MIT/LCS/TR-208

A MINICOMPUTER NETWORK SIMULATION SYSTEM

Brock C. Krizan

This blank page was inserted to preserve pagination.

MIT/LCS/TR-208

A MINICOMPUTER NETWORK SIMULATION SYSTEM

Brock Collins Krizan

September 1977

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

This empty page was substituted for a blank page in the original document.

A MINICOMPUTER NETWORK SIMULATION SYSTEM

by

Brock Collins Krizan

Submitted to the Department of Electrical Engineering on June 25, 1977 in partial fulfillment of the requirements for the Degrees of Bachelor of Science and Master of Science in Electrical Engineering.

ABSTRACT

The design, development and use of cost-effective computer networks require information about system behavior given a variety of network structures and operational policies. Because computer networks are complex systems whose behavior is generally not intuitively understood, there is a need for system analysis tools to provide a wide range of performance information.

This thesis describes a simulation system that generates behavioral information for a class of minicomputer network systems. This simulation system is modularly designed with modules for network modelling, specification of the network processing load, and simulation (a discrete event simulator). The network modelling done with the simulation system is based on a general purpose discrete modelling discipline. Flexible network model building blocks made from the basic modelling discipline structures are provided to the simulation system user. To prepare a simulation experiment the user assembles a network model from the building blocks and specifies a network processing load. To generate performance information the network model and load specification are input to the simulator along with simulation control parameters. On completion of the simulation experiment the generated performance information is output in a palatable form to the user. Overall this simulation system is a convenient and flexible system analysis tool for minicomputer networks.

Thesis Supervisor (Academic): Liba Svobodova
Title: Assistant Professor of Electrical Engineering and Computer Science

Thesis Supervisor (VI-A Company): William Gimple
Title: Section Manager, Hewlett-Packard General Systems Division

TABLE OF CONTENTS

ABSTRACT		2
TABLE OF CONTENTS		3
LIST OF FIGURES		6
ACKNOWLEDGEMENTS		7
CHAPTER 1:	INTRODUCTION	8
CHAPTER 2:	BACKGROUND	13
2.1	Simulation Modelling	14
2.2	Simulation System Design	20
2.3	Computer Network Simulation System Examples	23
CHAPTER 3:	DEVELOPMENT OF A SIMULATION SYSTEM	28
3.1	Motivation For The Development Of MNSS	28
3.2	Development Of MNSS	31
3.2.1	The HPC Network Architecture	32
3.2.2	MNSS Design And Use	35
CHAPTER 4:	MNSS MODELLING	38
4.1	The MNSS Modelling Discipline	39
4.2	The Structure Of MNSS Models	46
4.2.1	The Processing Submodel	47
4.2.2	The Communications Submodel	56
4.3	Simulation Using The HPC System Model	62

CHAPTER 5: MNSS IMPLEMENTATION AND USE 65
5.1 Overall Structure Of A MNSS Implementation 65
5.2 Representation Of A Simulation Experiment 68
5.3 MNSS Simulator Characteristics 73

CHAPTER 6: VERIFICATION OF THE MNSS IMPLEMENTATION 77
6.1 Simulation Model Parameters 79
6.2 MNSS Functional Verification 82
6.2.1 Process Switching 83
6.2.2 Remote Request Functioning 84
6.2.3 Store-and-Forward Functioning 86

CHAPTER 7: MNSS EXPERIMENTS 88
7.1 The MNSS Experimental Process 88
7.2 Two Simulation Studies 90
7.2.1 Incremental Network Expansion 90
7.2.1.1 Experiment Results One 93
7.2.2 Four Node Network Expansion 97
7.2.2.1 Experiment Results Two 99

CHAPTER 8: CONCLUSIONS 103
8.1 Limitations Of MNSS 104
8.2 Extensions To MNSS And Further Study 106

APPENDIX A: THE MNSS/3000 IMPLEMENTATION 108
APPENDIX B: VERIFICATION EXPERIMENT 111

APPENDIX C: EXPERIMENT RESULTS 113

REFERENCES 119

LIST OF FIGURES

2.1	Communications Subnetwork Models	25
4.1	MNSS Modelling Structures	42
4.2	A Job Processing Model	44
4.3	General HPC System Model Structure	48
4.4	Processing Submodel	49
4.5	Communications Submodel	57
5.1	MNSS Implementation Organization	67
5.2	Experiment Representation Breakdown	70
5.3	MNSS Probability Distributions	72
5.4	MNSS Simulator Characteristics	75
7.1	Remote Request Level Table	94
7.2	Incremental Network Expansion Experiment Results	95
7.3	Four Node Network Experiment Results	100

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to the many people who have assisted in the creation of this thesis. First of all, thanks go to Hewlett-Packard and its employees; to Jim Cockrum, Bill Gimple, Eric Ha and John Hawkes for their managerial and technical support, and to John Williams for his help in producing a polished copy of this thesis. Special thanks go to Liba Svobodova for her invaluable advice and encouragement, provided from inception of the thesis project to completion of the thesis document. Appreciation is extended to John Tucker for his enlightened management of the 6A program which made possible the author's engineering internship at Hewlett-Packard. And finally, thanks go to Karen Doyle for her personal support during the many months of work which went into this thesis.

CHAPTER 1

A network is an interconnected or interrelated group or system. Network systems have been developed for railroads, telephone communications, mail services and electric power. In each case the development of a network from previously independent components has brought improved service at lower cost.

Based on this tradition of success in the development of network systems, it is not surprising to see a worldwide move towards the use of data processing-oriented communications networks (ie. computer networks). These networks are generally formed by the interconnection, via communications links, of computer systems. Among the goals in the move to develop computer networks are to provide information security, processing reliability and cost-effective processing services.

The physical disribution of processing services in a computer network can be exploited in providing the appropriate degree of information security to interacting processing tasks with different security requirements. One way to do this is to customize each node in a network to provide a different level of security (in terms of operating system safeguards, physical protection of the computer, etc.). Then processing tasks can be assigned to the computer system with the appropriate security and at the same time retain a communications capability to tasks with other security requirements. The network approach to security can be more effective than

placing all tasks within the security structure of a single computer.

The network approach to reliability is based on the redundant processing capability inherent in a computer network. If one computer system in a network fails, the other systems in the network can be used to dynamically recover from the failure. Optimally, the processing load of the system that failed can be shifted without penalty to other systems and the globally observed operation of the network is unchanged. In most cases though, it must be expected that a degradation of network performance will occur after a single system failure. This gradual degradation of services, made possible through networking, is one way to avoid abrupt and total discontinuations of service (ie. when the processor fails in a computer system with no backup processor).

Perhaps the most important feature of computer networks is the potential to provide cost-effective processing services. There are a number of ways to realize this potential. Sharing of hardware, software and information resources can be done with a network of computers of different characteristics. A user can access costly resources without having those resources associated with the locally available computer system (eg. an expensive peripheral need be located on only one system in a network). Another way a network can be cost-effective is by allowing local data processing at the site of data acquisition and use, while at the same time

maintaining data links to remote processing centers (eg. a bank with many branches would be a potential network application). This arrangement yields significant savings in communications costs over a completely centralized processing facility. Yet another cost-effective characteristic of networks is the allowance for a gradual and consistent expansion of the processing capabilities as they are required by an application. Networking can offer a range of processing capabilities more expansive than is offered by any single computer line.

While computer networks have great potential in providing a variety of data processing capabilities and services, they also pose difficult design and implementation problems. A network system is inherently more complex than the individual components that are linked together in the system. In so far as the behavior of a computer system is difficult to predict for a given user environment, prediction of the behavior of a computer network system can be mind boggling. But this behavior must be anticipated to some extent in order to develop effective data processing systems.

There are two primary phases in the development of computer networks. First, communications hardware and software must be developed to allow a computer system to function in a network environment. Modems, communications processors, programs to maintain communications protocols, and a host of other computer communications components must be

designed and implemented. The second phase of network development involves selecting the appropriate computer and communications options in order to fill specific data processing needs. During both phases a network designer needs insight into the behavior of network systems.

System analysis tools provide information to supplement a designer's intuitive understanding of network behavior. These tools include hardware and software monitors for system measurement, analytic models, and system simulators. Most analysis tools are restricted in their application to particular aspects of network behavior. The complexity of network systems, coupled with the inherent limitations of the analysis tools cause this restriction. Therefore, in order to cope with new types of networks, there is a continuing need for new network analysis tools.

This thesis describes the development of a simulation system to be used for analysis of the behavior of a minicomputer network system. The following chapters trace the development of the Minicomputer Network Simulation System (MNSS). Chapters 2,3 and 4 review the research into, and the design of a flexible network model of a particular class of minicomputer system (designated Hewlett-Packard Computer systems or HPC systems). Chapters 5 and 6 describe the design and implementation of an effective interactive simulator that uses HPC network models. Chapter 7 provides a summary of some simulation experiments designed to demonstrate the

capabilities of MNSS. Chapter 8 reviews the progress made by this thesis project and the potential for further work.

CHAPTER 2

BACKGROUND

There is a rich background of work concerned with computer system analysis. From this background are drawn the techniques that are used to develop computer network analysis tools. These tools include analytical models, performance monitors, and simulation systems. Analytical models are usually very efficient but are limited in application to relatively simple, well characterized systems. The use of performance monitors in network analysis is aimed at getting very accurate and specific information. A fundamental limitation of performance monitors is the necessity to have available a working version of the system to be analyzed. Simulation systems can be used to do system analysis at any stage in system development. In addition, a network system can be analyzed in as much detail as is necessary. The efficiency of a simulation system in analyzing network behavior is in general lower than that of an analytical model and the behavioral information derived with simulation is less accurate than that gathered by a performance monitor. Simulation systems, performance monitors, and analytic models, by having different operational characteristics, provide the capabilities for a range of computer network analysis at various stages in system development and use.

The following sections review the most pertinent

background to the development of MNSS. This includes simulation modelling techniques and the implementation and use of computer network simulators. The principles behind the design of simulation systems in general, and MNSS in particular, can be distilled from the sea of work that has been reported. This work can be characterized into two groups corresponding to the principle stages in the MNSS development process. Initially models of minicomputer network systems had to be designed to describe all significant features. Then an effective simulator had to be designed and implemented that uses these network models.

2.1 Simulation Modelling

A representative model of a system is required to do simulation. Models used for simulation can be divided into two general classes: structural models and functional models [27]. Individual system components and their connections are represented in a structural model. This level of detail is appropriate for tasks such as logic simulation of digital systems [5]. A functional model provides a more abstract representaion of a system than a structural model. It describes how a system operates and can be used for mathematical or empirical system analysis. Extensive use of functional models, in the form of flowchart models [2], finite-state models [10], parallel nets [22] and queuing

models [23], is made in simulating complex hardware and software computer system structures.

The effectiveness of a simulation model can be gauged by its ability to represent all significant system features and at the same time minimize the effort required to use it for system simulation. In the process of constructing a simulation model there are a number of factors that impact its effectiveness. These factors involve characteristics of the modelling language used to describe the model, and various qualities of the model. Specifically they include:

1. the flexibility of the modelling language (ie. the model description language) in abstracting real system structures,
2. the efficiency of the modelling language in a simulation context,
3. the ease with which a model can be adapted to changes in the structure of the system modelled,
4. the level of detail that is used in a model to represent significant system features (that may be at different levels of detail in different parts of the system structure), and
5. the effort required to verify and validate the model.

Work is continually being done on modelling languages and methods in order to deal with these factors.

A number of approaches have evolved for developing effective modelling languages. One approach is to have the modelling language be identical to the simulator programming language (ie. the programming language used to implement the mechanism for simulating the model) [11]. While this approach can yield efficient simulation processing, the process of abstracting a system in this type of modelling language may be very difficult. An ever expanding set of instructions may be needed to handle new and evolving system structures. A second approach to developing a modelling language is to have only a partial correspondence with the simulator programming language (eg. using a simulation language such as SIMULA or GASP to implement a modelling language) [16]. With this approach the simulator programming language performs simulation housekeeping chores and well defined activities such as table construction. System representaion in the modelling language can be done with more ease than in the first approach, but the conversion to the simulator language is more complex. Another approach is to use a general purpose language (FORTRAN, APL) [23] to implement the simulator, and have a separate modelling language which specifically deals with the structures of the system to be modelled. Representing system structures in the modelling language becomes increasingly straightforward at the cost of more effort to implement the simulator.

Whatever the relationship between the modelling language and the simulator language, it is advantageous to have the

modelling language provide convenient model description building blocks [7]. The building blocks may vary from the 56 instructions in GPSS-5 to the two principle structures in DYNAMO. By having an effective set of building blocks a modelling language can be used to represent a range of different types of systems; a modeller can be spared the task of mastering several limited (special purpose) languages. In addition a model constructed using a well known set of building blocks can be understood by more people than one constructed using specialized, one-of-a-kind structures.

The complexity of a modelling language can be gauged by the number and types of the associated building blocks. In general a modelling language becomes more difficult to use and understand with larger numbers of building blocks. The degree of correspondence between building blocks and real system structures also influences the complexity of language use. A modelling language for a limited and well defined set of systems can incorporate just a few very specific building blocks and be very easy to use. But a modelling language used for general classes of systems must either have a great number of specific building blocks or a smaller number that are very general. In either case the generalized modelling language is harder to use than a specific modelling language.

Given a particular modelling language, the effectiveness of a model is affected by the technique of the modeller. Modelling techniques that promote the development of effective

models include top-down development, modular design, and submodel organization.

Top-down development is extensively used in software development to facilitate debugging, functional verification, and maintenance. It is not surprising that top-down development has been constructively applied to simulation modelling [6]. Design and verification activities can be integrated together during top-down implementation of a simuation model. Since the conceptual model is usually derived in a top-down manner, this is a natural and efficient approach.

Modular design allows a modeller to adapt a model to changes in a system structure by making localized changes to the model. The model can be effectively used in simulations in which both parametric and structural system characteristics vary [24]. Parametric characteristics are subject to variation by changing single, usually numeric values; structural characteristics involve functional aspects of a system and are varied by changing the functional modules that make up the model description. For example, with a modularly designed network simulation model, not only can the line speeds and message frequencies be varied (parametric characteristics), but also the network configuration, communications protocols and processor scheduling algorithms (structural characteristics) can be changed. Modular design makes a model easier to develop and adaptable to structural variation for

simulation studies.

Submodel organization is a hign level application of modular design (several modules may be included in a submodel) which is facilitated by the decomposition of a system into several system components or subsystems. The simulation of a system is restricted by time and cost limitations when a model is developed at a very fine level (low level) of detail. By using a submodel organization, different levels of detail can be used in individual submodels to ease simulation restrictions while maintaining the accuracy of the model representation [21]. Submodel representaions may range from a decision table to a highly detailed model (subject to simulation apart from the rest of the model). By using a decision table or a simple function to directly generate submodel outputs from the inputs, then analytical results, measurement data, and previous simulation results can be utilized to reduce needless simulation. Where a detailed representaion of a subsystem is needed for modelling accuracy, simulation on the desired low level of detail can be limited to the specific submodel. Excessive simulation is avoided by not having to model all subsystems at the same low level of detail.

A submodel organization can also be exploited in model validation [3]. Each submodel can be exercised and compared to the corresponding real subsystem. The submodels will be less complex than the complete model and easier to validate. Once

each submodel is shown to operate correctly then submodel interactions can be proven correct (or redefined to become correct). This stepwise approach to validation avoids many of the problems of working with large and very complex models. It is especially valuable where there is a model comprised of many duplicate submodels (eg. a homogeneous network model with each node being a submodel).

2.2 Simulation System Design

A computer simulation system provides facilities to build a simulation model, takes as input a workload description, and uses a simulator mechanism to perform experiments. The simulation model and workload description determine the behavioral information that is generated and recorded for performance analysis by the simulator. Once a simulation model or the building blocks necessary to form a set of related simulation models have been developed, the next steps in the development of a simulation system are the design and implementaion of an efficient simulator and a comfortable user-simulator interface.

Computer network simulators can be classified according to the form of the workload description they use; among the most common classes are stochastic and trace driven [27]. For a stochastic simulator the workload is described by probabilistic distributions; resource demands are generated as

random variables from these distributions. In contrast, a trace driven simulator operates with a workload represented as a deterministic sequence of resource demands. Trace driven simulation is very useful in tuning a system for well defined applications. It cannot be used for an application that cannot be represented accurately by a deterministic sequence of resource demands or where exact resource demand information is not available; in these cases stochastic simulation may be appropriate. Both types of simulation can be used to examine the performance of new system designs, alternative system configurations and resource management strategies.

An efficient performance analysis simulator generally simulates only those events that change the system state. Such simulators, called discrete event simulators, jump from event to event in simulated time. The length of the jumps (ie. the real time that would elapse between two events) does not effect the processing required to do a simulation; rather the total number of events is the key factor. Clearly the efficiency of the simulator is directly related to the level of detail of the simulation model. Increased detail is bought at the cost of more events being generated during simulation, and concequently more simulation processing.

MacDougall's BASYS simulator [19] for a disk-based multiprogrammed computer system is a prime example of a discrete event simulator. It does stochastic simulations and is based on a queuing model. Events correspond to the

assignment of a job to the CPU, the release of the CPU to wait for the completion of an I/O request, etc. Handling of the events is implemented in the simulator as event routines. Each event routine performs the actions which correspond to the associated event and also predicts followup events (ie. schedules event routine executions). Event handling is maintained by the simulator through the use of an event list (that reflects the time sequence of events yet to occur), queue information structures, and a job table. Though the BASYS simulator is used to simulate a simple system, its operating principles can be adapted to do more complex simulations.

The character of the user-simulator interface often plays a large part in determining the effectiveness of a simulator. Given an efficient simulator and an accurate simulation model, the user should be able to easily set up, run, and get back the results of simulation experiments. This dictates a flexible interactive environment, not batch processing. The Computer Networks Simulation System developed at the University of Waterloo [14] provides an example of a comfortable user-simulator interface. Network topology and traffic characteristics are input via a conversational dialog. Simulation output is available with or without data analysis. In addition, message delay information can be displayed during the simulation run.

2.3 Computer Network Simulation System Examples

One way to review computer network simulation systems is in terms of the degree to which network components are represented in the corresponding simulation model. A simulation model may focus on a particular component of a computer network (eg. the communications links) and represent the rest of the network system at a very high level. A simulation system that incorporates such a simulation model is used to examine component behavior as opposed to overall network behvior. A simulation system that is used to examine overall network behavior requires a simulation model that represents all network components at some non-trivial level. An important simulation system design decision is to decide how completely the system is to be simulated.

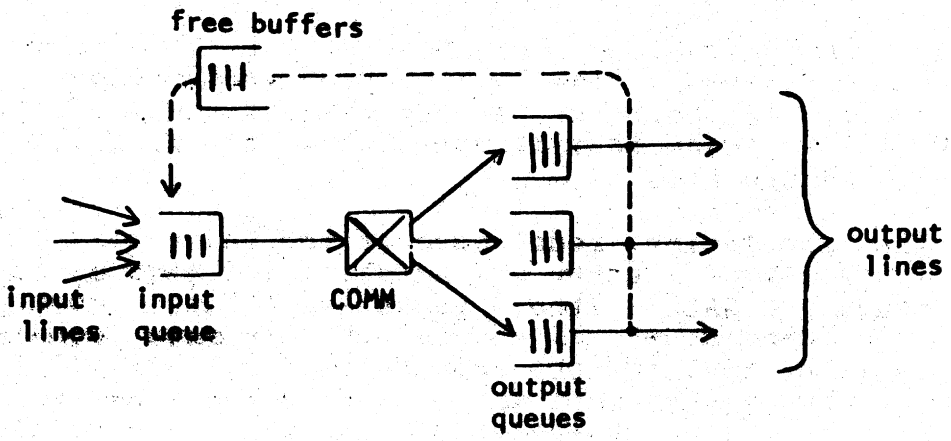
Many simulation systems have been developed that focus on the communications subnetwork in a packet switching network [7,12,13,26]. The communications subnetwork includes a number of dedicated communications processors interconnected by communications links. Each communications processor receives messages from one or more attached host devices (computers, terminal concentrators, etc.), routes messages through the communications subnetwork, and delivers messages to attached host devices when appropriate. The simulation systems that simulate this activity use simulation models that range in detail in representing the structures (eg. communications

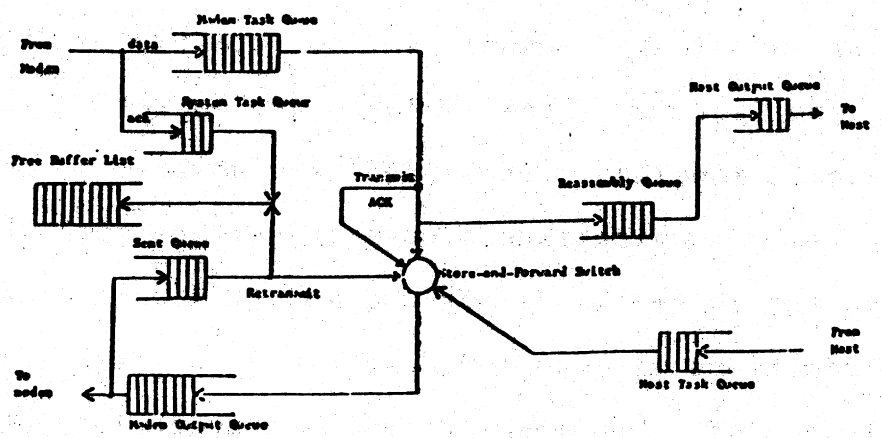
links, communications processors) of the communications subnetwork. The contrast between several simulation models is shown in Figure 2.1.

The simulation model used at the University of Waterloo to study CIGALE (the packet-switching communications subnetwork of the CYCLADES computer network) is illustrated in Figure 2.1a. The objective of this study was to observe the behavior of CIGALE under various traffic conditions. To do this line speeds, line delays and buffer utilization were accurately modelled. The lines were assumed to be error free and the communications processor service time (to do packet routing and buffer handling) was modelled as a constant. These simplifications give an indication of how insignificant detail (in light of the modeller's objectives) is kept out of the simulation model representation.

Figure 2.1b shows a simulation model, used at the Ecole Polytechnique in Montreal [12], which can be considered an enhanced version of the Waterloo model. A more sophisticated host device-communications processor interface is included, representing message segmentation into packets (eg. messages are allowed that are larger than the maximum packet size) and message reassembly at the destination. Also packet transmission acknowledgement and retransmission in case of nonacknowledgement (eg. line errors or insufficient packet buffering at the destination) is represented in the simulation model. Overall, this simulation model can be used by a

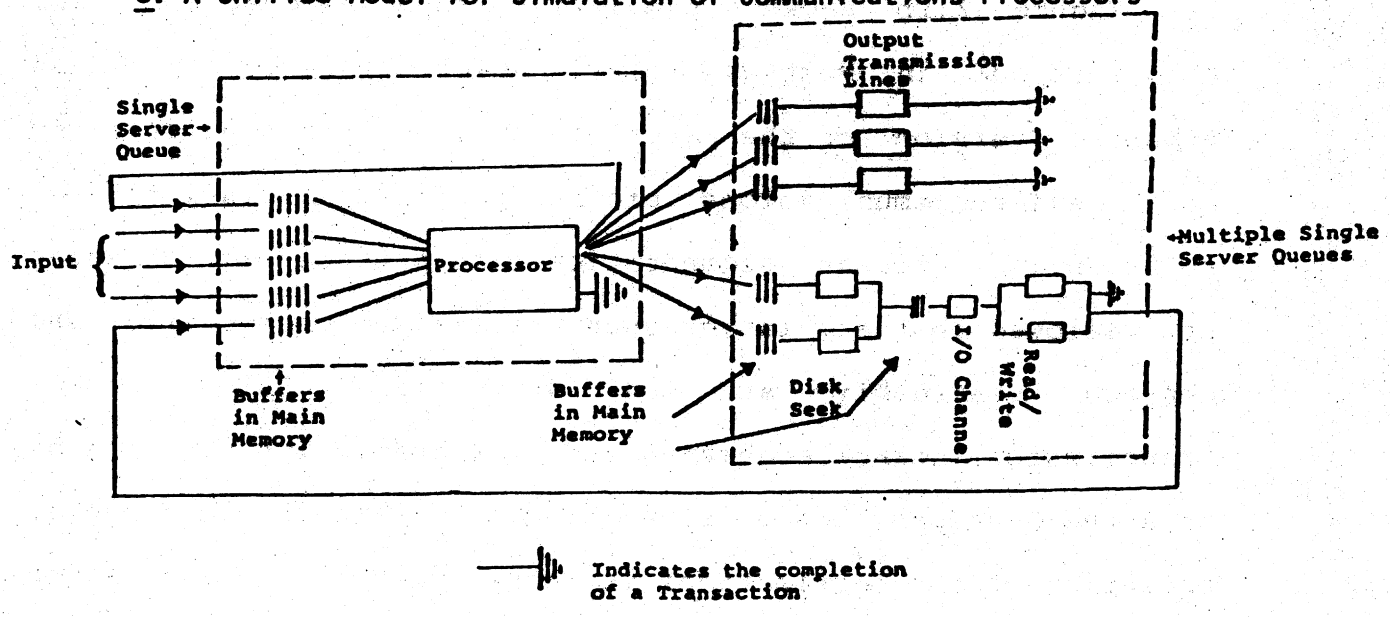
a. Model of a Packet Switch (Waterloo)





b. IMP Packet Processing (Ecole Polytechnique)

c. A Unified Model for Simulation of Communications Processors



Indicates the completion of a Transaction

Figure 2.1 Communications Submetwork Models

simulation system to generate more detailed packet (message) traffic information than the Waterloo system.

A significant variation of the previously discussed simulation models is shown in Figure 2.1c. This simulation model, developed at the Network Analysis Corporation [7], is intended to be used in simulations of communications processors in general. The interesting feature of this simulation model is the detail used to represent packet handling (routing, buffer management). This detail can be contrasted with the constant service time for packet handling in the Waterloo model. In focusing attention on the operations of the communications processor as opposed to network message traffic, a detailed computer system representation for the communications processor was required.

A fundamentally different computer network simulation system was developed by Linsenmayer and Ligomenides [17]. Their simulation model combines a communications subnetwork model (such as those in Figure 2.1) and models of the computer systems attached to the communications subnetwork. In representing in some detail all major components in a computer network, the Linsenmayer model can be used to simulate computer networks as interdependent combinations of hardware/software elements and user job environments. In particular, studies are planned to examine various aspects of global job allocation in a computer network. The ultimate justification of a complete network simulation model, in light

of its inefficiency relative to limited models (eg. those in Figure 2.1), is its usefullness in generating overall network behavior.

MNSS is used for studying the overall behavior of a class of network systems. As such, it is closest in design (functions and limitations) to the Linsenmayer and Ligomenides system.

CHAPTER 3

DEVELOPMENT OF A SIMULATION SYSTEM

The development of another simulation system is justified in light of the inadequacies of existing simulation systems in generating the behavior of constantly evolving network structures. Even where simulation systems accurately generate network behavior, there are often limitations in the user-simulation system interface. The success of MNSS as a network system analysis tool rests on its capability to provide a friendly user environment from which interesting network behavior can be studied. Each step in the MNSS design process was directed at achieving these goals.

This chapter outlines the development of MNSS and identifies those features that make the effort worthwhile. The primary reasons for MNSS development will be presented, followed by a description of the network system to be studied and a general overview of the design of the corresponding simulation system.

3.1 Motivation For The Development Of MNSS

The motivation for the development of MNSS is based on two primary desires: the desire to study the behavior of HPC network systems and the desire to build a 'better' simulation system.

The design of a computer system (composed of one or more computers) is based to some extent on the anticipated behavior of the system in selected environments. The system design may be optimized for throughput, response time, security or some special requirement. In all but simple environments, the behavior of most computer systems is too complex to be derived by intuition and mental wizardry. HPC network systems are under development and there is a need for an analysis tool to study the complex network behavior generated by alternative