3)\*1.,511.-(yaxis-3)\*1.); 1501 1502 color(c2); block((xaxis-3)\*1.,511.-(yaxis-7)\*1.,(xaxis+3)\*1.,511.-(yaxis-13)\*1.); 1503 1504 color(c3); 1505 block((xaxis=3)\*1.,511.-(yaxis+13)\*1.,(xaxis=3)\*1.,511.-(yaxis=7)\*1.); color(c4); 1500 block((xaxis+7)+1.,511.-(vaxis+3)+1.,(xaxis+13)+1.,511.-(vaxis-3)+1.); 1507 1508 color(c5); 1509 clock((xaxis-13)\*1.,511.-(yaxis+3)\*1.,(xaxis-7)\*1.,511.-(yaxis-3)\*1.); 1510 color(có); block((xaxis-13)+1.,511.-(yaxis+13)+1.,(xaxis-7)+1.,511.-(yaxis+7)+1.); 1511 1512 color(c7); block((xaxis+7)\*1.,511.-(yaxis+13)\*1.,(xaxis+13)\*1.,511.-(yaxis+7)\*1.); 1513 1514 color(class); 1515 ofltcolor=class; 1516 if(xexis>250) { printa(0, xaxis+38.0,511.-yaxis, "%d", tota1-7); 1517 1518 3 1519 else ( 1520 printg(0, xaxis=50.0,511.-vaxis, "%d", total=7); 1521 ÷ 1522 breaki 1523 } 1524 returna 1525 ) 1526 1527 1528 reset() ( 1529 /\* reset function for successive network iterations \*/ 1530 int i,mark,x,y,z; 1531 if(vers==1 :: vers==2) {
 for(i=0;i<nhrplot;i++) {</pre> 1532 1533 1534 color(14); 1535 x=linktbl[i][1]; 1536 y=linktbl[i][2]; 1537 z=linktbl[i][3]; 1538 if(z==0) ( 1539 block((x-3)\*1.,511.-(y-7)\*1.,(x+3)\*1.,511.-(y-13)\*1.); hlock((x-13)\*1.,511.-(y+13)\*1.,(x+13)\*1.,511.-(y-3)\*1.); 1540 1541 ъ 1542 else f 1543 block((x-16)+1.,511.-(y-2)+1.,x+1.,511.-(y-10)+1.); block((x+16)+1.,511.-(y+10)+1.,x+1.,511.-(y+2)+1.); 1544 1545 ) 1546 } 1547 ł 1548 else { for(i=u;i<tblctr-1;i++) { 1549 1550 color(14); 1551 x=uniquetb1[i][1]; y=uniqueth1[i][2]; 1552 1553 z=uniquetb1[i][3]; 1554 if(z==20) ( block((x-3)+1,,511.-(y-7)+1.,(x+3)+1.,511.-(y-13)+1.); 1555 block((x-13)+1.,511.-(y+13)+1.,(x+13)+1.,511.-(y-3)+1.); 1556 1557 ) 1558 else ( block((x-16)+1.,511.-(y-2)+1.,x+1.,511.-(y-10)+1.); 1559 block((x-16)+1.,511.-(y+10)+1.,x+1.,511.-(y+2)+1.)} 1560

.

172

linkgraph.c	Page	27 F	ri Feb	8	07:09:4	45	1980		
1561	}	•							
1562	>								
1563 }									
1564 returni									
1565 )									
1560									
1567									
1568 coltab(	) {								
1569 int	12								
1570									
1571 i=1	1+167						-		
1572 loa	col(i++	,15,15,1	5);	/*	color (	0 1	<b>A</b> /		
1573 104	col(i++	,0,10,0)	;	/*	color 1	1	k/		
1574 100	col(i++	,15,0,0)	;	/*	color a	2 1	*/	-	
1575 lod	col(i++	,15,15,0	);	/+	color 3	3	*/		
1570 103	col(i++	,12,0,12	);	/*	color 4	4 1	k/		
1577 lod	col(i++	,5,5,12)	;	/*	color S	5 1	*/		
1578 103	col(i++	,6,6,5);		/*	color (	5	A/ .		
1579 103	col(i++	,5,3,3);		/*	color i	7 1	k/		
1580 100	col(i++	,10,6,10	);	/*	color &	9	*/		
1581 10.1	col(i++	,12,5,5)	;	/*	color "	9	*/		
1582 lod	col(i++	,5,5,3);		/+	color 1	10	*/		
1583 lod	col (i++	,12,12,0	);	/*	color 1	11	*/		
1584 103	col(i++	, 8, 7, 3);		/*	color 1	12	#/		
1585 103	co](i++	,5,4,2);		/*	color 1	13	×/		
1586 103	col(i++	,0,0,6);		/*	color 1	14	*/		
1587 103	col(i++	. 6 , 0 , 0 ) ;		/*	color 3	15	k/		
1588 returni									
1589 }									
1590									
1591									
1592	/****	******	*****	***	******	* * * :	******	******	***/
1593	/****	****						*****	***/
1594	/****	* * * *	END OF	PR	GRAM L	INK	GRAPH.C	*****	***/
1595	/****	****						*****	***/
1596	/****	*******	*****	***	******	* * * :	*******	******	***/
1597									

## LIST OF REFERENCES

1.	Abra	rms 00	0 n 7 jr	un.	No ic	or ca	ma ti	n 0	a ņ	n ( Ne	i Pt	E: W(	ra Dr	n] k	kl 5,	1	r Pı	Fe		K ti	ัน C	0, e-	. =	C( a]		р •	u	t e 1 S	97 97		•			
2.	Ande	rs St Ex Te	on ru am c	• pl 19	Ge ui es	20 re: 5	Γ£ 5:	9	≜ T C C	'a) (7)	r a pu	C ( n ( i t )	om om In	ipi iy iz	ut S	e C u	r ha rv	I er ve	n ar Y	te s,	e e	co ri Vo	: n . 5 ) 1	r: ti		`t `s '.	i •	יי ה יו (	n a n 0 .	đ	4,	•		
3.	Awad	Ri	Fl ch	ia ar	s	M D	••	I	Sy rw	'S'	te n,		5 In	A	na •••	1	у: 19	51 97	s	•	n	i	D	e	5 1	E.	r	•						
4.	Erov	ne Th ir	• • ?	J. Co of	nı tv	C. ne va	, ct re		Pe or Te	er: 1 1 1	f c t c n r	) r   10]	ra Re lo	a S	c∈ li y,	t	A y y Ti	ia ie	1	ys Re MI	1 5 7	s ea F	a r R		1 1 5 5	E D.	v. i	a] re 19	1 <b>u</b> 20 27	ia t '9	ti ic	l o Dn	n : 5	
5.	Cox,	L Ar Mo UC	ch de D	A. it ls av	ec i	Jr ct	ur Ph Re	е • •	"F S D.	e Oj	rf pe Th U	or era Ici		i i s	n e 1 e 5 2	c 5 5	1 01 81		e: L: u: (?	te Se	c e r r	ti ar t		n Ma 1 S	at er	of h c l 9	ei e 7(		er le ).	i i p	ut ca	;e: 1	r	
5.	Cyps	Di Co	• s • mp	P. ri an	bı y	J. ut	, ed 19	ה) 7	or Sy 8.	וייזו S'	ur te	i) Pri	ca 5,	t	ic Ad	n d	s i:	A 5 0	Г! ŋ-	ch -W	i	te s]	ec .e	tı Y	u I F	e Yu	Տ Ծ	11	f c I s	er h	1:	ıg		
7.	Pavi	es Th	, ei	D. r	Pi	C 0	, t c	a IC	nd ol	•	o t J	h [0]	er nn	•s	, R 1	C 1	0! e3	"p 7	u ai	te nd	r	N S c	le ,r	tı s	# ( •	)r 1	k 9'	s 79	a 2.	n	đ			
ê.	Γίπο	Ta an	7 ct 1	ic St	a] aj	"T L ff	he Ai C	r C	Ir C 11	יף סי פו	a c n t ge	t r	0 1 ס א	f	IJ Sy rc	'e 's 'n	cl t	1n 900 19	0 :5 :7'	10 7.	2	y y	0	n r	t	:h Co	e •	I ma	Fu	t	uı	e		
9.	Ersl	Low Su Ma	• r v r c	Ph ey h,	1	li 19	բ Ը դ 77	च ग 7	• pu	J: it:	r. 17	, Ig	s	M: u	u 1 r v	.t re	i) y	or 5,	0	ce Vo	1	•	) r 9	' ( '	C I N	io	a: •	n i	iz 1,	8	ti	. 01	n	A
12.	Hamn	rin is Sc	ै ह ह ho	el 01	eı	W va 1	n t 97	?9		h	7 5 1	D ( . m1	0 11	Y a	ou ti	ו . כ	B( n	el ,	i) Na	ev av	e	1	;h P	e 'c'	5 5 1	31 ;	П. G	u] ra	l a d	t: .vi	i c a t	)r ;e		
11.	Heit	C⊤e Pa NR ₩a	ye ck L sh	r. et Re	() () () () () () () () () () () () () (	Co Sw or to	ns it t	t c S D	an hi 23	100 100 15		L T N O	ec av	h a	en n 1 1 2	d q R	ue ee	ot ≘s se	h : : ?	er fo rc	s r h	I	Br La	A De De		iu le ra	r a: t	v e s 1 0 1	⊇y t ry	Y	of ed	1	5 .	•
12.	Hobi	Co In	י 5 t	Gr ut it	a e u	ce r te	Ar	rt 1	10 97	1	"I es	)a ; '	v1 78	.d	a	)e	đ P	G t.	0	li	a	tr De	'' ?f	-1 -1	5 2	se e	1	e d C d	ct om	פו	1 u 1	e	r	

- Kahn, Bobert E., "The Organization of Computer Resources into a Packet Fadio Network", IFFF Transactions on Communications, Vol. Com-25, No. 1, Jan 1977.
- 14. Kleinrock, Leorard, "Performance of Distributed Multi-Access Computer-Communication Systems", Proceedings of IFIP Congress 1977, Toronto, Canada, Aug 1977.
- Kleinrock, Leonard, "Principles and Lessors in Packet Communications", Proceedings of the IFEE. Vol. 66, No. 11, Nov 1978.
- 16. Kunzelman, Ronald C., and others. "Packet Radio Experimental Network Research and Development. Final Technical Report", SEI International, Jan 1978.
- 17. Lawson, Billy R., "A Testbed for Evaluation of Packet Padio in an Army Tactical Corps", U.S.Army letter.
- 18. Moli, Cesualdo Ie, "On Networking", Computer Networks and Simulation, North-Holland, 1978.
- 19. Peterson, James L., "Petri-Nets", Computing Surveys, Vol. 9, No. 3, Sept 1977.
- 20. Price, W. L., Computer Networks and Simulation, North-Hollard Publishing Company, 1978.

• •

- 21. Stowers, Douglas M., "Computer Architecture Performance Prediction for Navel Fire Control Systems", Navel Post Graduate School Thesis, Dec 1979.
- 22. Terman, Lewis, M., "The Role of Microelectronics in Data Processing", Scientific American, Sept 1977.
- 23. Thurber, Kenneth J., and Leon D. Wald, "Associative and Parallel Processors", Computing Surveys, Vol. 7, No. 4, Dec 1975.
- 24. FM-24-1, Combat Communications, U.S. Army Field Manual.
- 25. Letter of Agreement (LOA) for PLRS/JTIDS Pybrid, U.S. Army Training and Tectrine Command. ACM 56401, Pept of the Army, 6 July 1979.
## INITIAL DISTRIBUTION LIST

-----

and the second se

1

Ŧ

- --

		No. Conies
1.	I. Larry Avrunin DTNSBDC Code 19 Bethesda, Maryland 20084	1
2.	Mr. P. P. Crabb Code 9134 Naval Ocean Systems Center San Diego, California 92152	1
3.	Mr. C. H. Gleissner DTNSRDC Code 19 Bethesda, Maryland 20084	1
4.	Ms. Kathryn Heninger Code 7503 Navel Pesearch Laboratory Washington, D. C. 20375	1
5.	Mr. Roanld P. Kasik Code 4451 Naval Underwater Systems Center Newport, Rhode Island Ø2840	1
б.	Dr. J. McGraw U. C L.L.L. (L-794) P.C. Box 808 Livermore, California 94550	1
7.	Mr. Penry G. Stuebing Code 503 Naval Air Development Center Warminster, Pennsylvannia 18974	1
8.	Mr. Joel Trimble Code 221 Office of Maval Research 807 North Guincy Arlington, Virginia 22217	1
9.	Mr. Mark Underwood Code P234 NPROC San Diego, California 92152	1

176



10.	Mr. Michael Wallace DTNSRIC Code 1928 Bethesda, Maryland 20064	1
4 4		4
11.	NSWC Code K70 Fahlgren, Virginia 22448	1
12.	Dr. John Zenor	1
	Code 31302 Naval Weapons Center China Lake, California 93555	
13.	Office of Research Administration	1
	Code 012A Naval Postgraduate School	
	Monterey, California 93940	
14.	Defense Documentation Center	2
	Alexandria, Virginia 22314	
15.	Library, Code 0142	2
	Naval Postgraduate School Monterey, California 93940	
16.	Department Chairman, Code 52	2
	Department of Computer Science Naval Postgraduate School	
	Morterey, Ĉalifornia 93940	
17.	Prof. L. Gox, Code 52C1 Department of Computer Science	5
	Naval Postgraduate School	
4.0	Base I Wassessee	~
18.	Pror. J. Wozencraft, Code 74 Chairman, C3 Academic Group	2
	Naval Postgraduate School Monterey, California 93940	
19.	Prof. R. Schell, Code 52Sj	1
	Department of Computer Science Naval Postgreiuate School	
	Monterey, California 93949	
22.	Prof. O. Kodres, Code 52Kr Tepartment of Computer Science	1
	Naval Postgraduate School	
	rorterev, talifornia 93940	

Ŷ

ŗÌ

i

177

VAL.

1.2

21.	Cpt. Robert J. Hartel Code 110	5
	Defense Communications Agency Weshington D.C. 20305	
22.	Capt. Stephen C. Jennings MCDEC	2
	Communication Officers School Quantico, Virginia 22134	
23.	Mr. Fouglas M. Stowers Naval Sea Systems Command Code Sea 62721D	1
	Washington D.C. 20362	



