ELEMENTS OF PRACTICAL PERFORMANCE MODELING

Edward A. MacNair

IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Charles H. Sauer

IBM Engineering Systems Products
Austin, Texas 78758

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Library of Congress Catalog Card Number: 85-061832

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-269267-8 01

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

To Betsy, John and Scott, Caroline and Elizabeth

CONTENTS

LIST OF FIGURES

PREFACE

Computer systems, communication networks, and manufacturing lines are examples of systems which are sufficiently complex that carefully developed models are necessary for understanding system performance. The behavior of these complex systems needs to be understood in order to be able to design new systems or to improve the operation of existing systems. Typically, contention for resources is the fundamental issue which must be addressed in performance models of these systems. For example, computer system users contend for processing and peripherals, messages in a communication network contend for media and buffers, and jobs in a manufacturing line contend for tools and conveyors. For this reason, we will refer to the systems of interest as *contention systems.*

There have been many other books written about performance evaluation and modeling. Most are concerned with the mathematics involved in solving models. These mathematical techniques have been incorporated in many of the readily available modeling tools. In order to understand how to use one of these modeling tools to construct and solve models, the analyst does not need to have an in-depth knowledge of the mathematical techniques. Rather, he or she needs to fully understand the system to be modeled and the building blocks, or model elements, available with the tools to be used. The purpose of this book is to present the type of information a systems analyst needs to conduct modeling projects employing a model solution package. We will discuss very little of the mathematics used in solving models. Instead, we will be concentrating on the practical approaches needed to construct and solve models.

This book is a practical guide for someone who is planning to use, or is using, a general-purpose software package to do performance modeling. We are interested in fairly general queueing systems like manufacturing systems and detailed computer communication protocols, as well as computer system capacity planning. For this reason, the software packages need to have general capabilities and not be tailored to a particular type of system. These general capabilities need to be present in both the user interface and the solution methods. In addition, we need to be more concerned with problems such as the proper level of detail in the model and problems related to model validation. These types of issues are not as difficult to deal with in more restricted application domains like capacity planning for specific operating systems.

The first chapter gives an overview of the remainder of the book. In the second chapter we discuss the process of modeling. There is a description of what contention systems are and the various components found in these types of systems. The important aspects of formulating a model are introduced, and there is a discussion of how to represent the flow of tasks through a model.

A model diagram aids the analyst in illustrating the behavior of a system. A diagram which employs symbols which correspond to model elements facilitates the building of the model. This type of diagram also makes it easy to describe the model to someone else. Model diagrams and the corresponding model elements are presented in Chapter 3.

This book discusses two methods of solving extended queueing network models. Analytic solutions are described in Chapter 4, and simulation is discussed in Chapter 5. Having the ability to use both types of solution methods is very important to an analyst. There are situations in which one of the solution methods is more appropriate than the other. The advantages and disadvantages of each approach are discussed with recommendations given about which method should be selected in different situations. The Research Queueing Package (RESQ), a modeling tool developed at the IBM Thomas J. Watson Research Center, employs both analytic and simulation solutions.

Chapter 6 focuses attention on the structure of models. Just as the hierarchical approach is recommended for developing programs, constructing models hierarchically is also a beneficial practice. Models with submodels are exhibited to illustrate this point. Some models can be decomposed into submodels which can be solved separately and replaced in the main model. This type of decomposition, when it is appropriate, can reduce the amount of time necessary to find the results of the model.

It is necessary to be able to interpret the meaning of the results obtained from a model and to determine how they relate to the behavior of the system being studied. This is the topic of Chapter 7. Some additional analyses of the results are also discussed. In particular, some graphical representations are illustrated.

It is a difficult task to describe, in general, how to build models of many different types of systems. Chapters 8–11 discuss models of many different types of systems at various levels of detail in an attempt to exhibit this process. Chapter 8 describes models of systems encountered in everyday life situations. Chapter 9 presents some computer system models. Chapter 10 illustrates models of communication networks. Chapter 11 exhibits models of manufacturing systems. Studying the techniques used in these models aids the analyst in constructing models of other systems. These models are not intended to be realistic models of actual systems, but rather to demonstrate how currently available model elements can be used to represent complex features which exist in actual systems.

This book is suitable both as a guide to the practitioner and as a text for an introductory modeling course. Any systems analyst involved in the design of new systems or in improving the operation of existing systems should find this book useful. As a textbook, this book should be appropriate for a senior level or first-year graduate course. The material presented here has been used as the basis for a graduate level course entitled "Computer Systems Modeling Workshop" at the IBM Systems Research Institute. Many of the students who participate in this course have little or no background in modeling before starting the course. Some background in basic probability and statistics is desirable but is not an absolute prerequisite.

ACKNOWLEDGEMENTS

We are grateful for the support of the Computer Science Department of the IBM Thomas J. Watson Research Center and the IBM Engineering Systems Products unit in Austin, Texas, in the preparation of the book. The concepts discussed have been formulated over a number of years while both authors were located at the IBM Thomas J. Watson Research Center. We would like to thank K.V. Karlstrom and P. Henry of Prentice-Hall for encouraging us to write this book. We would also like to thank H. Fromm of IBM and E.D. Lazowska of the University of Washington for their editorial comments on a draft of the book. Their comments have been very helpful in improving the final version.

The development of the extended queueing networks and of RESQ has been an evolutionary process with substantial contributions by many persons. Discussions with K.M. Chandy, D.V. Foster, and C.N. Waggoner provided the original concept of simulation based on extensions to analytically tractable queueing networks. M. Reiser and L.S. Woo made substantial contributions to the initial version of RESQ and the definition of extended queueing networks used by that version. J.F. Kurose and S. Salza helped define the syntax and semantics of the language used in the current version of RESQ and implemented the translator for that language. L. Berger and P. Loewner have made extensions in the model construction phase of RESQ. A. Blum, P. Heidelberger, E. Jaffe, K. Plochinski, P. Rosenfeld, S. Tucci, and P.D. Welch all contributed to the definition and implementation of the numerical and simulation components of RESQ. Discussions with K. Bharath-Kumar and P. Kermani have helped demonstrate the application of extended queueing networks through their extensive application of RESQ in modeling systems developed using the IBM Systems Network Architecture. We are indebted to numerous RESQ users for the many suggestions we have received, for the discussions that have helped crystallize our thinking and improve RESQ, and for the encouragement we have received in the development of this methodology and this tool. Finally, we would like to thank the many students at the IBM Systems Research Institute who have helped us in focusing our attention on the concepts presented here.

Edward A. MacNair
Charles H. Sauer

CHAPTER 1

INTRODUCTION

1.1. SYSTEM BEHAVIOR

In today's world of high technology, computer systems, communication networks, and automated manufacturing systems are in use in many places. These systems are decidedly expensive to design. Once they are installed, it is costly to improve their efficiency. When designing a system, it is often difficult to decide which of the many possible alternatives would give the best performance. After a system is running, it is a complicated task to improve its operation and plan for future changes. There is every reason to believe that these conditions will continue into the foreseeable future.

When we speak of a system we will simply mean a collection of objects which work together to perform a certain goal. Computer systems, communication networks, and manufacturing lines are examples of complex systems whose behavior is of interest to system designers and analysts. Because of contention for service, limited waiting areas, parallel operations, simultaneous activities, multiple interactions, and complex decision mechanisms, the operation of these systems is not easy to predict. The complexity of these systems requires the use of methods and procedures to understand their behavior.

System designers want to be able to predict the behavior of a new system which is being designed and to select the best design from a set of possible alternatives. System analysts are concerned with the effects of changes made to existing systems and whether the systems can handle an increasing amount of work. These systems are rarely static. The design specifications for new systems are continually changing. The amount and type of work processed by existing systems is very dynamic.

1.2. RESOURCE CONTENTION

The systems we are discussing are composed of many components called resources. Examples of resources are a central processing unit (CPU), a communications link, a terminal in an office, or a work station on a manufacturing line. Customers, of course, make use of these resources by visiting them and requesting service from them. During the time that a customer is receiving service, other customers can arrive to request service from the same resource. This causes contention among the customers for the resources and results in queues or waiting lines. The amount of contention and the length of the service requests affect the behavior of the system.

This contention for resources within the system is a fundamental issue which must be understood in studying the operation of these systems. If there is no contention in a system, the system performance is often easy to calculate. However, computer systems, communication networks, and manufacturing lines usually exhibit quite a bit of contention. Jobs in a computer contend for memory, the CPU, input/output (I/O) units, channels, control units, and other resources. Messages in communication networks compete for lines, buffers, tokens, buses, transmission control units, window mechanisms, polling messages, and other transmission media. Tasks in a manufacturing system contend for tools, storage areas, presses, buffers, robots, baths, transfer units, and conveyor mechanisms. Because of the important role that this contention plays in these systems, they will be referred to as contention systems.

1.3. MEASUREMENTS

If we want to determine the behavior of an existing system, we could observe the system while it is running. This is referred to as a measurement of system performance. System measurement is frequently used for conducting performance evaluation of computer systems. There are two major types of measurement tools—hardware monitors and software monitors. Hardware monitors are plugged into a system and measure electrical signals. They normally do not use any of the system resources, but have limited capabilities for the performance measures they can produce. Software monitors are programs which execute on the system. They can produce much more detailed information about the system operation, but they use the system to do this. Therefore, they perturb the measurement data away from the actual values. Since the systems are usually very complex, measurements are often costly and impractical. Even when the information obtained from measurements is sufficient to understand the current operation of the system, it is difficult to use these data to predict the future behavior of the system as the workload and configuration change. When we are trying to design a new system, measurement is rarely of much use.

1.4. PERFORMANCE MODELING

In order to predict the future performance of a system, we need an abstract representation of the system which will embody its behavior. We call this a model of the system. A model contains parameters which represent factors that can be varied to portray different systems. Values of the parameters can depict the amount of service demanded by the customers and the rate at which they arrive at the system. The purpose of the model is to analyze the contention among the customers and the effect it has on the flow of customers through the system. One benefit of modeling a system, in addition to being able to study its behavior in a controlled fashion, is that we frequently gain a deeper understanding of how the system performs.

The procedure of developing a model of a system is not an easy task. One necessary prerequisite is an in depth knowledge of the system to be modeled. Chapter 2 is an overview of the process of modeling. It discusses model formulation and construction, parameter estimation, and model solution and gives a sample model of each of the major types of systems. Because of the complexities involved in performance modeling, methodological approaches are very beneficial.

In building a model, it is helpful to draw a diagram of the flow of customers through the system. This type of diagram also aids in describing the model to others who need to know what the model depicts. In Chapter 3 we present symbols that can be used in drawing diagrams of models. The symbols used in these diagrams will also represent model elements which will be described and used in the remainder of the book. Once the individual building blocks of models are understood, it is relatively easy to put them together in the appropriate manner to represent a specific system in which we are interested. Figure 1.1 shows a typical model diagram of a simple computer system with some interactive terminals, a CPU, and two direct-access storage devices (DASD).

Notice the parallelism which exists in the computer system. Different jobs can be receiving service from the terminals, the CPU, and the DASDs simultaneously. This parallel operation will also be represented in the models we construct.

Figure 1.2 shows a simple model of a communication network. There are some remote terminals connected to a computer by a full duplex line. A separate service mechanism for inbound and outbound messages provides the capabilities of the full duplex transmission.

The model diagram in Figure 1.3 illustrates a portion of a manufacturing system. Jobs entering the system wait in a staging area until a transfer mechanism moves them one at a time through some tool processing and a transfer unit prior to the next operation.

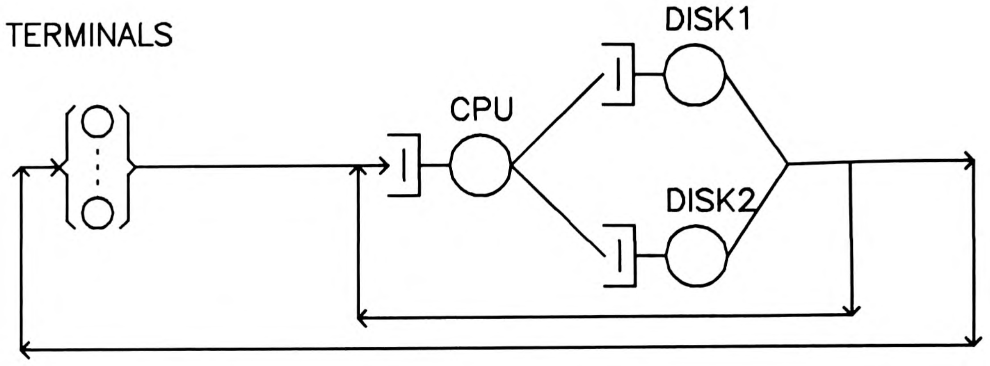


Figure 1.1. Model Diagram of a Simple Computer System

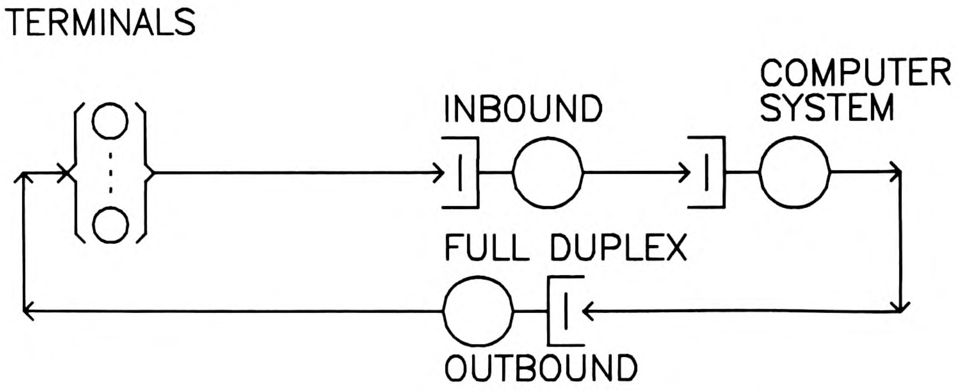


Figure 1.2. Model Diagram of a Simple Communication Network

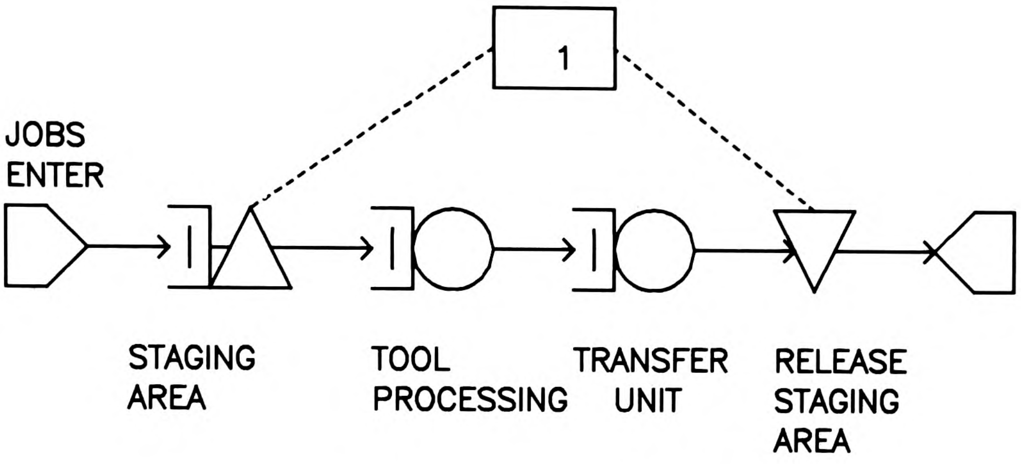


Figure 1.3. Model Diagram of a Simple Manufacturing System

We will be discussing two methods of solving models. A model solved by an analytic method will represent the system by a set of mathematical equations. We give values to the parameters of the model and solve the equations to obtain performance measures which estimate how the system behaves. In Chapter 4 we will discuss analytic solutions in greater detail. A model solved by simulation is a computer program which acts like the system. When we run a simulation, the computer program keeps track of the contention for resources represented in the model and calculates the performance measures based on what it has observed. Simulation will be covered in more depth in Chapter 5.

There are many different kinds of models. We will be concerned with a specific kind of model called a queueing network model. A network is a collection of resources interconnected in some fashion. A queue is a waiting line at a resource. A queueing network model gives us a way of depicting the resource contention we are interested in. A queueing network model permits a high level of representation of the system resources. Queueing network models normally are used to solve simplistic representations of systems. We will describe extensions to queueing network models which permit the representation of complex features that exist in real systems.

This book is a practical guide to using a general-purpose software package to do performance modeling of fairly comprehensive queueing systems. To maintain its generality, the modeling tool cannot be custom tailored to any particular type of system. The user interface and solution methods must be capable of representing and evaluating a broad spectrum of systems. Associated with this universality are problems concerning the proper level of detail in the model and model validation.

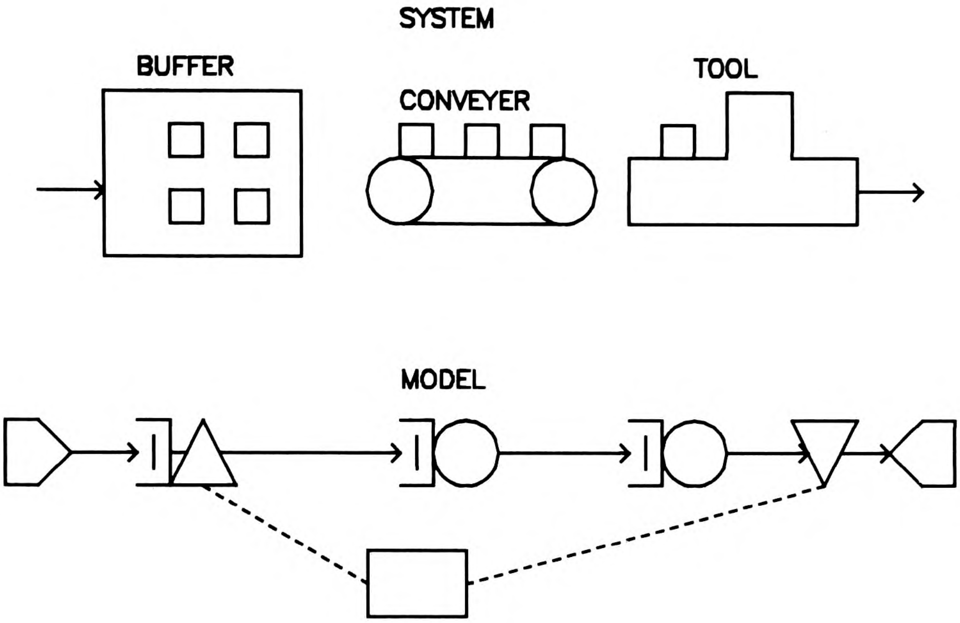
We have mentioned the performance of the system, but we have not defined what we mean. System performance can be measured in various ways. We are often interested in how long it takes a customer to go from one place in the system to another. The fraction of time that a resource is busy could be important. The length of the queues at the resources gives us some insight into system behavior. Sometimes we are interested in the number of customers which complete service at a resource in a unit of time. We will be discussing different performance measures for these kinds of behavior.

Figure 1.4 shows a diagram of a simple system, followed by a model diagram and some sample results. In Chapter 3 we will discuss what the different symbols in the model diagram stand for. The types of results displayed might be used for predicting the system behavior.

When we speak of customers we may be referring to a person, a program in a computer system, a message in a communication network, an office activity, a task to be performed at a manufacturing work station, or a part to be assembled. There can be many different types of customers in the same model.

Our models will give us estimates of system performance. The inaccuracies which are present in the performance measures can come from several sources. The models we develop are only approximate representations of the real system. If certain details of the system are not included, the model will not accurately predict the true behavior of the system. In order to solve for the performance measures, it is necessary to supply values for the model parameters. These parameters could include the rate at which the resources serve the customers, the amount of service demanded by the customers, the number of customers, or the rate at which customers arrive at the model. We usually do not know the exact values of these parameters. When using simulation, therefore, we are actually conducting a statistical experiment and observing the model behavior. The statistical nature of a simulation program can also introduce inaccuracies in the performance measures. However, there are ways of dealing with the statistical nature of the results from a simulation, and we will discuss some of these ways in Chapter 5.

The basic challenges in using queueing network models are to (1) determine which resources are important to have in the model and the



Model Solution

Fraction of time busy: 0.65
Job completion rate: 9 per hour
Time in system: 21 minutes
Number of jobs:12

Figure 1.4. System, Model Diagram, and Solution

characteristics that will most affect performance, (2) formulate a model representing these resources and characteristics, and (3) determine the values for the performance measures of the model. Item (1) requires that the modeler understand the system, and that he or she use intuition in determining what is important. Since a model is a simplistic representation of the system, it is necessary to decide how much detail the model will contain. In doing this, we must make many simplifying assumptions. The structure and level of detail of models will be discussed in Chapter 6. The second problem we mentioned involves a description of the flow of customers through the system and the amount of service required at the resources. After the model has been constructed, calculation of the performance measures could be a time consuming task if a modeling tool is not available. A modeling tool, like many currently available, simplifies the construction and solution of models. The model elements aid in formulating the model. After the model has been defined, the performance measures are calculated automatically.

Some of the currently available modeling tools will be briefly described in Chapter 3. The Research Queueing Package (RESQ), a modeling tool developed at the IBM Research Center, is a collection of programs for constructing and solving analytic and simulation models. RESQ will be used to demonstrate, in a precise manner, the content of actual models and the performance measures which can be calculated by such a modeling tool. Because of the availability of many modeling tools, it is not necessary for an analyst to have an in depth knowledge of the mathematics used in solving models. The mathematical techniques are built into the modeling tools. In addition to knowing when a model has been validated, what is necessary is a good understanding of the system to be modeled and a knowledge of the model elements available in the modeling tools used. In Chapter 2 there is more discussion about the skills necessary for modeling.

After the performance measures are calculated, the analyst must be able to interpret them in order to determine how they are related to the behavior of the system. It is also necessary to be able to change the model or the parameters of the model to investigate how different system designs or various resource characteristics affect the behavior. Chapter 7 will describe how the results of models can be used in determining the behavior of a system.

In order to show how to build models of many different types of systems, we will be discussing many models. In Chapter 8 we will see models of systems which may be encountered in everyday life situations. These systems are not important by themselves, but the models of these systems will illustrate several ways of representing certain types of features in models. The last three chapters before the epilogue, Chapters 9, 10, and 11, investigate models of computer systems, communication networks, and manufacturing systems.

1.5. FURTHER READING

Performance modeling has been discussed by many authors. There are some books which deal with a broad spectrum of modeling including both analytic and simulation solution techniques like Ferrari [62], Kobayashi [98], Lavenberg [100], and Sauer and Chandy [152]. The following books might be of interest for information related to measurements: Drummond [58], Ferrari, Serazzi, and Zeigner [63], Hellerman and Conroy [78] and Svobodova [179]. The two books by Allen [3] and Trivedi [183] contain probability and statistical information related to queueing models. Beizer [17], Gelenbe and Mitrani [69] and Lazowska, Zahorjan, Graham, and

Sevcik [108] are primarily devoted to analytic solutions of queueing network models. There are many books which discuss simulation. Some of the better ones are Fishman [64, 65], Gordon [71], Law and Kelton [106], Maisel and Gnugnoli [117], Sauer and MacNair [156], and Shannon [171]. The special issues of *ACM Computing Surveys* (September 1978) [11, 39, 45, 57, 73, 126, 145, 189] and *Computer* (April 1980) [4, 151, 175] will provide further background. Information pertaining to RESQ can be found in the RESQ documentation [155–163].

CHAPTER 2

THE PROCESS OF MODELING

The process of modeling a system is not an easy task. A great deal of knowledge, intuition, and ingenuity are necessary to conduct a successful modeling project. It is difficult to explain exactly how this process should be conducted. In this chapter, we will attempt, in an informal manner, to describe fundamental aspects of this task.

The noun "model" has several interpretations in the field of systems performance evaluation. Two of the definitions offered by Webster are "a) a small copy or imitation of an existing object, as a ship, building, etc., made to scale. b) a preliminary representation of something, serving as the plan from which the final, usually larger, object is to be constructed." Many performance practitioners use "model" to mean a detailed imitation of system behavior, essentially using Webster's definition (a). Others, ourselves included, primarily use "model" to mean an abstract representation of a system, more in line with definition (b).

A model designed to imitate system behavior in detail is very expensive to construct and often requires a great amount of effort to maintain and use. On the other hand, if used properly, such a model can study very subtle issues of system behavior. In our view, however, the expense of such detailed construction usually outweighs its advantage.

In our experience, it is usually more effective to spend a fair amount of effort hypothesizing which system characteristics are most likely to determine system performance. One can then build an abstract representation of the system which focuses on these characteristics and ignores many system details.

The fundamental steps in this approach are as follows:

1. Study the system design and hypothesize which characteristics determine its performance.

2. Develop a model of the system using a modeling language such as RESQ.

3. Obtain numerical values for the system characteristics represented by model parameters.

4. Use the modeling package to obtain values for the desired performance measures.

These steps are usually not carried out strictly sequentially or even in this order. The availability or nonavailability of the numerical values called for in step 3 will usually influence the representation developed in steps 1 and 2. Once one sees results from a preliminary version of the model, the entire process will likely be repeated, possibly several times. When measurements of prototype systems are available, these may be compared against model results, resulting in further model revision. Once the model is satisfactory, then graphs or tables of model results will be developed for ranges of numerical parameter values and design alternatives.

2.1. UNDERSTANDING THE SYSTEM DESIGN

It is important for the analyst to understand the system to be modeled. Since the model is an abstract representation of the system, it must contain enough information to appear to behave similarly to the real system. Of course, a model can be built at many different levels of detail, but even the highest level of detail must incorporate some of the complexities which exist in the real system. It is probably not possible to develop a reasonable model without knowing quite a bit about the operation of the system.

Some people can begin a modeling project without a deep understanding of the system. However, as they progress, they will find they must acquire such knowledge. This can be done by speaking with knowledgeable people, by reading system documentation, or by studying the actual system. The act of modeling a system usually develops insight into how the system behaves. Thus an added benefit of modeling is being compelled to learn the characteristics of the system behavior.

2.2. FORMULATING MODELS

Model formulation is an art rather than a scientific discipline. It requires a comprehensive knowledge of the system to be modeled. Intuition is needed to determine which system resources are important and how they are to be represented in the model. There are no rigorous methods for accomplishing this task. However, there are some guidelines which can be followed. We will discuss some guidelines and describe different types of models that can be used.

The purpose of the model is an important aspect of formulating the model. Should the model be very simple or should it contain most of the details which are found in the real system? Is it sufficient that the results be gross estimates of system behavior, or is it necessary to produce very accurate performance measures? The purpose of the model will strongly influence the answers to these questions.

The *important* system resources and mechanisms must be identified and included in the model. In a computer system, the processors and I/O devices are usually important resources, and the scheduling algorithms for these resources are important mechanisms. In a communication network, both the links and the protocols are significant. In a manufacturing system, the important resources include tools, buffers, and transfer units. It is necessary to determine how fast demands on a resource can be satisfied. This depends on resource capacity, service demands, scheduling, and so on. The frequency of visits to each resource and/or the flow of work through the system must be represented.

2.3. TYPES OF MODELS

Usually, one begins with definite notions of the type of model to be constructed. For example, one can think in terms of relatively imitative models, corresponding to our first definition, or relatively abstract models, corresponding to the second definition. Often analysts classify models according to the methods used to obtain numerical results, for example, simulation models versus "analytic" models. As suggested above, we will focus on relatively abstract models. Though most of the models we consider will be solved by simulation, we will largely ignore solution methods while constructing models.

An analyst will rarely develop an abstract model without consciously considering previously constructed models. Rather, one will use general characteristics of one's own models or others' models as a framework for constructing new models. Assuming we are working with relatively abstract models, we can classify models corresponding to the framework used in constructing the models. We will use "queueing networks" as the framework for the models we construct. The further reading section at the end of this chapter cites descriptions of various frameworks others have found useful.

A basic queueing network consists of one or more entities which we call "service centers" and a set of "customers" which receive service at the centers. A service center consists of one or more servers, corresponding to resources in the modeled system, and a waiting area (a "queue") for customers needing service. Some analysts refer to service centers as "queues," or as "servers." A customer corresponds to an entity which circulates in the modeled system, for example, a computer system transaction, a message in a communication net, or a printed circuit board to be populated with chips. Some analysts refer to customers as "jobs" or as "transactions."

The simplest queueing network consists of a single service center, representing the entire system. The system may be very complex, but the model is simple. Of course, single service center models cannot capture all the complexities which exist in real systems. However, sometimes this type of model provides useful information. A single service center in a queueing model has several mechanisms relating the customer to the waiting area and the server. Figure 2.1 illustrates some of these mechanisms. The customers arrive at the service center and demand service. The patterns of arrival and service will be discussed in Section 2.4. When customers are waiting for service and a server is free, a decision must be made as to which customer goes into service next. Scheduling algorithms are discussed in Section 2.5.

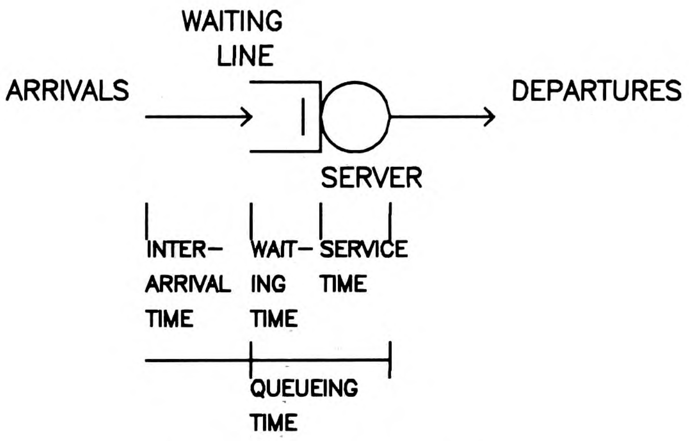


Figure 2.1. A Service Center

There can be any number of servers at a service center. The customers wait in a waiting line or queue until a server is free. A server can serve a customer according to a constant rate, or the rate of service can be dependent on the number of customers in the queue. The waiting line can have an infinite capacity for customers waiting for service, or the waiting room can be finite. There can also be different types of customers which we refer to as classes of customers, demanding different amounts of service.

The type of resource which can be represented by a service center such as those just described will be called an active resource because the servers are actively engaged in providing service. There are some resources which do not behave like active resources. We call these passive resources. There are usually a limited number of items of this type of resource which have to be allocated to customers, held onto for a while, and released by the customers. The passive resource permits the sharing of a finite number of items of the resource. The items which are allocated and released are called tokens of the passive resource. They are analogous to the servers of an active resource. However, the amount of time that a customer holds onto the passive resource is determined by the amount of time that it takes to visit other resources after it has been allocated tokens and until it has released the tokens. One important use of a passive resource is to represent the simultaneous possession of more than one resource. For example, it can represent a job in a computer system which needs to be allocated some memory, and while holding onto this memory, also receives service at an active resource like a CPU. In a communication network or manufacturing system, a passive resource can represent a finite number of buffers or storage areas. In Chapter 3 we will describe more features of passive resources. Passive resources are very useful for representing many instances of complex system behavior. We will see many examples of using passive resources in models.

In actual systems there are many resources which are interconnected. In models, we can connect single resources in series, in parallel, or in any complex fashion. If we have two resources connected in series, and after completing service at the second resource we go back to the first, we call this a cyclic queueing model. Figure 2.2 illustrates a cyclic model along with several other types of models which are briefly described here. A model which allows new customers to be generated and eventually to depart is called an open model. A closed model contains a fixed number of customers. There are never any new customers to arrive, and no customers depart. A special case of the closed model is referred to as the central server model. In this type of model there is a special resource to which all customers return. When a customer leaves this resource, it has a certain probability of going to any of the other resources.

There are models which can be constructed using different combinations of the types of models just described. A model which contains both open and closed paths is called a mixed model. A hierarchical model is one which is constructed at different levels of detail. A hybrid model will employ different solution techniques for solving various parts of the model. We will have more to say about hierarchical and hybrid models in Chapter 6.

There can be many different types of customers in a model. In a computer system, one type of customer might represent interactive jobs and another type of customer might be for batch jobs. In a communication network, one type of customer might represent the messages which are flowing through the network and a second type of customer could be the

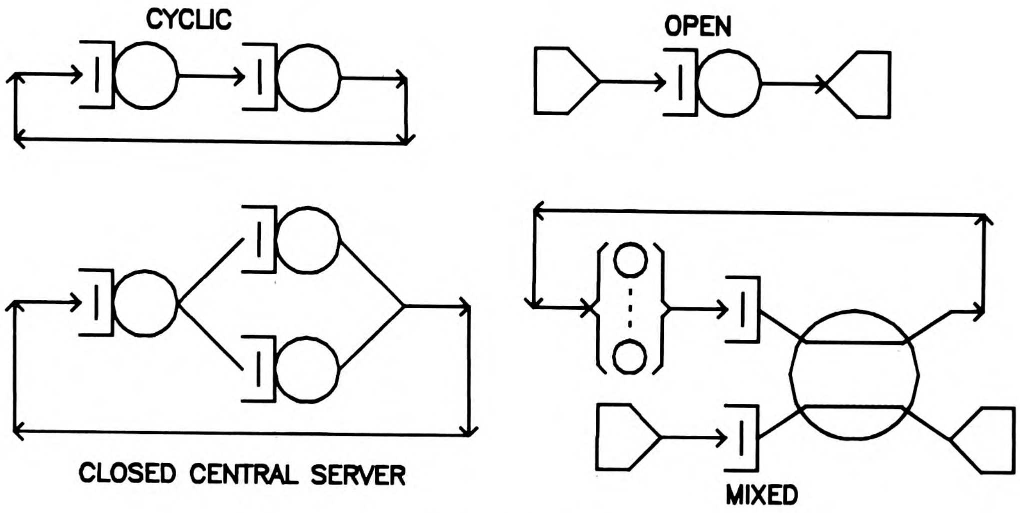


Figure 2.2. Different Kinds of Models

acknowledgements. In a model of a manufacturing line there could be normal types of jobs and jobs which represent tool failures. The different types of customers can travel over different paths and demand different amounts of service.

When a customer of any type leaves a center some mechanism must be provided for determining what center the customer will visit next. This is called a routing decision. We can make routing decisions based on a set of probabilities which add up to one for all of the possible destinations, or we can make the decision based on some condition which exists in the model at the time of the decision. For example, we could choose to visit a service center if there is a server available, otherwise we could go to a different center.

2.4. PROBABILITY DISTRIBUTIONS

Customers arrive for service according to a certain pattern called an "arrival distribution." The amount of service required has a corresponding pattern, which we call the "work demand" distribution. In the special, but typical, case where the server has a fixed capacity, the values of the work demand distribution can be divided by the capacity to determine a pattern called the "service time distribution." Passive queues have corresponding distributions for numbers of tokens needed by customers. Other distributions are associated with other network elements.

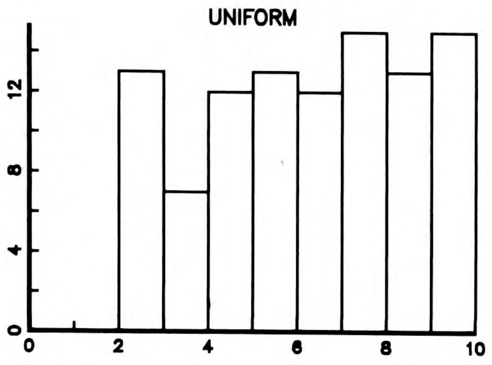
One approach to modeling, known as "trace driven modeling," requires the analyst to measure the patterns which occur during a particular operational period of an actual system. The specific values are recorded on magnetic tape and then used as input to a simulation model of alternate system designs and configurations. However, the quantity of data involved is usually large, and direct representation of distributions in this manner is usually impractical.

A fundamental aspect of the modeling process is characterizing distributions so that the data are manageable. Usually this consists of making a series of assumptions about the distribution to allow a convenient representation. The most important of the typical assumptions is that there is no pattern in the data values, that is, that a given value is independent of prior values. This assumption can be relaxed somewhat, to assume that the values have a relatively simple pattern, but without some sort of independence assumption or assumption of a simple pattern of values, the analyst is forced to trace driven modeling. *From now on, we will make the assumption that, for a particular distribution, the values are independent and identically distributed.*

With this assumption, we can focus on frequency of particular values. The most straightforward way to do this is to define a set of intervals of values, for example, [0,0.1), [0.1,0.2), . . . and determine the relative frequency of values in each of these intervals. Graphically, this is equivalent to producing a histogram such as the one in Figure 2.3.

Characterizing a distribution in terms of frequency of value intervals is often practical, both in terms of developing the characterization and in terms of using it in a simulation model. However, this "brute force" approach usually is intractable for solution methods other than simulation. Further, the analyst may not know much about a distribution and may not be able to determine (or conjecture) the distribution in terms of frequency of intervals. A representation which is simpler still is likely to be easier to develop, easier to deal with in the model, and produce satisfactorily accurate model results.

The simpler representation usually used is a "probability distribution function," or PDF. A PDF specifies the probability that a distribution value



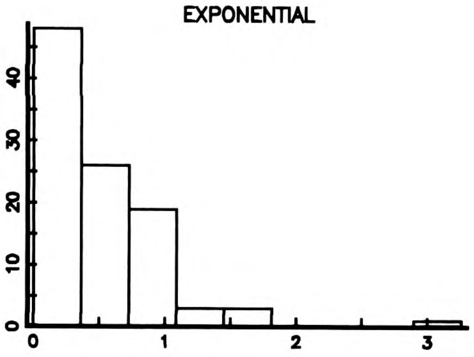


Figure 2.3. Histograms (Relative Frequency)

will be less than or equal to the function argument for all possible values of the distribution. For example, the PDF of the "uniform" distribution is

$$F(x) = \begin{array}{ll} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & x > b \end{array}$$

Where the derivative of the PDF is well defined, it is often mathematically useful to deal with that derivative. The derivative of the PDF is known as the "probability density function." For example, for the uniform distribution, the density function is

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

The uniform distribution gets its name because the density function is uniform over the interval $[a,b]$. Other well known and useful PDFs include the exponential, Erlang, constant, hyperexponential, and normal PDFs. Figure 2.4 illustrates some of these PDFs, and Figure 2.5 shows the corresponding density functions.

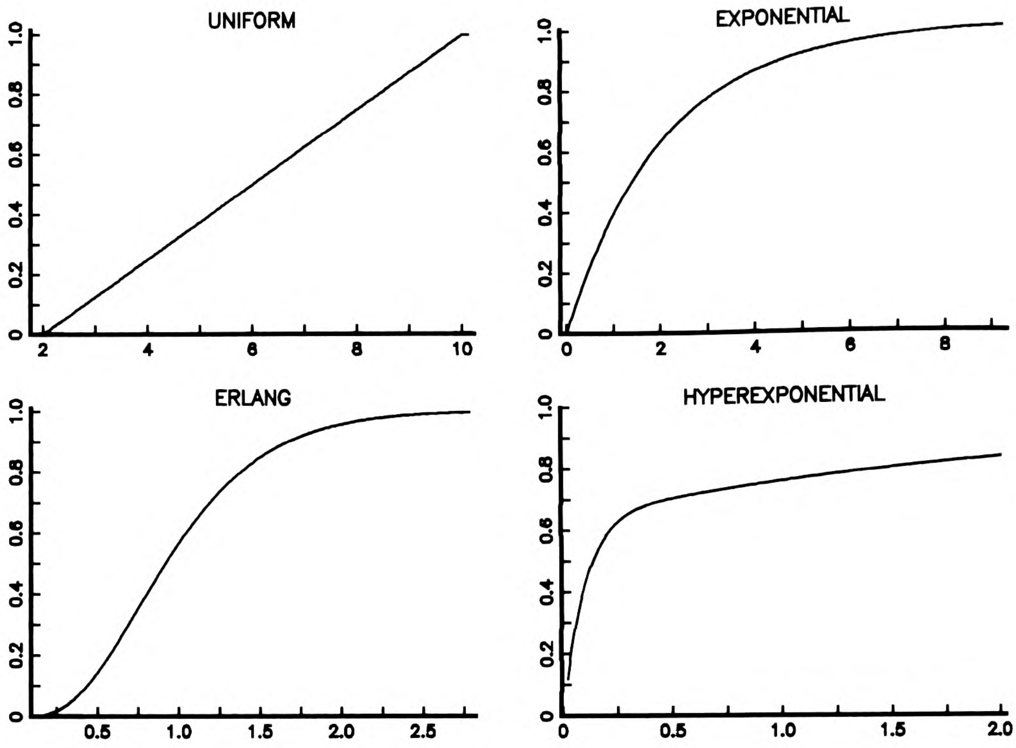


Figure 2.4. Probability Distribution Functions

Often the analyst will speculate on the shape of the PDF/density function and then estimate the parameters. For example, one might hypothesize that the distribution can be represented as uniform, and then estimate the end points of the interval $[a,b]$. Rather than directly estimating the defining parameters of the distribution, an analyst will usually estimate the mean (average) value of the distribution and the variability of the distribution. (The variability can be determined either as the variance, the standard deviation, or the coefficient of variation. Assuming the mean and one of these values is known, the other two can be directly obtained.) These are the values usually required by modeling packages. For example, for the uniform distribution, the mean is given by

$$\bar{x} = \frac{a+b}{2}$$

the variance is

$$\sigma^2 = \frac{(b-a)^2}{12}$$

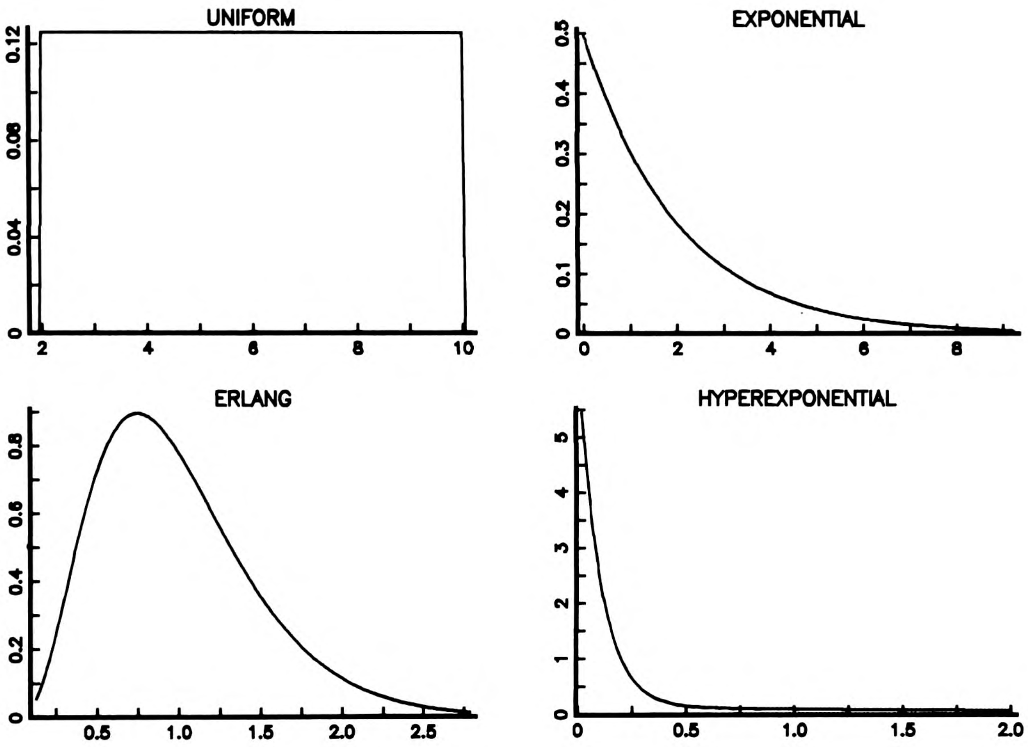


Figure 2.5. Probability Density Functions

the standard deviation is

$$\sigma = \frac{b-a}{2\sqrt{3}}$$

and the coefficient of variation is

$$C = \frac{b-a}{(b+a)\sqrt{3}}$$

The types of distributions used in a model to represent arrivals, service, and other probabilistic decisions can have a significant effect on the results of a model. There are many sophisticated mathematical techniques that can be used to determine which distributions are appropriate. A thorough discussion is beyond the scope of this book. Section 2.11 lists several books which cover this topic extensively.

2.5. SCHEDULING ALGORITHMS

The scheduling algorithm used to decide which customer to place in service next is frequently referred to as the queueing discipline. Some common queueing disciplines used in models include first-come-first-served (FCFS), last-come-first-served (LCFS), processor sharing (PS), infinite server (IS), nonpreemptive priority (PRTY), and preemptive-resume priority (PRTYPR). With FCFS scheduling, a customer is put into service in the order in which it arrives at the service center. LCFS is just a push-down stack. If a customer arrives when one is in service, the customer in service is taken out of service, and the new customer begins its service. When a customer completes, the customer at the top of the stack is put back in service.

In order to explain the processor sharing queueing discipline, we will first explain the round-robin (RR) scheduling algorithm. With round-robin scheduling, when a customer arrives at a service center it has a total service request. Each customer is permitted to execute for a small amount of time, which is usually less than the total service request. This execution time is called a quantum. If the service request is not completed after executing for a quantum, the customer is placed at the end of the queue and the next customer begins its quantum. Eventually the customer will complete and leave the resource. If we reduce the quantum to zero, all customers at the resource will be served in parallel. This is exactly what happens in the case of processor sharing. If there are n customers at the resource, each customer receives $1/n$th of the processing power. Although the processor sharing discipline does not occur in real systems, it is often used as a good approximation of round-robin scheduling. Round-robin scheduling is very common in a computer system at a CPU.

At an infinite server service center, there is never any waiting. A customer begins its service immediately upon arrival. All customers present at the service center are served in parallel. This type of service center is useful in representing a delay where there is no waiting, as would be the case for think times at a cluster of terminals from which transactions are being generated.

Priority scheduling is used to give preference to certain types of customers. With nonpreemptive priority a customer which is in service is not preempted if a higher priority customer arrives. With preemptive-resume priority a higher priority customer will preempt a lower priority customer. The preempted customer will resume its service after all higher priority customers complete their service.

2.6. MODEL PARAMETERS

The models we build will contain parameters which must be given values before we can calculate the performance measures we are interested in. These parameters could include the amount and distribution of service demanded at each resource, the scheduling algorithms, the average number of times a customer visits a device, the number of customers in a closed system, and the interarrival time distributions for an open system. Some of these parameters are a characterization of the workload on the system. If the workload varies while the system is operating, the analyst may have to decide what operating period to use for determining the associated model parameters. The parameters can be estimated from our knowledge of the system, our intuition, or from measurements on the system. In estimating values for these parameters, we must be aware of the possibility of introducing errors which could result in inaccurate results.

Frequently, models are evaluated for many different sets of parameter values. One set of values might represent a system as it currently exists. A different set of parameters might represent changes in the workload, a different configuration, or different equipment. Predicting the performance of a system by solving a model with different parameters is called a modification analysis. Determining what parameter values to use is often a difficult task which requires a great deal of skill by the analyst. Some changes are simple to represent, like doubling the speed of a processor. Others are very difficult to categorize. As an example, how would a different operating system affect the behavior of interactive users at terminals?

The most important skills that an analyst needs are an understanding of how a system behaves, a means of estimating the initial set of model parameters, and the ability to perform a modification analysis accurately. It is not necessary to understand the mathematical analysis which is used to solve models. This knowledge is incorporated into readily available modeling packages. Many people are capable of performing excellent modeling studies without having a sophisticated mathematical background.

2.7. MODEL SOLUTION

Using the types of models we have described, we must choose one of the available solution techniques. We will be discussing two possible solution techniques. An analytic solution will involve solving some equations which relate the model parameters to the performance measures. Simulation is a statistical experiment which observes the behavior of the model and generates the performance measures from the observations. We will have much more to say about these two solution techniques in Chapters 4 and 5.

At this time, we want to describe briefly how we would choose between them. An analytic approach is usually a faster solution method and is preferable when it is applicable. The problem is that many simplifying assumptions must be made in order to be able to solve a model analytically. Simulation is much more general and can be applied to very complex situations. The price which must be paid for this generality is the longer time required to obtain accurate performance measures.

We will be interested in several different performance measures calculated by the solution technique. The *utilization* of a resource is the fraction of time a server is busy. The *queue length* is the number of customers either waiting or in service. The *throughput* is the customer completion rate. It is the number of customers which complete their service in a given unit of time. The *queueing time* is the time a customer spends in the waiting line and in service at a center. We will be interested in mean values for these results and sometimes in the distributions of the queue length and the queueing time. Chapter 7 will discuss these performance measures in more depth.

When we build a model, we should structure the model in a logical fashion. A good approach is to begin with a very simple and high-level model with very few details. As we progress, we can add more of the details that exist in the real system by expanding the resources which exist in the higher level model. This can be done by defining submodels that contain more realistic representations of the additional complexities and using these submodels in place of the simpler representations. In this way, the model can be constructed in a top-down manner and can be refined to any level of detail that we desire. The structure of models will be discussed in more detail in Chapter 6.

Even after we decide to construct a detailed simulation model, it is a good idea to have a simpler model which can be solved analytically and which will be an approximation of the more complicated model. The analytic model will provide a means of partially assuring the simulation model is working correctly. In addition, if it turns out that the analytic model is a close enough approximation for the simulation model, the analytic model may suffice. The analytic model could also be useful in obtaining gross estimates of the performance measures over a large parameter space, and then the simulation model could be used to obtain more accurate results for a subset of the parameter values.

2.8. COMPUTER SYSTEM

Many computer system models contain identical submodels. Almost every model of a computer system contains a submodel representing the CPU and the I/O devices. The central server model depicted in Figure 2.6 shows a CPU and an arbitrary number of I/O devices. The central server model is a good representation of a batch workload under heavy load. If there is always a new batch customer to replace one that finishes, this type of model gives good results. This is a closed model. The number of customers in the batch workload is equal to the multiprogramming level.

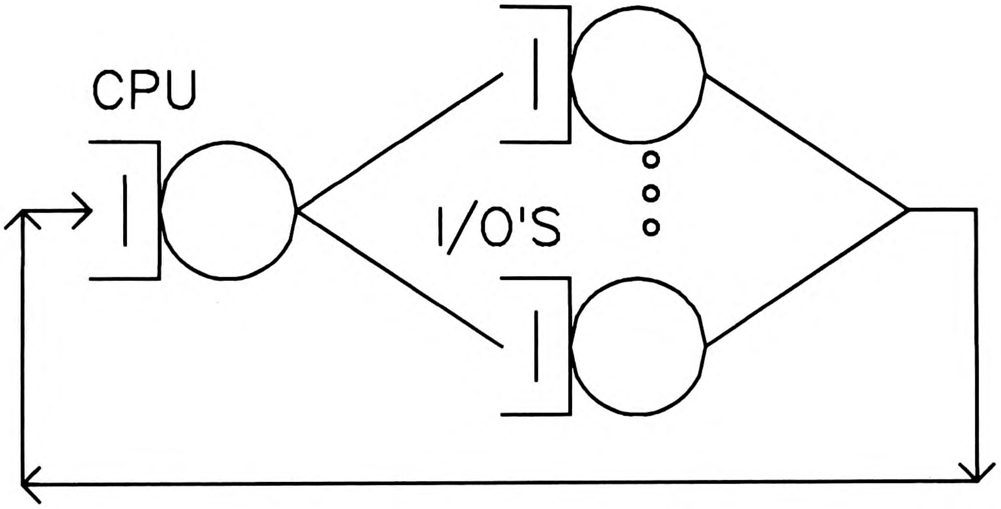


Figure 2.6. Central Server Model

Many computer systems contain other types of workloads in addition to batch workloads. An interactive workload can be modeled as a closed model with an infinite service center for the terminals. The number of customers in the model will be equal to the number of terminals. In an MVS type of system, TSO workloads can be represented as interactive workloads. Figure 2.7 illustrates an interactive workload. It contains the terminals, a passive resource representing a memory constraint, and a central server submodel.

One other type of workload which appears in many computer systems is called a transaction processing workload. Data-base systems like IMS and CICS fall into this category. Figure 2.8 shows a transaction processing workload. The transactions arrive according to some arrival distribution.

There is a passive resource for the memory constraint, and a central server subsystem for the CPU and I/O devices.

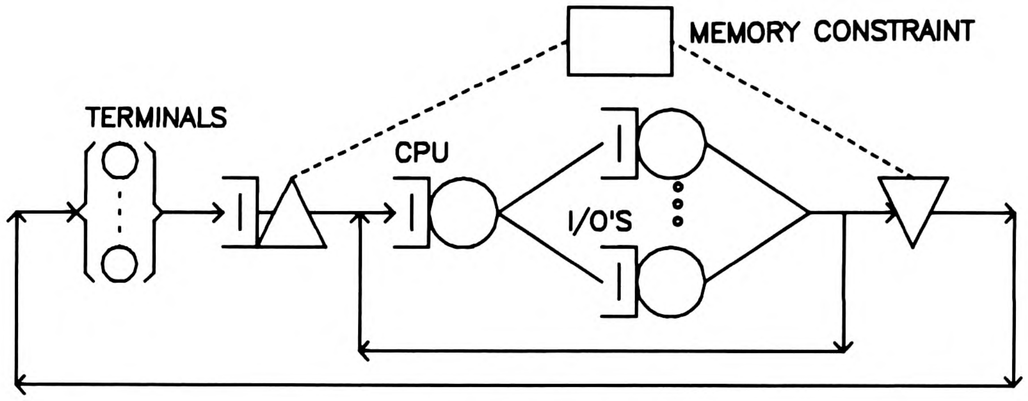


Figure 2.7. Interactive Workload

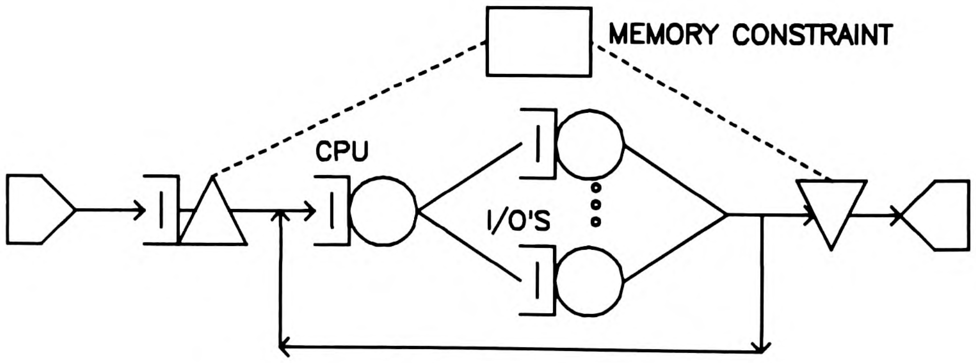


Figure 2.8. Transaction Processing Workload

Most computer systems contain a number of different types of workloads. There might be two different types of interactive workloads, a batch workload and a transaction processing workload. These can be easily included in the model by combining the models described previously. A model of many different workloads requires more input parameter values but gives more detailed performance measures.

2.9. COMMUNICATION NETWORK

Most models of communication networks contain at least one queue representing a communication line. In this section we will introduce a very simple model with a single half duplex line. A half duplex line can only communicate in one direction at a time. This will be modeled as a single resource with one path for inbound messages and another path for outbound messages. There is a single server which can transmit either inbound or outbound messages. Figure 2.9 shows the half duplex line with some terminals and a computer system. The terminals are depicted by an infinite server. As soon as a response returns over the outbound line from the computer, a user at a terminal begins his or her next think time. When a transaction is entered, it is transmitted over the inbound line, performs some processing at the computer, and sends a response back over the outbound line. There is contention at the line from messages being entered from multiple terminals. There is also contention between inbound and outbound messages. The rate of service at the line is the transmission rate in the system. Messages of different sizes will require the use of the line for different lengths of time.

2.10. MANUFACTURING SYSTEM

Generally, manufacturing systems have machines that break down and require repair. Figure 2.10 shows a simple model of a tool which experiences breakdowns. Arriving customers are either transferred to a tool for processing or bypass the tool and are transferred to the next operation. Jobs which complete processing at the tool also go through a transfer unit on the way out. When the tool breaks down, customers queue up waiting for the tool to be repaired. The tool failure can be represented by another customer which circulates between the tool and another queue. When this special customer is not at the tool, the tool is operating normally. When the special customer arrives at the tool, a failure occurs. The special customer takes control of the tool because it is assigned a higher priority than the normal customers at the tool. The amount of time the special customer spends at the tool represents the downtime. The amount of time the special customer

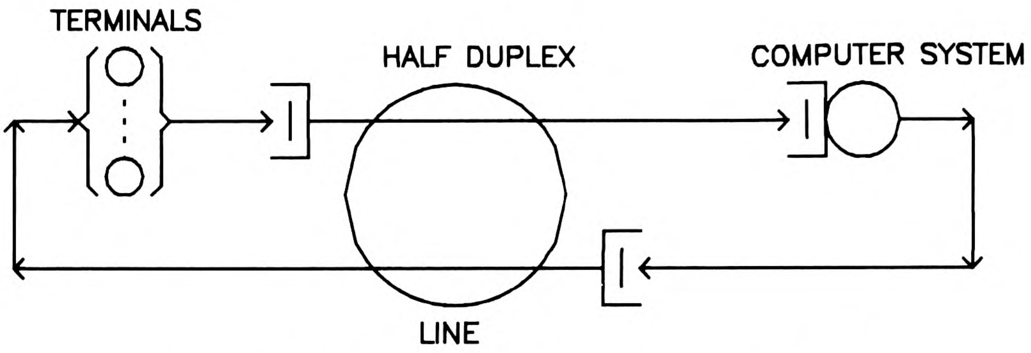


Figure 2.9. Model of a Simple Communication Network

spends at the other queue is the uptime. This simple type of model very nicely captures the tool failures.

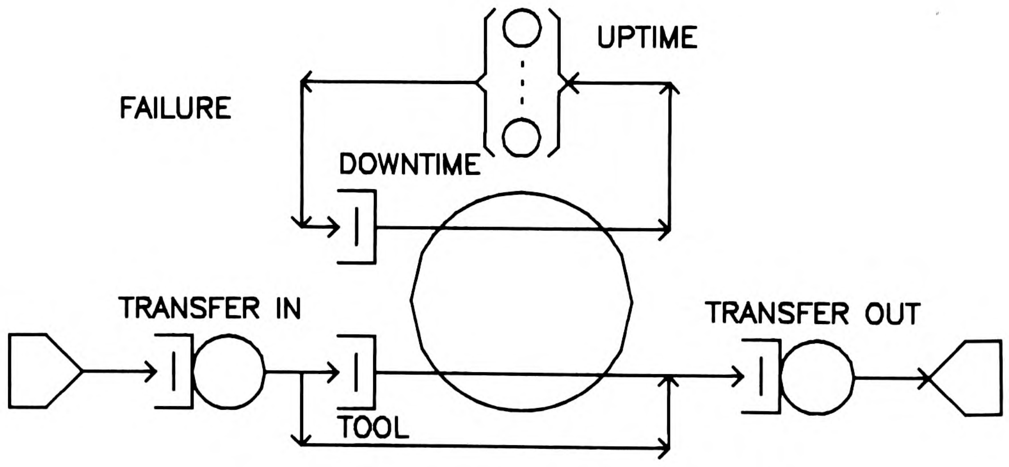


Figure 2.10. Model of a Manufacturing System

2.11. FURTHER READING

The process of modeling is introduced by Kobayashi [98], Lavenberg [100], Lazowska, Zahorjan, Graham, and Sevcik [108], and Sauer and Chandy [152]. Petri Nets provide a modeling framework which is different from the queueing networks we have discussed. The following references contain information about using Petri Net models: Balbo, Marsan, Ciardo, and Conte [8], Garg [68], and Peterson [132]. More information on probability distributions can be found in Allen [3], Cramer [53], Feller [60], Chapter 2 of Lavenberg [100], Mood and Graybill [124], Parzen [130], Trivedi [183], and other books on probability and statistics. See Buzen [40] for a good description of the skills necessary to carry out a performance analysis. More details about scheduling algorithms can be found in Kleinrock [95] and Sauer and Chandy [152]. There are many computer system models in Allen [3], Ferrari [62], Lavenberg [100], Lazowska, Zahorjan, Graham, and Sevcik [108], Sauer and Chandy [152], and Trivedi [183]. Communication network models can be found in Bharath-Kumar and Kermani [21], Kleinrock [96], Sauer and MacNair [156], and Schwartz [165,166]. The following references contain some manufacturing models: Law and Kelton [106], Medeiros and Sadowski [121], Oates [129], and Taylor and Clayton [180].

2.12. EXERCISES

2.1 Discuss the differences between an open model and a closed model.

2.2 Describe some features in systems you are familiar with that can be represented by a passive resource.

2.3 Obtain some measurement data from a system representing service times or interarrival times. Determine the sample mean, sample standard deviation, and the coefficient of variation of these values.

2.4 Plot a histogram for the following data and determine what distribution it might have come from: 4.06, 0.56, 1.56, 1.26, 3.04, 6.11, 0.775, 0.773, 0.135, 1.92, 1.31, 0.37, 6.73, 5.86, 1.27, 0.798, 9.73, 1.92, 5.41, 1.75, 0.752, 1.06, 0.144, 0.334, 1.28, 4.77, 0.85, 1.75, 0.71, 0.188.

2.5 Draw a model diagram, representing in a simplistic fashion the flow of work through a system you are familiar with.

CHAPTER 3

MODEL ELEMENTS AND DIAGRAMS

A model diagram is essential to understanding how the model represents the system to be studied. It is an exact, unambiguous representation of the order in which the resources are visited by the customers. It is an excellent means of communicating what the model is. We will discuss a relatively small set of diagram symbols and the corresponding model elements which are fairly widely accepted for capturing the flow of work through many different types of models. Some modeling tools employ a large collection of symbols for use in diagrams. The advantage of having a small set of symbols is the simplicity of learning them and using them in diagrams. The disadvantage is that some situations may not be explicitly represented or some information may be missing from the diagram. Even with the small set of symbols we will discuss, the diagrams will be able to represent most aspects of the models.

The model elements are the building blocks of a modeling tool. Once we understand what the model elements are and how to use them, it becomes a relatively simple task to construct a model of a complex system if we understand how the system operates. We just need to know which elements to use and in which order to use them. By putting the building blocks together in different fashions, we can construct many different models.

3.1. CUSTOMERS

In building models we will focus our attention on the customers that are circulating through the model and demanding service from the resources. The customers can represent many different kinds of entities. They can be people, computer programs or jobs, communication messages, acknowledgements or polling responses, manufacturing tasks or parts to be assembled, among other items found in systems.

We will not explicitly draw the customers in the model diagrams. However, all the resources they visit will be shown along with the paths that the customers follow. The paths followed by customers are depicted by solid lines and arrows. Each place a customer visits in a model is called a *node*. There are several different kinds of nodes which represent various kinds of actions performed when a customer arrives at each node. The different types of nodes are discussed in the remaining sections of this chapter. The collection of nodes visited by a customer and the order in which they are visited is referred to as the routing.

The customers can have different attributes which distinguish different kinds of customers and could represent the amounts of service demanded. Some examples of attributes of customers include the type of job, for example, interactive or batch in a computer system, the message length, the path length, the priority level, the number of times the customer should visit a portion of the model, the time of arrival at or departure from a particular node, and many other identifying characteristics. The attributes are attached to the customers and can be interrogated while making routing decisions or when determining how much service a customer demands.

There are certain instances when we want the customers to make copies of themselves, with the original customer and the copies possibly proceeding over different paths. If we want 100 pieces of a subassembly to arrive at a service center all at the same time, we can have one customer split itself into 100 separate customers which then progress separately through the model. The copies which are produced can be independent customers which follow different paths, or they can be related to each other and join back together again at an appropriate place in the model. An example of related customers is found in a communication network where large messages are broken down into smaller packets. The packets are transmitted over the network and reassembled after transmission.

There are different symbols associated with the generation of the copies of customers depending on whether the customers are related or not. There is also a symbol for a node where the related customers are reassembled. Figure 3.1 shows the symbols for split, fission, and fusion nodes. Customers passing through a split node generate unrelated customers. The fission and fusion nodes are used in pairs. The copies of customers generated at a fission node are joined together at a corresponding fusion node. Only a single customer leaves the fusion node. This occurs after all of the related customers arrive.

3.2. SERVICE CENTERS

Service centers are the major model elements used in extended queueing networks. They are composed of one or more servers, one or more queues, and a queueing discipline or scheduling algorithm for determining which customer to put into service next. The customers arrive at the service centers and request a certain amount of service. This service is usually

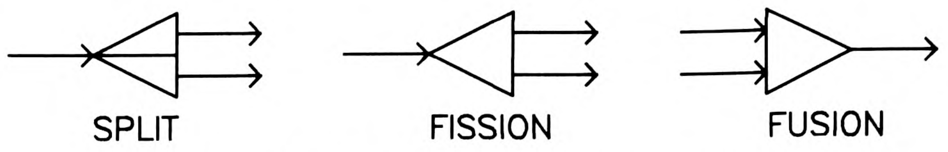


Figure 3.1. Split, Fission and Fusion Nodes

determined by a service time distribution specified when defining the service center.

Figure 3.2 shows a service center with a single server, one with two servers, and one with an infinite number of servers. A circle is used to represent a server at an active resource. The queues, which are shown as a rectangle with one side missing and a vertical line in the middle, are also called classes. Some service centers have more than one queue. Several reasons for having multiple queues at a service center are to specify different service time distributions, different priority levels, and alternate routing paths. The classes are the nodes at service centers which are used in the routing definition.

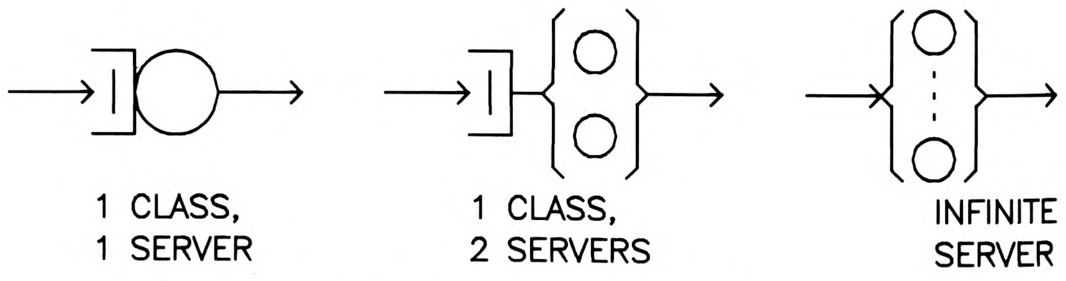


Figure 3.2. Three Service Centers

Since there is no waiting at an infinite server, there are no queues shown with the symbol of the IS center. There can still be multiple classes at an infinite server so that customers can have various routing paths.

When a customer arrives at a service center other than an infinite server, the customer waits in the queue until a server is free. When a server is available, customers are scheduled according to the queueing discipline. Recall that some of the queueing disciplines include FCFS, LCFS, processor sharing, round robin, preemptive priority, and nonpreemptive priority. The queueing discipline determines whether the service may be preempted by other jobs arriving at the service center or whether the server is shared among the customers. A customer's activity is usually focused on the resources of a service center and typically has no interaction with other modeling elements while at a service center.

When a customer is put into service, the amount of service requested can be specified in two different ways. The first approach is to specify a service time which is the amount of time spent in service during a single visit to the service center. The second method is to specify an amount called the work demand and a rate of service. Examples of work demands are the number of instructions to process and the message length. The service time is then calculated as the work demand divided by the service rate. The rate of service is the amount of work a server can perform in one unit of time. When the servers work at the same fixed rate, the service rate can be set equal to one. In this case, the service time is equal to the work demand. The service rate can be thought of as a scaling factor.

The amount of service requested is normally specified by a distribution. With simulation, a random sample is taken from the distribution to determine the amount of time each individual customer will spend in service. The service time can also be obtained from a file containing trace data. With an analytic solution, only the mean service time will affect the results, which can be calculated using a queueing network model with standard numerical solutions. This will be discussed in more detail in Chapter 4. Some of the commonly used service time distributions are constant, discrete, uniform, Erlang, exponential, hyperexponential, and normal. The different distributions represent different patterns of service.

3.3. PASSIVE CENTERS

Given only customers and service centers as model elements, there are many situations which exist in real systems which are difficult or impossible to represent accurately. Passive centers permit us to represent many of these complex features.

3.3.1. Allocate and Release

Passive centers are mainly used to model a resource that has a limited number of elements that are allocated to customers, held on to by the customers while they receive service at service centers and then released by the customers. The major difference between service centers and passive centers is that a service center has one or more servers actively engaged in providing service to customers. This is not the case at a passive resource. There are no servers actively providing service. However, there are elements called tokens which are in some ways analogous to servers. There is usually a limited number of tokens at the passive center. These tokens can be used to represent a finite number of elements of a resource like the number of buffers, memory units, channels, and other limited resources.

As an example of a passive center, we will consider how to represent memory contention in a computer system. Figure 3.3 depicts a passive center with the number of tokens, which is shown in the box, being equal to the number of memory partitions. The rectangular box is the pool of tokens from which the memory partitions are requested. AL1 is an allocate node where customers request tokens. If the number of tokens remaining in the pool is less than the number of tokens a customer requests, the customer waits in the queue associated with the allocate node until a sufficient number become available. It is important to remember that customers retain possession of the tokens until they are explicitly released. Tokens are returned to the pool when a customer which is holding tokens from the passive resource passes through release node RE1. As usual, the customer flow is shown with solid lines and arrows. The flow of tokens is illustrated with dashed lines.

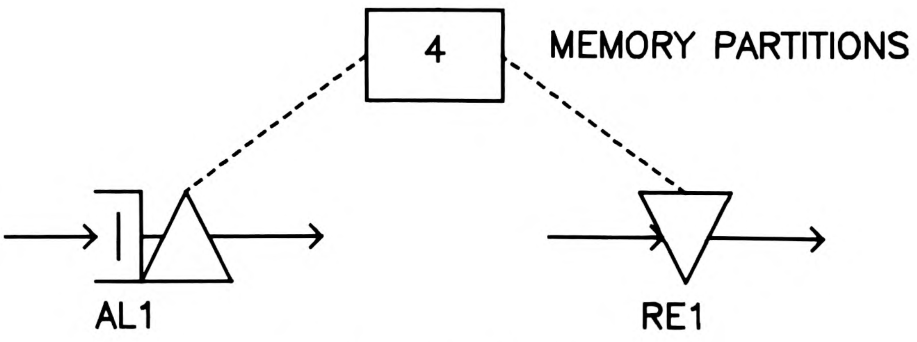


Figure 3.3. Allocate and Release Nodes

The passive center facilitates the representation of many situations where customers simultaneously hold multiple resources. A customer which acquires tokens from a passive center can also request service at service centers and can also request tokens from other passive centers. This type of model element is a very powerful extension to conventional queueing networks.

In Figure 3.3 we drew one allocate node and one release node. However, there is no restriction on the number of allocate or release nodes which belong to a passive center. There is also no one-to-one correspondence necessary between allocate and release nodes. There can be any number of allocate nodes and any number of release nodes.

3.3.2. Create and Destroy

With only allocate and release nodes, there is no way to change the number of tokens at a passive center. There are times when we would like to increase or decrease the number of tokens. This can be accomplished as shown in Figure 3.4 using create and destroy nodes. A customer which goes through a create node will add a specified number of new tokens to the pool, and a customer holding tokens when it is routed through a destroy node will discard the tokens it holds. This permits the number of tokens associated with a passive center to change dynamically. One use for these model elements is to hold customers in the queue at an allocate node until another customer creates tokens for them to advance. This is a type of synchronization which is very common in contention systems. These model elements can also be used for communicating between independent processes. In an operating system, this is often referred to as a semaphore.

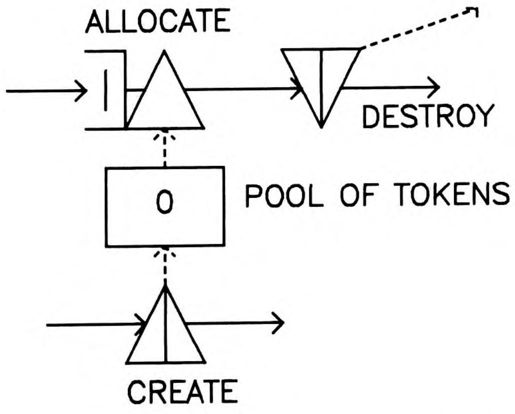


Figure 3.4. Create and Destroy Nodes

3.3.3. AND/OR Allocate

It is sometimes necessary for customers to contend for elements of several passive centers at the same time. Special types of allocate nodes provide the model elements to handle this situation. AND allocate nodes belong to multiple passive centers. A customer which arrives at an AND allocate node requests tokens from all the passive centers it is a member of. The customer waits in the queue until all of its demands can be met. Similarly, OR allocate nodes belong to multiple passive centers. A customer which is routed to an OR allocate node is allocated tokens from the first passive center which has a sufficient number of tokens to satisfy the number demanded. This is illustrated in Figure 3.5.

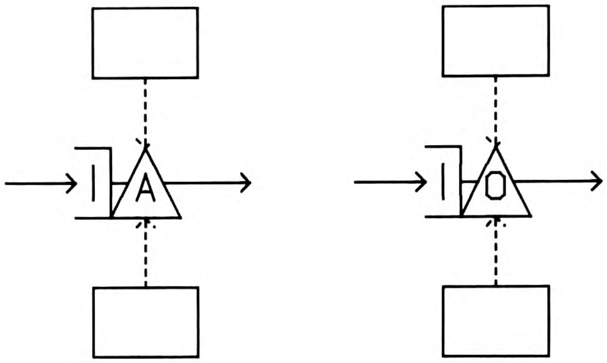


Figure 3.5. AND Allocate and OR Allocate Nodes

Passive centers provide very high-level model elements to depict many complexities found in real systems. In addition to representing contention for finite resources, they can be used in modeling complicated flow control algorithms like polling and pacing found in communication networks. We will see many examples where passive centers are very useful.

3.4. SOURCES AND SINKS

In open-path models where customers enter from outside the model, we need places where customers are generated and locations where customers depart. These nodes are called sources and sinks. Associated with every source there is an interarrival time distribution. This distribution determines how frequently customers arrive at the model and according to what type of pattern. The exponential distribution is often used when there is no information available to suggest some other distribution. Customers arriving at a source circulate through the model and eventually leave at a sink.

In building models an analyst must determine whether to use paths which are open or closed. In choosing an open path, we are assuming that the customers are being generated from an infinite population. The number of customers in the model at any given time will vary from zero up to any value, but we will be able to calculate an average customer population.

Figure 3.6 shows an open model with a source, a service center with one server and a sink. If we choose an exponential arrival distribution and an exponential service distribution, this model corresponds to an M/M/1 queue. This type of service center is described in many books on queueing theory.

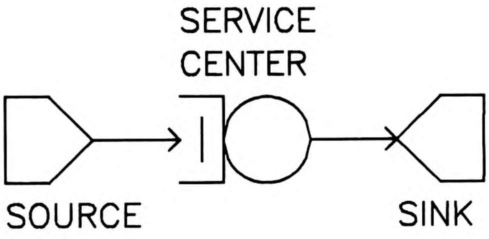


Figure 3.6. Open Model with Source and Sink

Open models can have any number of sources and conceptually any number of sinks. Since a sink is just a place for customers to leave the model, we need only one sink in the entire model. All the customers which leave will depart through the same sink. The reason for having multiple sources is to allow customers to be routed to different nodes after arrival.

3.5. MODEL VARIABLES AND STATUS

In order to make certain types of decisions, we will use two different types of variables. One type of variable is used to hold the set of attributes of each customer. Each customer has its own set of attributes which can be set to different values and used to make routing decisions and to determine the amount of service requested and the customer's priority, among other purposes. The other type of variable is accessible by all customers. This type of variable is a global variable in the sense of a variable in a programming language. As customers proceed through the model, these global variables can be assigned values and used in ways similar to the customer attributes.

All customer attributes are automatically initialized to a value of zero, and global variables are initialized to a specified value when they are defined. They must be explicitly assigned other values as the model is progressing. In order to assign values to the customer attributes and the global variables, we need a special kind of node which we designate as a set node. As customers pass through a set node, one or more assignments are made to the values of the customer's attributes or to the global variables. The symbol for a set node is a rectangular box as shown in Figure 3.7. When there is room, the assignment statements are given in the rectangle. The assignment statements can contain expressions. In this case the global variable V is being incremented by one.

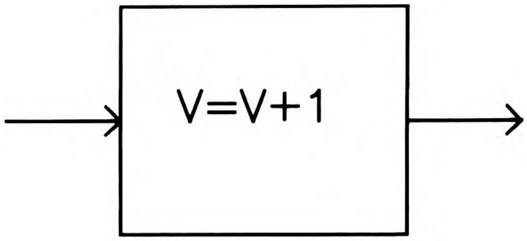


Figure 3.7. A Set Node

In addition to making decisions based on the values of variables, we can also interrogate the status of various conditions of the model. Some of these conditions include the queue lengths at classes, service centers, allocate nodes and passive centers, the number of servers or tokens available for service or allocation, the number of customers related to a customer which has gone through a fission node, and the number of tokens a customer holds from a passive center. These few conditions will provide most of the status of the model necessary to control the flow of customers through the network.

Figure 3.8 illustrates some routing decisions being made based on the value of a global variable. The variable V is incremented by one, and its value is tested. If V is equal to ten, the customer is sent to Q2 for service. If V is less than ten, the customer joins the end of Q1 again. By initializing V to zero, this approach can be used to send customers to a service center a specified number of times. As mentioned above, these routing decisions could be based on the model status as well as the values of variables.

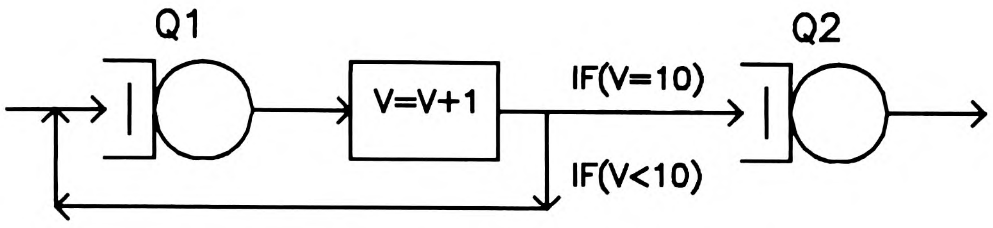


Figure 3.8. Routing Decisions and Set Nodes

3.6. WAIT UNTIL

There are times when customers are held at a particular location in a system until a certain condition occurs. The wait node provides this type of capability. Customers are placed in a queue and wait there until a specified condition occurs. When the condition becomes true, all waiting customers for which the condition is true proceed from this node.

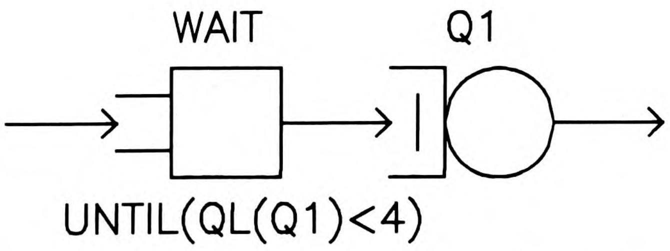


Figure 3.9. Wait Node and Active Service Center

A wait node is a rectangular box with a queue in front of it. In Figure 3.9, customers join the queue at the wait node and wait until the queue length at Q1 is less than four. The conditions tested at a wait node can include any of the model status tests discussed in Section 3.5 and the values of customer attributes and global variables.

3.7. CHAINS

Chains are used to classify different types of customers into different routing paths. A path consists of all the nodes visited by a customer. When a model contains different types of customers, it is convenient to define separate chains for each customer type. Different types of customers may belong to the same chain. In this case, the customer attributes can be used to identify the customer type, and routing decisions can send customers to different resources.

By using different chains for different types of customers, the model does not have explicity to check the customer type. This is implicit in the chain to which the customer is assigned. Customers and nodes are uniquely assigned to one chain. A customer in one chain can never visit a node which belongs to a different chain, and a node which belongs to one chain cannot be used in a routing statement of another chain. This does not preclude customers in different chains from contending with one another. A center representing either an active or passive resource can have nodes belonging to different chains. The customers belonging to the different chains can still contend for the same servers or tokens.

A standard example of using multiple chains in a model is a computer system that has two different workloads. One type of work is from interactive terminals where people are submitting transactions and receiving responses. A second type of work is batch jobs. Figure 3.10 shows two chains for the two different workloads. The interactive jobs travel over a closed chain visiting the terminals, the CPU, and an input/output (I/O) device. The batch jobs arrive from a source of an open chain and contend with the interactive jobs at the CPU and I/O devices.

In Chapter 2 we briefly discussed open, closed, and mixed models. The types of chains in a model determine the type of model. An open chain usually contains one or more sources where customers enter the chain and a sink where customers depart. It is possible for an open chain to contain no source. In this case, customers would have to be initialized at one or more nodes of the chain. This approach is used when customers produce copies of themselves by going through a split node and the copies eventually are routed to the sink. The original customers can continue to circulate through

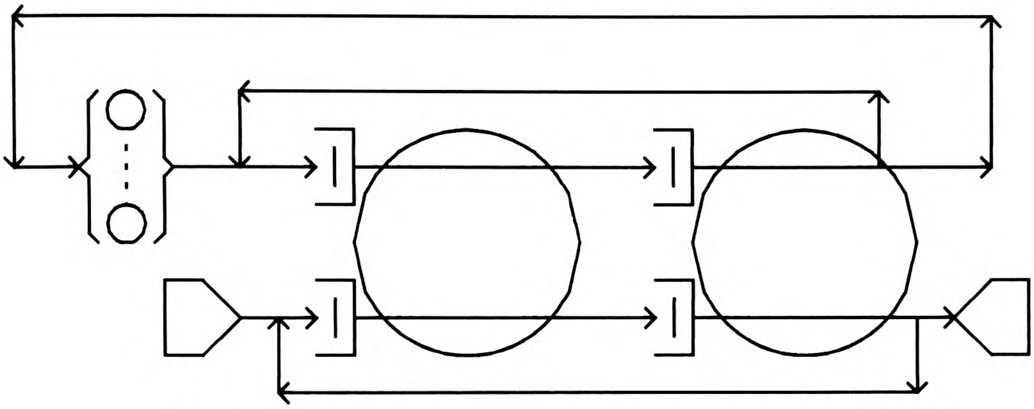


Figure 3.10. A Model with a Closed and an Open Chain

the chain. Open chains permit the number of customers in the chain to vary. There are customers arriving and departing at various times, and the number of customers present is continually changing. An open chain is frequently used to model a system where the population is not static.

A closed chain contains a fixed number of customers. We specify the chain population, and these customers always remain in the chain. A system that has a finite number of customers, like an interactive computer system with 50 terminals, is conveniently represented by a closed chain. When we can identify a relatively small, fixed number of customers in a chain, a closed chain is the appropriate model element to use.

3.8. SUBMODELS

A submodel is a subset of the model's resources which are separated from the rest of the model either to add structure and clarity or to solve the submodel in isolation. We will not discuss submodels to any great extent here because they will be described in detail in Chapter 6. Submodels do appear on diagrams in the form of rectangular boxes drawn with dashed lines. Figure 3.11 shows a submodel in the top part of the figure and the resources which belong to the submodel below it.

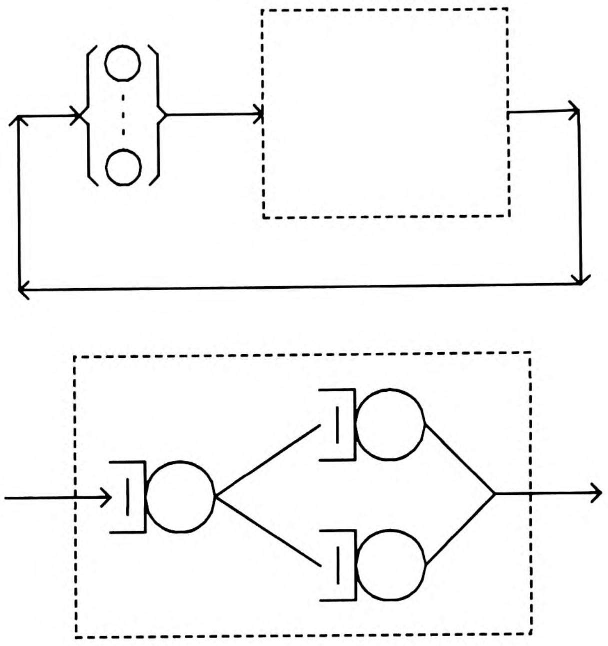


Figure 3.11. A Model with a Submodel

3.9. NONSTRUCTURAL INFORMATION

The model diagrams we have been discussing represent the flow of customers visiting the resources. However, there is some information which is pertinent to the model which does not appear explicitly in the diagrams. Since the customers are not shown on the diagram, their attributes are also not displayed. The assignments made to the attributes are given at set nodes, and the routing decisions based on the values of the attributes are illustrated. In a similar fashion, global variables are not depicted in the diagrams, but their assignments and routing decisions based on their values are shown.

When there are different types of customers with different priorities, very often the priority levels are not shown on the diagram. When this information is crucial to understanding the model, it can be represented. Sometimes when customers visit a service center with many different classes, it may be difficult to depict the fact that the different classes belong to the same service center. The mapping of the classes onto the service centers is important, but is not always necessary for the model diagram.

The tokens allocated to customers are held by the customers until they are explicitly released. Since the tokens can be allocated according to a distribution, the number of tokens a customer holds is not shown. Interarrival time and service time distributions are also not displayed in diagrams.

Most of the information which is omitted from the diagrams is left out to reduce the complexity of the picture. Even with this information missing from the diagrams, we can still get a good picture of the flow of work through the system. This is the main purpose of the diagrams.

3.10. SAMPLE DIAGRAMS

We will illustrate most of the model elements which have just been discussed in several diagrams in this section. Many more diagrams will be presented as various models are discussed throughout the remainder of the book.

The first diagram, Figure 3.12, depicts an interactive computer system with memory contention. Users at the terminals submit jobs to the computer system. The terminals are represented by an infinite server. The jobs wait in a queue at the allocate node if all four memory partitions are in use. A memory partition could correspond to a region in some operating systems. When a memory partition is available, it is allocated to a job which then receives service at the CPU followed by one of the I/O devices. With a certain probability the job cycles back through the CPU and I/O subsystem. Eventually the job completes, releases its memory partition, and returns to the terminal. Notice that a job simultaneously holds onto a memory partition and receives service at an active service center when it is in the CPU and I/O subsystem.

The next diagram, Figure 3.13, illustrates the use of fission and fusion nodes for producing copies of customers. A customer which goes through the fission node makes a related copy of itself. The original customer goes to the CPU and the copy goes to one of the I/O devices. This permits the simultaneous execution of the customer on two different devices at the same time if both devices are free. Any customer with a relative which arrives at the fusion node waits until its relative arrives. The original customer and the copy are reunited at the fusion node and just one customer proceeds.

Model diagrams give us a concise way of seeing the structure of a model. These diagrams aid in the construction of models when using

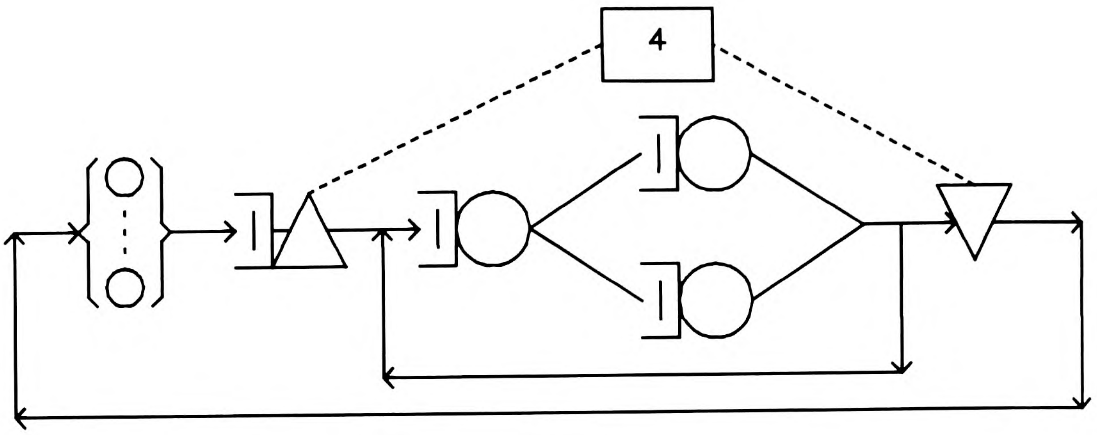


Figure 3.12. Interactive Computer System

modeling packages. In the next section we will briefly discuss some of the modeling packages which are commonly used.

3.11. MODELING PACKAGES

Modeling packages aid in the construction and solution of models. They impose some structure on the construction and simplify the accurate solution of a model. Most of the models we discuss could be solved without a modeling package. The analytic models could be solved by manipulating mathematical equations. Simulation models could be solved by writing a program in a higher level language or in a simulation language such as GASP [133], GPSS [70,71,164], SIMAN [131], SIMSCRIPT [91,146], or SLAM [135]. However, these approaches will normally be more costly in terms of the human effort necessary to produce a solution than the use of an appropriate modeling package.

Modeling packages fall into two categories. There are packages which are oriented toward the performance evaluation of specific systems. Packages of this type include BEST/1 [39,18,19,20], CMF [23], FIVE [128], MAP [137], PERFORMS [90], PET [21], SNAP/SHOT [176], the VM

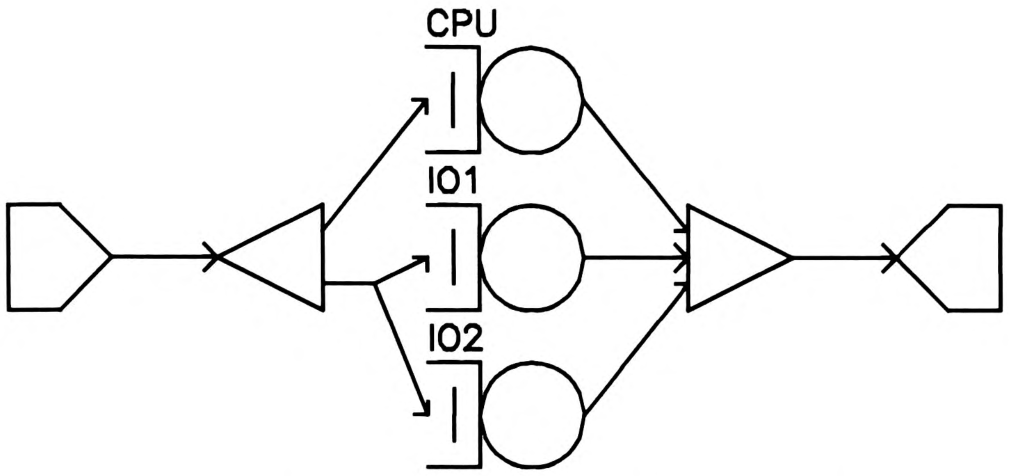


Figure 3.13. Overlap of CPU and I/O Processing

Predictor [11], and XL [29]. The systems modeled by these tools are mostly the large computer systems and their associated operating systems. Some modeling packages have data collection and analysis facilities. The formulation of the model is built into the tool, with the model parameters specified through a language related to the system being modeled.

The other category of modeling packages consists of general-purpose tools which can be used to model many different types of systems. This generality is an advantage in that a wide range of systems can be studied, but it requires that the model formulation be done by the analyst. General-purpose modeling packages include BORIS [116], CADS [83], COPE [15,16], Micronet [110], NUMAS [125], Panacea [119,120], PAWS [84,127], PNET [34], Q-GERT [134], QNA [187,188], QNAP [122,184], QNET4 [140], RESQ [114, 155-163], SCERT II [136], SNAP [24,25], STEP-1 [1], and Supernet [35]. These are the kinds of tools the remainder of the book is about. The discussion will concentrate on the formulation of models and the effective use of general-purpose modeling packages. Model elements and diagrams similar to the ones described in this chapter are used with these types of packages.

3.12. FURTHER READING

Most of the model elements and symbols which we discussed in this chapter are in common usage in the modeling literature and used with various modeling packages. RESQ makes extensive use of them, and the publications related to RESQ [114, 155–163] contain much more information about these model elements and diagrams.

Many papers, books, and reports have been written about modeling packages. Some of these have been mentioned in Section 3.10. Further discussions of some of these tools can be found in Reiser and Sauer [142] and Sauer and MacNair [154]. The books by Allen [3], Beizer [17], Ferrari [62], Kleinrock [95,96], Kobayashi [98], Lavenberg [100], Lazowska, Zahorjan, Graham, and Sevcik [108], Sauer and Chandy [152], and Trivedi [183] also provide information about the mathematical methods used in modeling.

3.13. EXERCISES

3.1 Use the symbols discussed in this chapter to draw a model diagram of a system you are familiar with.

3.2 The concept of bulk arrivals permits multiple jobs to arrive at a service center at the same simulated time. Take a sample from a distribution to determine the number of copies which should arrive simultaneously and produce this number of arrivals at an active service center. Draw a model diagram of a bulk arrivals system.

3.3 Draw a model diagram of a system with a finite capacity queue. Send the arrivals to the sink when the capacity is exceeded.

3.4 Draw a model diagram of a manufacturing system which contains a buffer where jobs wait for a robot to pick them up one at a time and move them to a tool subsystem for processing. Assume there are two robots and three tools in series.

3.5 Draw a model diagram of a simple street intersection with a traffic light and traffic flowing in one direction only. Make sure you represent the traffic light changing from red to green.

3.6 Draw a model diagram which explicitly represents round-robin scheduling.

3.7 Draw a model diagram of machine breakdowns and repair by a single repair person.

3.8 Draw a model diagram of a parking lot with one hundred spaces and send the cars to the sink when the parking lot is full.

3.9 Draw a model diagram of a full duplex line which is used to send messages between three different locations.

3.10 Draw a model diagram of a portion of a communication network which contains buffer contention and acknowledgements to permit the transmission of additional messages.

3.11 Draw a model diagram which contains multiple servers and picks one of the servers at random.

3.12 Draw a model diagram which sequences jobs to their original order of arrival after they come out of an infinite server. (Jobs coming out of an infinite server can be in a different order than the order of their arrival.)

CHAPTER 4

ANALYTIC SOLUTIONS

An analytic solution is conducted by solving equations which relate the model parameters to the performance measures. For example, if we are working with an open model and are given the routing probabilities, the mean interarrival time, and the mean service times of the resources, it is easy to calculate the utilization, throughput, mean queue length, and mean queueing time at each resource. The next two sections discuss the solutions of a simple open model and a simple closed model. The purpose of this explanation is to contrast analytic solutions with the method of simulation presented in Chapter 5.

4.1. SOLUTION OF AN OPEN MODEL

An open model contains at least one source where customers enter the model. By using a source, we are assuming that there is an infinite population from which we are producing customers. In an open model, there is also a sink where customers leave the model. Figure 4.1 is a diagram of an open model which might represent a simple computer system. After a customer is generated at the source, it visits the CPU. From the CPU, the customer goes to either the disk or the drum. After service at one of the I/O devices, the customer leaves through the sink.

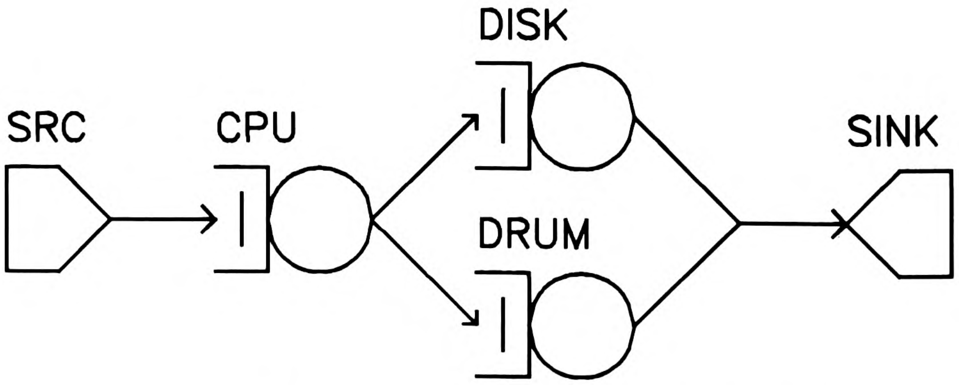


Figure 4.1. Open Model of a Simple Computer System

As a concrete representation of this model, we will present a version of it constructed using RESQ. RESQ uses some language which is slightly different from the corresponding terms found in some of the literature related to queueing networks. These differences will be identified as the model is presented. The first two items in the RESQ model are the model name and the method of solution. RESQ uses the term numerical for an analytic solution.

```
MODEL:EX4.1
   METHOD:numerical
```

The next section of the model contains the definition of each of the resources in the model. A service center is called a queue in RESQ. The type of queue used in this model is related to the scheduling algorithm or queueing discipline. For this simple system, the queueing discipline is first-come-first-serve. A RESQ class is a waiting at a service center. Each class has a service time associated with it and possibly a priority level. The service time given in this example is the mean of an exponential distribution. This is used to determine the amount of service requested each time a customer visits the service center. This model contains three service centers—the CPU, a disk, and a drum.

```
   QUEUE:cpuq
      TYPE:fcfs
      CLASS LIST:cpu
         SERVICE TIMES:.02
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.044
   QUEUE:drumq
      TYPE:fcfs
      CLASS LIST:drum
         SERVICE TIMES:.008
```

After defining the service centers, the RESQ model contains a definition of the path the customers follow. The path is called a chain in RESQ. Since this is an open model, there is a definition for the source where the customers enter the model. Associated with the source is an interarrival time distribution. Again we will use an exponential distribution for the interarrival times. RESQ uses the names of the classes to define which resources the customers visit. Here we see the customers go from the source SRC to the CPU. After completing service at the CPU, the customer goes to either the DISK or the DRUM. The routing decision is based on a probability. With a probability of 0.2, the customer will go to the DISK. With one minus this probability, the DRUM will be selected. After finishing service at one of the I/O devices, the customer will leave through the SINK.

```
    CHAIN:chn
       TYPE:open
       SOURCE LIST:src
          ARRIVAL TIMES:.0209
       :src->cpu->disk drum;.2 .8->sink
END
```

If we use RESQ to solve this model, the following performance measures are calculated:

```
    ELEMENT  UTILIZATION  THROUGHPUT  QUEUE LENGTH  QUEUEING TIME
    CPUQ       0.95694     47.84688     22.22211       0.46444
    DISKQ      0.42105      9.56938      0.72727       0.07600
    DRUMQ      0.30622     38.27750      0.44138       0.01153
```

These performance measures can be calculated by a modeling package very simply. The first part of the calculation involves finding the visit ratio which is the average number of times a customer visits a service center. Each customer makes one visit to the CPU, 0.2 visits to the disk, and 0.8 visits to the drum. For this model the visit ratios are equal to the routing probabilities. In general, it is necessary to solve a set of simultaneous linear equations relating the flow into and out of each resource. The reciprocal of the mean interarrival time is equal to the arrival rate (AR), and the throughput (TF) at each service center is equal to the arrival rate times the visit ratio (VR). The arrival rate is equal to 1 divided 0.0209 which is 47.84688. For this model, the throughput at the CPU is equal to

$$TP = AR \times VR = 47.84688 \times 1.0 = 47.84688.$$

The throughputs at the disk and drum are 47.84688 times 0.2 and 47.84688 times 0.8, which equal the numbers cited in the table.

The utilization (UT) is equal to the service time (ST) multiplied by the throughput (TP). At the CPU, it is

$$UT = ST \times TP = 0.02 \times 47.84688 = 0.95694.$$

The disk utilization is 0.044 times 9.56938 or 0.42105, and the drum utilization is 0.008 times 38.27750. This is equal to 0.30622. For the types of service centers present in this model, the queueing time (QT) is equal to the service time divided by one minus the utilization, and the queue length (QL) is equal to the throughput times the queueing time. This last formula is just Little's rule [112]. Performing these calculations for the CPU yields a queueing time of

$$QT = \frac{ST}{1 - UT} = \frac{0.02}{1 - 0.95694} = 0.46444$$

and a queue length of

$$QL = TP \times QT = 47.84688 \times 0.46444 = 22.22211$$

Similar calculations for the disk and the drum will produce the numbers shown above.

4.2. CLOSED MODEL

In a closed model, there is no source and no sink. There are a fixed number of customers which continue to circulate among the service centers. Figure 4.2 shows a diagram of the same system which was used in the previous section, but now it is represented as a closed system.

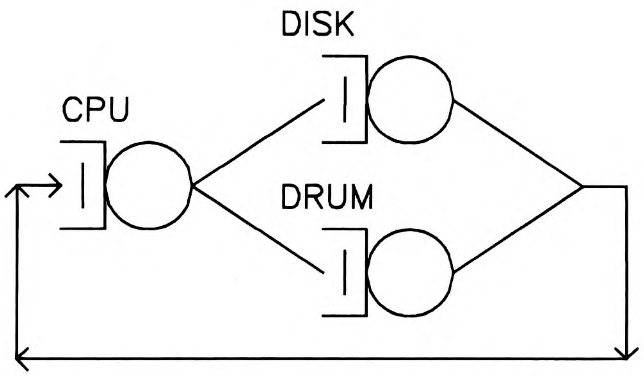


Figure 4.2. Closed Model of a Simple Computer System

The following is a listing of the model. It is exactly the same as the previous model, except when describing the type of chain. This is a closed chain with a chain population of four. This might represent a computer system where the multiprogramming level is four and the load on the system is very heavy. As soon as one job completes, another job replaces it so that the multiprogramming level never varies.

```
MODEL:EX4.2
   METHOD:numerical
   QUEUE:cpuq
      TYPE:fcfs
      CLASS LIST:cpu
         SERVICE TIMES:.02
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.044
```

```
    QUEUE:drumq
       TYPE:fcfs
       CLASS LIST:drum
          SERVICE TIMES:.008
    CHAIN:chn
       TYPE:closed
       POPULATION:4
       :cpu->disk drum;.2 .8->cpu
END
```

If we solve this model with RESQ, we will obtain the following performance measures. Notice that the utilization and throughput numbers are very close to the numbers obtained in the open model. This was done by design by specifying an arrival rate to the open model which would produce about the same throughput as the closed model with four customers. However, notice the difference in the queue lengths and queueing times. When we do not have an infinite population, these last two performance measures can be significantly different.

```
ELEMENT  UTILIZATION  THROUGHPUT  QUEUE LENGTH  QUEUEING TIME
CPUQ       0.95675     47.83765      2.91501       0.06094
DISKQ      0.42097      9.56753      0.66303       0.06930
DRUMQ      0.30616     38.27011      0.42196       0.01103
```

The discussion of how these performance measures are calculated will involve a very simplistic description of the Mean Value Analysis (MVA) algorithm [141]. When dealing with a closed model, there is no arrival rate. Therefore, the throughput is more difficult to calculate. We still need to obtain the visit ratios. They are calculated exactly as in the case of the open model.

The following items are defined and used in the MVA equations.

n = the number of customers ($n=1,...,N$)
m = a queue number ($m=1,...,M$)
VR_m = visit ratio for queue m
ST_m = service time of queue m
$QL_m(n)$ = queue length of queue m with n customers
$QT_m(n)$ = queueing time of queue m with n customers
$TP_m(n)$ = throughput of queue m with n customers

The MVA equations are started with $QL_m(0) = 0$, $m = 1,...,M$. Then the following equations are solved for each queue as n varies from 1 to N.

$$QT_m(n) = ST_m(1 + QL_m(n-1))$$

$$TP_m(n) = \frac{n}{\sum_{i=1}^{M} \frac{VR_i}{VR_m} QT_i(n)}$$

$$QL_m(n) = TP_m(n)QT_m(n)$$

When these equations are used with the given visit ratios and service times, the following intermediate results are obtained.

```
                CPU          DISK        DRUM
QUEUE            1             2           3
VR             1.0           0.2         0.8
ST             0.02          0.044       0.008

For   n = 1:
QT             0.02000       0.04400     0.00800
TP            28.40909       5.68182    22.72726
QL             0.56818       0.25000     0.18182

For   n = 2:
QT             0.03136       0.05500     0.00945
TP            40.05826       8.01165    32.04660
QL             1.25637       0.44064     0.30299

For   n = 3:
QT             0.04513       0.06339     0.01042
TP            45.35544       9.07109    36.28435
QL             2.04678       0.57500     0.37822
```

For *n* equal to 4, the results shown previously are obtained.

4.3. ADVANTAGES AND RESTRICTIONS

Models which can be solved analytically provide several advantages over other solution techniques. Therefore, whenever a model can be solved with an analytic solution, this is the best choice. However, many models cannot be solved in this fashion. There are many restrictions which must be adhered to for analytic solutions. First we will discuss the advantages of solving models analytically, and then we will present the restrictions which must be met.

An analytiç solution gives the exact results for the model which is being solved. We will see that this is not so when using simulation. However, the model may not be an accurate representation of the actual system, so the results may not be close to the actual values. An analytic solution is generally very fast, often two orders of magnitude faster than simulation. Keep in mind that an analytic model may require a large amount of computation time if it is a closed model with a large number of customers or if there are many complex types of queues like ones with multiple servers. Since the model parameters are directly related to the performance measures, the effects of changes in some model parameters can be easily predicted. Both open and closed models are permitted. Mixed models which contain both open and closed chains are also allowed. The active resources are permitted to have a single server, multiple servers, or an infinite number of servers. A single server can serve at a fixed rate or can serve at a rate which varies as the number of customers at the resource changes. We call this a queue dependent server. Except at a FCFS service center, there can be different types of customers at an active resource. The different types of customers are routed to different classes at the resource. Each class has an associated service time distribution and provides a place for customers to wait before being served.

With all of these advantages you might think that there would not be a need for other types of solution methods. In actual practice, many models are solved with other techniques. This is because there are many complexities which exist in real systems which cannot be dealt with efficiently with an analytic approach. When solving a queueing network model analytically, there are a number of restrictions which must be satisfied. Many of these restrictions are not applicable when solving a single resource model. There are many formulas which can be applied with a single resource which do not hold when dealing with a network. The following restrictions apply to solving queueing networks. The interarrival time distribution at sources must be exponential. The routing decisions must be made by specifying a set of branching probabilities. We will see more general ways that these decisions can be made when using simulation. Simultaneous resource possession is not permitted. This would occur when a job acquired a passive resource and held onto that passive resource while receiving service at one or more active resources. Only active resources are allowed in an analytic model. Only sources, classes, and sinks are permitted in the routing. The queueing disciplines are limited to four types: first-come-first-served (FCFS), processor sharing (PS), last-come-first-served (LCFS), and infinite server (IS). For a resource with FCFS scheduling, there are further restrictions which require that the service time distribution be exponential and that all classes at the resource have the same mean service time. Priority queueing disciplines are not permitted. Each waiting line must have an infinite capacity. Finite waiting rooms are not allowed. For a multiserver resource, every server must serve at the same rate. The only performance measures available are utilization, throughput, mean queue length, mean queueing time, and the queue length distribution. The distribution of response time cannot be calculated for most models.

Because many of these restrictions are violated in real systems, analytic models cannot be used in modeling many systems. However, we recommend that a model be started with a simple analytic approach. Even if the model is not very realistic, it will provide a check on a more realistic model which is solved by another solution method. Also, when it is necessary to evaluate a model for a large set of parameter values, simulation is usually too time consuming. In this case, an analytic model could provide a large set of results quickly.

4.4. APPROXIMATIONS

There are many models which cannot be solved analytically because one or more of the above restrictions are violated. Some of these models can be solved by special techniques without resorting to simulation. The main reason for developing approximation techniques is to be able to solve the model in less time than would be required to simulate it.

There are many different types of approximation techniques. One approach which is very popular for representing simultaneous resource possession is to substitute a flow equivalent server for the passive resource and all of the active service centers visited while holding an element of the passive resource. Figure 4.3 shows an example of a model with a passive resource representing memory partitions. These memory partitions are allocated to customers while they visit the CPU and I/O devices. This model violates one of the restrictions for an analytic solution.

To approximately solve this model, it is decomposed into a submodel containing the CPU and I/O devices, and the aggregate model with the terminals and a flow equivalent server in place of the submodel. The submodel and model are shown in Figure 4.4. They both satisfy the restrictions discussed above and can be solved analytically. The parameters for the flow equivalent server are found by solving the submodel for all populations from one up to the number of elements of the passive resource. The submodel throughputs from each solution are used as the rates of the flow equivalent server.

Let us take a look at a concrete example of this technique. The following is a simulation model which corresponds to the diagram shown in Figure 4.3. Interactive users at a set of terminals submit requests to a computer system. Before entering the system, a request must obtain a memory partition. There are only four memory partitions available in this system. If they are all in use, the request must wait until one is free. After a request is allocated a memory partition, it receives service at the CPU and one of the I/O devices. With a specified probability, it will cycle back through the

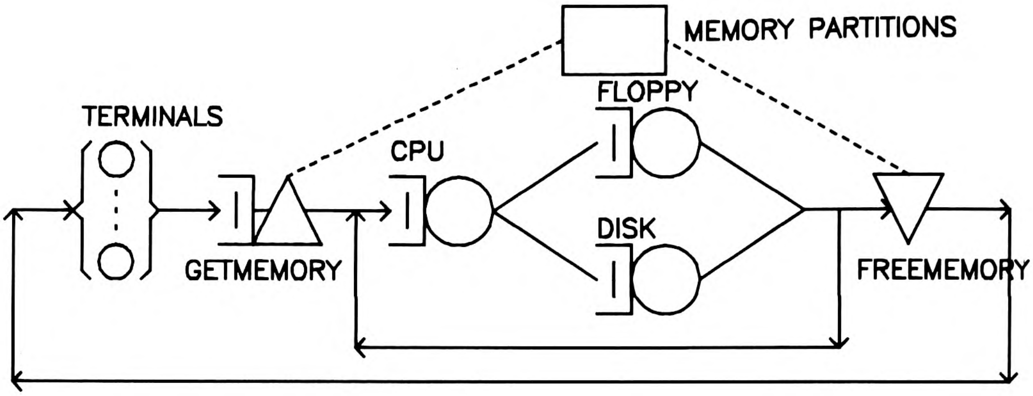


Figure 4.3. Model with Simultaneous Resource Possession

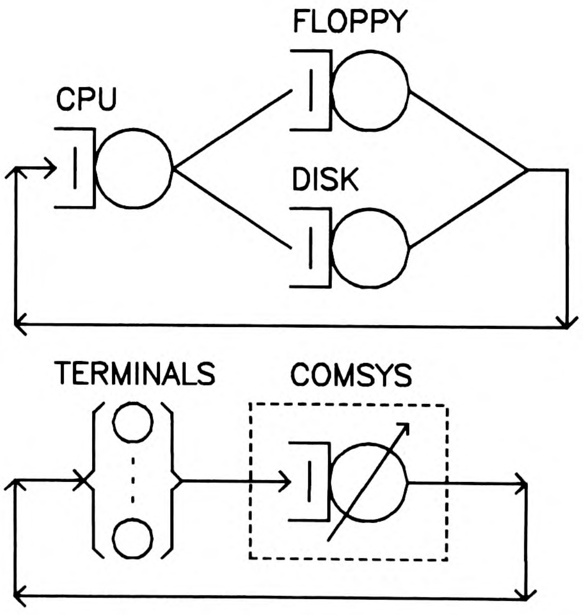


Figure 4.4. Submodel and Model with Flow Equivalent Server

CPU and I/O device again. When the request has completed, it releases the memory partition and sends a response to the terminal.

```
MODEL:EX4.3
   METHOD:simulation
   QUEUE:floppyq
      TYPE:fcfs
      CLASS LIST:floppy
         SERVICE TIMES:0.22 /* seconds */
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:0.019 /* seconds */
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:0.05 /* seconds */
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:10 /* seconds think time */
   QUEUE:memory
      TYPE:passive
      TOKENS:4   /* partitions */
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:freememory
   CHAIN:interactiv
      TYPE:closed
      POPULATION:30   /* users at the terminals */
      :terminals->getmemory->cpu->floppy disk;.1 .9
      :floppy->freememory cpu;1/8 1-1/8
      :disk->freememory cpu;1/8 1-1/8
      :freememory->terminals
```

This model is solved using simulation because of the passive queue representing the number of memory partitions. The FLOPPYQ and DISKQ are FCFS queues like the ones we have seen in the previous models in this chapter. The CPUQ and TERMINALSQ queues are very similar to these FCFS queues, except that the queueing disciplines are processor sharing and infinite server. The MEMORY queue is a passive queue. A passive queue is chosen to represent the memory contention because we do not know a priori how much time a job will hold onto a memory partition. This is determined by the length of time the job spends at the CPU and I/O devices. The number of tokens at the passive queue represents the number of memory partitions. GETMEMORY is the allocate node where the memory partitions are acquired by the jobs, and FREEMEMORY is the release node where they are freed.

This is a closed model. The chain population is equal to the 30 terminals which are actively using the system. The routing statements should be easy to follow. The only new feature present in this model which has not been shown previously is the arithmetic expressions used for the routing probabilities of choosing the FREEMEMORY or the CPU. With a probability of 1/8, FREEMEMORY is chosen. With a probability of 1−1/8, or 7/8, CPU is chosen. This means that on the average a job will cycle through the CPU and I/O subsystem eight times before returning to the terminals.

The following results have been obtained from a simulation of this model. We are omitting any information related to the accuracy of the simulation results here. The next chapter will discuss this topic.

```
ELEMENT          THROUGHPUT  QUEUE LENGTH  QUEUEING TIME
TERMINALSQ        2.29243      22.62959        9.75792
MEMORY            2.28742       7.37039        3.22209
```

To obtain an approximate solution to this model, it is decomposed into a submodel and an aggregate model which both can be solved by numerical solution. The following submodel contains a numeric parameter. This variable can be given different values while solving the submodel. This makes the submodel solution for a range of parameter values very convenient. NUMINSYS represents the number of jobs in the computer system. This is varied from one to four, because four is the maximum number of partitions available.

```
MODEL:EX4.4s
   METHOD:numerical
   NUMERIC PARAMETERS:numinsys
   QUEUE:floppyq
      TYPE:fcfs
      CLASS LIST:floppy
         SERVICE TIMES:0.22 /* seconds */
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:0.019 /* seconds */
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:0.05 /* seconds */
   CHAIN:interactiv
      TYPE:closed
      POPULATION:numinsys /* number of jobs in the system */
      :cpu->floppy disk;.1 .9->cpu
END
```

The following throughputs are obtained for the CPUQ for the four values of NUMINSYS. By dividing these throughput values by eight, which is the average number of cycles through the computer system, we calculate the rate of service for a flow equivalent server in the aggregate model.

```
NUMINSYS   ELEMENT        THROUGHPUT
   1         CPUQ          11.22334
   2         CPUQ          15.88912
   3         CPUQ          18.04608
   4         CPUQ          19.07506
```

The rate of service for the COMSYSQ is given as a vector with four elements. The first element will be used as the service rate when one customer is present. The second element will be used when there are two customers, the third when there are three. When there are four or more customers, the fourth rate will be used. A server with a vector of rates is a queue dependent server. This flow equivalent server acts very similar to the computer system submodel with the passive queue for the memory partitions.

```
MODEL:EX4.4a
   METHOD:numerical
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:10 /* seconds think time */
   QUEUE:comsysq
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:comsys
         WORK DEMANDS:1
      SERVER -
         RATES:11.22334/8 15.88912/8 18.04608/8 19.07506/8
   CHAIN:interactiv
      TYPE:closed
      POPULATION:30
      :terminals->comsys->terminals
END
```

When the following results from the aggregate model are compared with the results from the simulation of the original model, we see that they are very close. This is often the case in performing decompositions of this type. There will be much more presented about decompositions in the chapter on submodels.

```
ELEMENT          THROUGHPUT   QUEUE LENGTH   QUEUEING TIME
TERMINALSQ        2.26988       22.69876       10.00000
COMSYSQ           2.26988        7.30123        3.21658
```

There are many other kinds of approximation techniques. Most of them are applicable in certain specific situations. These other approximation techniques will not be discussed here. However, in many cases an approximation approach will be more efficient than using simulation.

4.5. FURTHER READING

The examples presented in this chapter are not representative of all types of analytic models. The following books provide much more in-depth information related to analytic solutions of queueing models: Allen [3], Kleinrock [95, 96], Kobayashi [98], Chapter 3 of Lavenberg [100], Lazowska, Zahorjan, Graham, and Sevcik [108], Sauer and Chandy [152], and Trivedi [183]. These books should be studied for a better understanding of analytic queueing network models. Further information on approximation techniques can be found in Chandy and Sauer [45], Chapter 4 of Lavenberg [100], and Sauer and Chandy [151].

4.6. EXERCISES

4.1 Calculate the utilization, throughput, mean queue length, and mean queueing time for a model similar to EX4.1 where the CPU service time is 0.015, the disk service time is 0.035, the drum service time is 0.01, the mean interarrival time is 0.02, and 30 percent of the jobs go to the disk and 70 percent to the drum.

4.2 Use the same model parameters as in Exercise 4.1 to solve a closed model with a customer population equal to 3.

4.3 Think of a simple model which can be decomposed into a submodel that can be solved separately. Use a flow equivalent server to represent the submodel.

4.4 Discuss the restrictions which must be imposed to solve a queueing network model with an analytic solution.

4.5 Add an infinite server representing a group of terminals to model EX4.2. Let the think time be 10 seconds. Find the performance measures.

4.6 Find the performance measures of a closed model with two service centers where one is an infinite server and the other a FCFS server. The service time at the infinite server is 10 hours and 0.5 hours at the FCFS server. There are 5 jobs in the model. This is called a machine repairman model. The infinite server represents the machines operating, and the FCFS server represents the machines being repaired one at a time.

4.7 Use a closed, cyclic queueing model to represent a service center with a finite capacity. Let the service time of the finite capacity queue be 0.5 seconds and the capacity be 2. Let the service time of the second service center be 1 second. This second service center represents a source of arrivals, where arrivals are discarded when the queue is saturated.

CHAPTER 5

SIMULATION

Simulation is a method of solution which mimics the behavior of the system. It is a statistical experiment which observes the behavior of the model as it evolves over time. We can run the simulation with trace data or with random numbers which are generated to represent arrival times, service times and routing probabilities. To use trace data to drive the simulation, measurements are taken from the system. These measurements can be arrival times of customers or service times at various devices. These measurements can be used by the simulation program in testing other aspects of the system. The random number approach is a more common way of simulating systems. The random numbers are not really random. They are produced by an algorithm according to specified probability distributions. Section 5.1 discusses random numbers in more detail.

If the model contains one or more sources, the simulation program schedules the time of arrival of the customers at the sources according to some probability distribution. The simulator generates the time of arrival of each customer at the service centers. When a customer arrives, the simulation program generates a request for service, that is, a service time. The program then schedules when a customer is to start its service and when it is to complete its service. From the time of arrival and the completion time it can calculate how much time the customer spent at the resource. By keeping track of all the different times, it can produce the desired performance measures. This type of bookkeeping performed by the simulator will be illustrated in Section 5.2.

In order to begin the simulation we need to specify whether there are to be any customers initially at any of the resources. This is called an initialization state. A state of the system is just the number of customers at every resource. For an open model, we might have no customers present initially. For a closed model, we must place the fixed number of customers at specific resources. As we run the simulation, the customers will begin to circulate among the resources. When the simulation stops, the performance measures which are calculated are random outputs based on the random numbers used in performing this run. For many simulations, the random outputs will reach a steady state. Being in a steady state does not mean that the state remains the same from that point on. Rather, it means that the distributions associated with the system states have converged to a limiting state.

Many times the results will exhibit transient characteristics. In a real system, a transient condition would exist initially when the system was turned on as customers begin to arrive for service. If the arrival pattern does not change and the system has the capacity to handle the requested service, a steady state or equilibrium condition may exist.

5.1. RANDOM NUMBERS

Random numbers are produced by an algorithm which produces a sequence of numbers which follow a specified probability distribution. Therefore the numbers are not random at all, they just appear as if they are random. Each number in the sequence is a function of the previous number. There are many different kinds of random number generators. We will discuss one which is widely used by many simulation programs.

We will first describe the method of producing random numbers between zero and one. The sequence of random numbers is started with an initial value called a seed. The formula for obtaining the next value in the sequence is

$$X_n = bX_{n-1}(\bmod m)$$

Mod stands for the modulo function. It produces the remainder by dividing the expression on the left, bX_{n-1}, by m. To use this formula, we need numbers for b, X_0 and m. Once these are specified, this formula can be used to produce numbers between zero and m minus one. We then divide these numbers by m to produce uniform random numbers between zero and one.

To produce random numbers from a probability distribution other than the uniform distribution, we can use the probability distribution function (PDF). This is a function which has values going from zero to one. A PDF for an exponential function is shown in Figure 5.1. After generating a uniform random number U between zero and one, we can find the corresponding exponential random number X as shown in the figure.

We can use this type of approach for generating random numbers from most of the commonly used probability distributions. We can therefore generate random arrival times, service times, and routing probabilities from many different probability distributions.

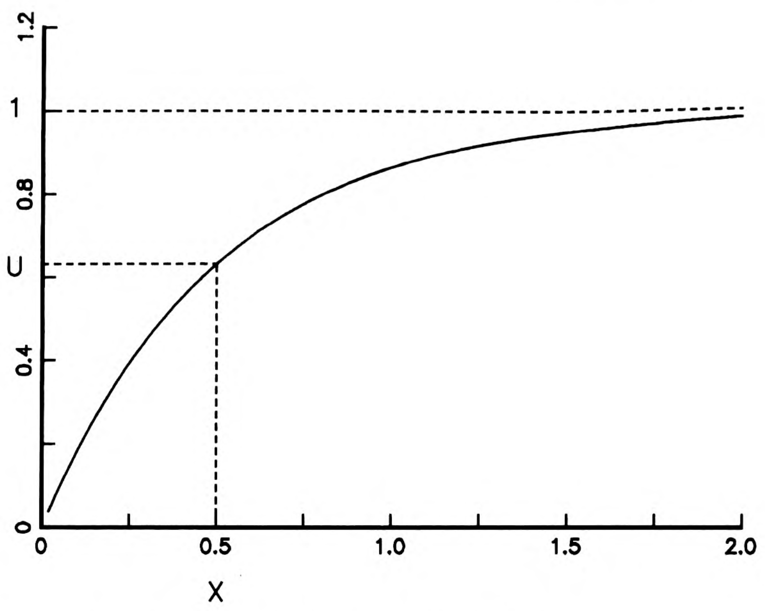


Figure 5.1. Exponential Cumulative Distribution Function

5.2. AN EXAMPLE

As an example of how a simulation works, we will use the closed central server model which was solved in Section 4.2.

```
MODEL:EX5.1
   METHOD:simulation
   QUEUE:cpuq
      TYPE:fcfs
      CLASS LIST:cpu
         SERVICE TIMES:.02
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.044
   QUEUE:drumq
      TYPE:fcfs
      CLASS LIST:drum
         SERVICE TIMES:.008
   CHAIN:chn
      TYPE:closed
      POPULATION:4
      :cpu->disk drum;.2 .8->cpu
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   CHAIN:chn
      NODE LIST:cpu disk
         INIT POP:3 1
   RUN LIMITS -
```

```
        NODES FOR DEPARTURE COUNTS:cpu
            DEPARTURES:10
    LIMIT - CP SECONDS:10
    TRACE:no
END
```

Except for using simulation as the solution method, this model is the same as model EX4.2 through the routing information. At the end of simulation models there are some statements pertaining to simulation. We have not discussed confidence intervals yet. They will be discussed in Section 5.6. In order to begin the simulation, the four customers in the closed chain must be initially placed somewhere. This model places three jobs at the CPU and one job at the DISK. The model will run until there are ten departures from the CPU.

We will concentrate our attention on the service requests and completions at the CPU. The service times at the CPU are from an exponential distribution, so random numbers are employed to generate service times of each customer. The following table illustrates various times associated with the CPU for the first ten customers.

CUSTOMER	ARRIVAL TIME	SERVICE TIME	SERVICE BEGINS	DEPARTURE TIME	QUEUE TIME
1	0.00000	0.00289	0.00000	0.00289	0.00289
2	0.00000	0.05121	0.00289	0.05410	0.05410
3	0.00000	0.01477	0.05410	0.06887	0.06887
4	0.00494	0.08494	0.06887	0.15381	0.14887
5	0.03642	0.01377	0.15381	0.16758	0.13116
6	0.05520	0.00795	0.16758	0.17552	0.12032
7	0.07143	0.03863	0.17552	0.21415	0.14272
8	0.15437	0.01454	0.21415	0.22869	0.07432
9	0.18031	0.00555	0.22869	0.23424	0.05393
10	0.18201	0.01387	0.23424	0.24811	0.06610
TOTALS		0.24811			0.86328

The arrival time at the CPU is zero for the first three jobs which were initialized there. The remaining arrival times are equal to the completion times of the jobs at one of the I/O devices. The service times are samples from an exponential distribution with a mean of 0.02. The remaining times are calculated by the simulation program. The beginning of service is either the arrival time if the queue is empty or is the departure time of the last customer. The departure time is equal to the start of service time plus the service time sample from the distribution. The time a job spends in the queue is equal to the departure time minus the arrival time. The simulation program keeps track of all of these numbers to produce the performance measures.

The service times and the queue times are summed to find the total amount of time the server was busy and the total amount of time jobs spent at the queue. The utilization can be found by dividing the total service time by the simulation time. In this example, the utilization is equal to one because the server was never idle. The mean queueing time is equal to the total queueing time divided by the number of customers, 0.86328/10, which equals 0.086328. The mean queue length is equal to the total queueing time divided by the simulation time, 0.86328/0.24811 = 3.47942. The simulation time for this example is the time of the last departure, because we stopped the simulation after ten customers finished. The throughput can be calculated using Little's rule. This is the queue length divided by the queueing time, 3.47942/0.086328 = 40.3047.

Notice that these performance measures are not equal to the ones calculated in Section 4.2. This simulation was run for a very small amount of time. To obtain more accurate estimates of the performance measures, the simulation would have to be run for many more completions.

5.3. ADVANTAGES

Using the simulation solution method has many advantages over the analytic approach. Attached to each customer we may have one or more attributes. These attributes can be used to save any information about a customer. An attribute might be used to identify different types of customers. Another attribute could be used as a counter to insure that the customer goes through a subsystem a specified number of times. An attribute could be used to determine the service time of a customer. In addition to customer attributes, a simulation model may contain global variables. Global variables are like variables in a programming language. They can be used to identify any type of condition, and their values are available to all customers. One example of a global variable would be to keep track of the number of customers present in an open chain. The variable would be incremented whenever a customer arrived and decremented when a customer went to a sink.

We have many different types of distributions which can be used for interarrival times, service times, work demands, allocation and creation of tokens, and assigning values to variables. The most common distributions include the constant, discrete, uniform, exponential, branching Erlang, hyperexponential, and normal. Additional distributions can usually be constructed by using these distributions in expressions or with other modeling facilities.

Routing decisions can be made based on the status of simulation conditions as well as by probabilities. We can test conditions like the queue length at a resource, whether there are servers or tokens available at a resource, the number of tokens held by a job, and the number of jobs related to a specified job. Access to this type of information gives us a lot of flexibility in determining where a job will go next.

More complex queueing disciplines are permitted with simulation. Both preemptive and nonpreemptive priority disciplines are available. In addition, it is possible to construct other disciplines using routing decisions and global variables. An explicit representation of round-robin scheduling can be constructed in this fashion.

In addition to the active resources which are available with a numerical solution, passive resources can be defined when using simulation. Passive resources permit the representation of simultaneous resource possession, blocking, finite waiting rooms, and complex algorithms and protocols.

With simulation we are able to produce multiple copies of a customer. The copies can be unrelated, travel over different paths, be different types of customers and never be reunited. Or the copies can be related to the original customer and be gathered together at some time in the future. Unrelated customers could be used to represent messages which generate acknowledgements. Related customers could be used to model parallel processing of a customer at multiple resources.

Of course, there are many disadvantages related to using simulation. Some of these were discussed in Chapter 4. The advantages of analytic solutions correspond to the disadvantages of simulation. The next three sections discuss disadvantages related to run length and statistical variability.

5.4. RUN LENGTH

How long should we run the simulation program? This is often a difficult question to answer. To illustrate some of the problems involved in answering this question, we will look at two simple models. The first model is a single resource model with customers arriving according to an exponential distribution. The service time distribution is also exponential. In queueing theory notation, this type of queue is called an M/M/1 queue. Here is the model:

```
MODEL:EX5.2
```

```
   METHOD:simulation
   QUEUE:q1
      TYPE:fcfs
      CLASS LIST:c1
         SERVICE TIMES:.4
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:src
      ARRIVAL TIMES:1
      :src->c1->sink
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   RUN LIMITS -
      QUEUES FOR DEPARTURE COUNTS:q1
         DEPARTURES:500
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

This simulation will not produce confidence intervals. When there is no initial state definition given, the model starts with no customers present. The model will run until there are 500 departures from Q1. In running the model we have continued the simulation by increasing the number of departures and looking at the performance measures at later times. The following results were produced for different number of departures:

```
Departures  Utilization  Throughput  Queue Length  Queueing Time
      500     0.37222      0.98429      0.58974       0.59915
     1000     0.38558      0.99383      0.60087       0.60196
     2000     0.38801      1.01451      0.62648       0.61752
     4000     0.39353      1.00925      0.64299       0.63709
     8000     0.39634      1.00175      0.64237       0.64125
    16000     0.39752      0.99692      0.65163       0.65364
    32000     0.39796      0.99348      0.65592       0.66023
    64000     0.39962      0.99597      0.65975       0.66242
   128000     0.39986      1.00005      0.66287       0.66284
```

This is a very simple model which we can solve analytically. The true values of the performance measures are 0.4 for the utilization and 1.0 for the throughput. The queue length and the queueing time are both equal to 0.66667. We can see that as the simulation is run for a longer time, the estimates of the performance measures are approaching the true values. However, it is very difficult with the information shown here to determine when to stop the simulation. The changes in the performance measures shown above are an illustration of statistical variability produced by random numbers. After a discussion of how to use confidence intervals, we will have more information about how to determine the accuracy of the results.

In addition to the above situation where the results are converging to some limiting values, some models go through a buildup period called an initial transient before reaching a steady state. To illustrate this type of behavior, the model shown in Figure 4.3 will be simulated. We are repeating the model definition for ease of readability.

```
MODEL:EX5.3
   METHOD:simulation
   QUEUE:floppyq
      TYPE:fcfs
      CLASS LIST:floppy
         SERVICE TIMES:.22
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.019
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:.05
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:10
   QUEUE:memory
      TYPE:passive
      TOKENS:4
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:freememory
   CHAIN:interactiv
      TYPE:closed
      POPULATION:30
      :terminals->getmemory->cpu->floppy disk;.1 .9
      :floppy->freememory cpu;1/8   7/8
      :disk  ->freememory cpu;1/8   7/8
      :freememory->terminals
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   CHAIN:interactiv
      NODE LIST:terminals
      INIT POP:30
   RUN LIMITS -
      QUEUES FOR DEPARTURE COUNTS:memory
         DEPARTURES:50
   LIMIT - CP SECONDS:180
   TRACE:no
END
```

This model definition includes information related to the initial length of the simulation run. Confidence intervals are not being produced. After initializing all 30 customers at the terminals, the simulation is run until there are 50 departures from the MEMORY queue. The following results for the queue length at the MEMORY queue have been found by continuing the simulation by increasing the number of departures.

Number of Departures	Queue Length
50	7.56759
100	9.26374
500	8.36907
1000	7.83167
2000	7.46833
4000	7.27863
8000	7.36439
16000	7.36102
32000	7.37758

We can see that in the beginning of the run the queue length at the MEMORY queue is very variable and appears to reach a steady state as the run increases. To overcome this initial transient condition, we can discard some portion of the data from the beginning of the run which is not representative. If we discard 1 percent of the initial portion of the run, the MEMORY queue length after 4000 departures is 7.28396. This is a slightly more accurate estimate. The discarding of the initial portion of the simulation, in other models where the initial transient is very different from the rest of the run, can have larger effects on the performance measures.

5.5. DIFFERENT SEEDS

In addition to the types of variability shown in the previous section, variability in the results can be encountered by running a simulation with different initial seeds. This will be illustrated by running model EX4.2 for 8000 departures using ten different seeds.

Seed	Utilization	Throughput	Queue Length	Queueing Time
1	0.39634	1.00175	0.64237	0.64125
2	0.39333	1.00299	0.66214	0.66017
3	0.39884	0.99834	0.64974	0.65082
4	0.39091	0.99573	0.63717	0.63990
5	0.39408	0.98063	0.63984	0.65226
6	0.40772	1.01557	0.70451	0.69274
7	0.39654	0.99221	0.67504	0.68019
8	0.41353	1.01695	0.73065	0.71842
9	0.40033	1.00410	0.67906	0.67629
10	0.40402	1.00874	0.67813	0.67224

An obvious observation from these results is that different random numbers definitely produce different results. The outcome also illustrates the danger of using results from one simulation run based on one set of random numbers. We need some additional information to determine the accuracy of the results. We need to perform a statistical analysis of the simulation output. In the next section, we will see how using different random numbers can help us in determining the accuracy of simulation results.

5.6. CONFIDENCE INTERVALS

One of the most difficult problems concerned with using simulation is how to determine the accuracy of the simulation estimates. In the last two sections we discussed the statistical variability of simulation estimates of performance measures as a function of the random numbers and the length of the run. There is no corresponding problem associated with solving a model numerically. When we solve a model numerically, we obtain an exact result for that model. Any difference between the results from a numerical model and the actual system is due to inaccuracies in the model or in the parameters of the model, and not due to inaccuracy of the solution. This is not the case when using simulation. Additional errors can also come from the simulation solution itself. Though we usually expect the inaccuracies of models to be the principal source of error in the performance measures, it is essential that we attempt to estimate the error introduced by statistical variability.

The usual method of estimating the variability in simulation results is to produce confidence interval estimates. Roughly speaking, a confidence interval is a range of values in which we expect to find the actual performance measure with a specified level of confidence. We can say that the true value of the performance measure will fall within the confidence interval with the probability specified by the confidence level. This can be discussed in more mathematical terms as follows. A point estimate of a performance measure is an average value of the result over the length of the simulation. Given some point estimate p, like the mean queueing time at the CPU, we can produce a confidence interval estimate $(p-\delta,p + \delta)$. Associated with this interval is a probability that the true value is contained within this interval. This probability, expressed in percent (e.g., 90%), is known as the confidence level. The quantity δ depends on the confidence level. The higher the confidence level is, the larger δ is. We will drop the term "estimate" from the phrase "confidence interval estimate," but it should be remembered that a confidence interval is only an estimate. Note that the true value may lie outside the confidence interval, but this happens only with a small probability. If a simulation is not run long enough, or if the performance measure considered is highly variable, then δ may be greater than p. In this case $p - \delta$ may be negative even though the performance measure must be greater than or equal to zero. Similarly, for performance measures known to be not greater than one (e.g., utilization), p and δ may be such that $p + \delta > 1$.

We will discuss three methods for estimating confidence intervals. There is no one method which works well in all situations, so we need to be able to choose an appropriate method.

- The method of independent replications is the preferred method for estimation of transient conditions. Independent replications may be applied to estimation of equilibrium characteristics, but one of the following two methods will usually be preferable for estimating equilibrium characteristics.

- The regenerative method is the preferred method for estimation of equilibrium behavior in models with regenerative characteristics. These types of characteristics will be discussed in Section 4.6.2.

- The spectral method is the preferred method for estimation of equilibrium behavior in models without regenerative characteristics. The spectral method may also be applied to models with regenerative characteristics.

For models that reach equilibrium, we recommend trying the regenerative method first. If the model does not exhibit regenerative characteristics, the spectral method should be used next. In all cases, the method of independent replications will always work, but it usually requires longer run times. These issues will be discussed further in the following sections along with their applications. The mathematical methods of constructing confidence intervals are built into some modeling packages. In this case, we just need to learn how to apply the different methods which are available.

5.6.1. Replications

A classical method for obtaining confidence intervals is the method of independent replications. With independent replications we repeat the simulation run several times with everything except the random number streams reset to the original initial state for each replication after the first. The random number streams for the second replication begin where the streams for the first replication ended; the streams for the third replication begin where the streams for the second replication ended, and so on.

To demonstrate how independent replications are used, we will simulate example EX5.2 from Section 5.4. The following information is related to using this method of generating confidence intervals.

```
   CONFIDENCE INTERVAL METHOD:replications
   INITIAL STATE DEFINITION -
   CONFIDENCE LEVEL:90    /* percent */
   NUMBER OF REPLICATIONS:5
   REPLIC LIMITS -
      QUEUES FOR DEPARTURE COUNTS:q1
         DEPARTURES:32000
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

Since the initial state is not given, there are no jobs present initially. The level of confidence is 90 percent. This will determine the widths of the confidence intervals that are estimated. The model is run five times with different random numbers. Each replication is stopped after 32000 customers complete at Q1.

The following results were obtained for this model.

```
          Utilization                  Throughput
   0.39894(0.39717,0.40071)  1.00036(0.99508,1.00564)

          Queue Length                Queueing Time
   0.66216(0.65742,0.66691)  0.66193(0.65626,0.66759)
```

The first value of each of these performance measures is the point estimate. The confidence interval values are between the parentheses. All of the confidence intervals are very narrow. These results are very close to the true values, and the confidence intervals do contain the true values.

Usually we are interested in equilibrium behavior of the modeled system. In this case we wish to have the replications long so that the effects of our choice of initial state will not be noticeable. *We prefer a few longer replications to many shorter replications.* Usually we choose the number of replications to be between five and ten. The only significant exception is when we want the replications short because we want to notice the effects of our choice of initial state. In this case we would be interested in transient behavior rather than equilibrium behavior. Then it may be quite reasonable to have many (20 or more) replications.

5.6.2. Regenerative Method

The regenerative method is a second method that can be used to estimate confidence intervals for equilibrium measures. The principal advantages of the regenerative method over replications are that we can make a single (long) simulation run instead of multiple (shorter) runs and that we need not be concerned about the effects of the choice of the initial state. The fact that there is no initial transient problem as with replications will be explained below. Even though the regenerative method has these advantages there are problems also.

With the regenerative method we must pick a "regeneration state," which is similar to the initial state. A regeneration state has the following properties.

- The model periodically returns to the regeneration state. The periods between occurrences of the regeneration state are called "cycles."

- When the model enters the regeneration state, the future behavior of the model depends only on the regeneration state. This means it is independent of the behavior that led to the entrance of that state. Because of this independent behavior, there is no initial transient problem. Every regeneration cycle, including the first one, behaves statistically identical to every other cycle.

The most convenient examples of regeneration states are found in Markov and semi-Markov processes. In a "nice" (semi-) Markov process, each state is a regeneration state, and except for practical considerations, all of the states are equally useful. A large subset of extended queueing networks can be described as (semi-) Markov processes, and these processes will usually be "nice" unless a queue of the network is saturated or a deadlock is possible in the network.

The principal practical consideration is that we would like the regeneration state to occur frequently during a simulation of reasonable length. By "frequently" we mean that there be at least some minimum number of cycles (say 20) during the simulation. If we do not have this property then we cannot reasonably use the regenerative method.

We would also like the state to be one which is easily detected by the simulation. For this reason, modeling packages which use the regenerative method usually only allow regeneration states which are specified by the number of jobs at each node with the understanding that additional characteristics of the states are specified implicitly. These implicitly specified characteristics include the following.

- Where arrival and service distributions are specified by the method of exponential stages—for example, branching Erlang or hyperexponential—any arrival and service times in progress are in the first stage in the regeneration state.

- At active queues where different orderings of the jobs in the queue are important—for example, FCFS queueing discipline—the ordering of jobs of different classes is the same as at the first occurrence of the regeneration state.

- At passive queues the ordering of jobs at different allocate nodes and different numbers of tokens requested is the same as at the first occurrence of the regeneration state.

For a further discussion of the regenerative method in general, see Crane and Iglehart [54, 55], Crane and Lemoine [56], Iglehart [81], Iglehart and Shedler [82], Chapter 4 of Kobayashi [98], Chapter 6 of Lavenberg [100], Lavenberg and Sauer [102], Lavenberg and Slutz [103], and Chapter 7 of Sauer and Chandy [152].

The following information is the type required when using the regenerative method.

```
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:ch1
      NODE LIST:c1
         REGEN POP:0
         INIT POP:0
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:q1
         MEASURES:qt    /* mean queueing time */
         ALLOWED WIDTHS:5    /* percent */
   SAMPLING PERIOD GUIDELINES -
      QUEUES FOR DEPARTURE COUNTS:q1
         DEPARTURES:16000
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

This will be used to simulate the M/M/1 queue which was simulated in the previous section. The regeneration state we have chosen is the empty system. As long as the load on the system is not too large, the open chain will be empty from time to time. Since the utilization of Q1 is 0.4, this regeneration state should occur frequently. The initial state is chosen to be the same as the regeneration state. This is not necessary in all models. If the initial state is different from the regeneration state, the simulation program will discard the data from the beginning of the run until the regeneration state is first entered.

The confidence level is 90 percent again. The regenerative method allows an automated run length control based on achieving confidence intervals of a previously specified width. A sequential stopping rule is employed. Periodically, the program checks the confidence interval of the mean queueing time at Q1 to see if it is less than 5 percent wide. This relative allowed width is determined by dividing the confidence interval width by the point estimate and multiplying by 100 to achieve a percent. For performance measures, like the utilization, which are between 0 and 1, we use the absolute width. This is just the confidence interval width, without dividing by the point estimate, multiplied by 100 to be a percent. If the above criterion is satisfied, the run is stopped. If it is not satisfied, the run continues for another sampling period. For this model the length of each sequential sampling period will be 16000 departures from Q1.

We could obtain the following performance measures for this model.

```
         Utilization                    Throughput
0.39796(0.39290,0.40302)  0.99348(0.98450,1.00245)

         Queue Length                 Queueing Time
0.65592(0.64062,0.67122)  0.66023(0.64746,0.67300)
```

These results are not quite as accurate as the ones from the simulation where we used replications. However, much less run time was used in this case. This run used only 23.09 seconds as opposed to 71.16 seconds for the replications solution. The regeneration state which was chosen for this model occurred 19,291 times. This is a large number of regeneration cycles.

5.6.3. Spectral Method

The spectral method is a third method we could use to estimate confidence intervals for equilibrium measures. Most methods for estimating confidence intervals depend on having items of data that are "independent and identically distributed." The method of independent replications achieves this "i.i.d." property by the protocol which repeats the simulation. The regenerative method depends on being able to observe the i.i.d. property during the simulation run as the simulation returns to the regeneration state. The spectral method does not depend on the i.i.d. property. Rather, it explicitly takes into consideration the correlation between data items in the simulation—for example, the dependencies between successive queueing times for a given queue. This is done without user awareness, other than the availability of confidence intervals. Therefore, the information necessary to use the spectral method is essentially the same as simulation without confidence intervals. A sequential stopping rule is also available with the spectral method. A significant advantage of the spectral method over independent replications is that we can make a single (long) simulation run instead of multiple (shorter) runs. Therefore we do not need to be as concerned about the effects of the choice of the initial state. The spectral method applies to equilibrium behavior of all models simulated using extended queueing networks, not just those with regenerative properties. For a statistical discussion of the spectral method, see Heidelberger and Welch [75].

Again we will use the M/M/1 queue model for describing how to apply the spectral method. The following information could be given when using it to generate confidence intervals.

```
   CONFIDENCE INTERVAL METHOD:spectral
   INITIAL STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      CONFIDENCE INTERVAL QUEUES:q1
         MEASURES:qt
         ALLOWED WIDTHS:5
   INITIAL PERIOD LIMITS -
      QUEUES FOR DEPARTURE COUNTS:q1
         DEPARTURES:16000
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

Since there is no initial state definition given, the simulation is begun with no customers present. The confidence level is 90 percent. We are using the sequential stopping rule to determine when the simulation should stop. The spectral method will only produce confidence intervals for the mean queueing time and the queueing time distribution. Here we use the same stopping criterion used previously with the regenerative method. The length of the initial sampling period is 16,000 departures from Q1.

The following results were obtained from a simulation using this information.

```
  Utilization   Throughput   Queue Length   Queueing Time
    0.39900       0.99553       0.66091     0.66385(0.64790,0.67980)
```

The confidence interval produced for the mean queueing time is wider than the one produced by the regenerative method, but the run using the spectral method was shorter. This run consumed only 11.14 seconds to produce the given data.

5.7. HYBRID MODELING

In evaluating a hybrid model, more than one solution method is invoked. Usually a numerical method is used to solve one or more submodels, and the results from the submodel solutions are used in conjunction with a simulation of the rest of the model.

We will present a simple example of a hybrid model. The model is EX4.3 from Section 4.4. We will use the same decomposition employed there for solving the central server submodel numerically. However, instead of replacing the passive queue and the submodel with a flow equivalent server, the passive queue will be kept in the model. The flow equivalent server will replace only the central server submodel. This decomposition is illustrated in Figure 5.2.

Since the submodel can be solved numerically, the substitution of the flow equivalent server using the service rates derived from solving the submodel is an exact method of replacing the submodel, at least as far as mean values of the performance measures are concerned. Since the model with the flow equivalent server has fewer resources, and therefore fewer simulation events to schedule, we expect the simulation of this model to be more efficient than the simulation of the entire model without the flow equivalent server.

We will use the results from Section 4.4 for the submodel solution. These results are shown as the rates of the COMSYSQ queue in the model definition below.

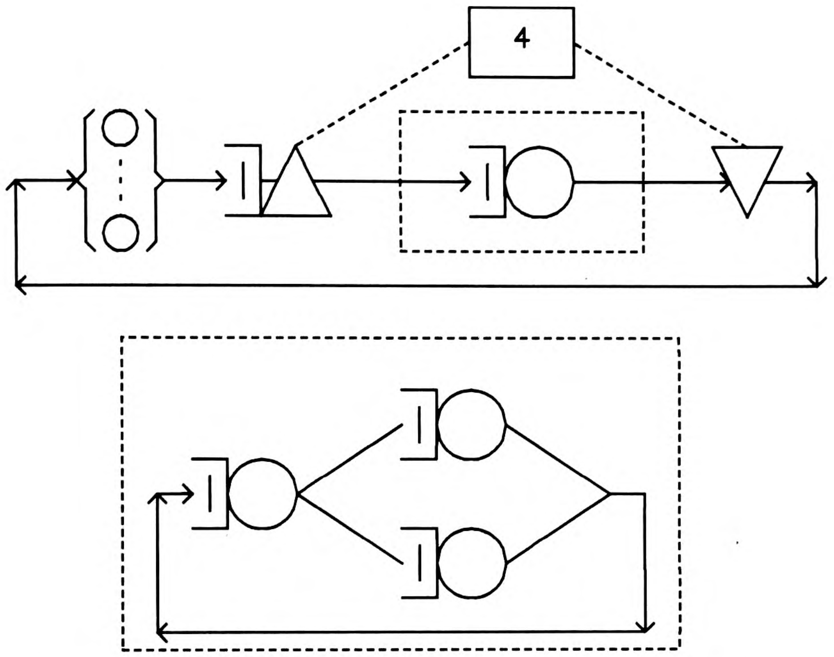


Figure 5.2. Decomposed Model and Submodel for Hybrid Model

```
MODEL:EX5.4
   METHOD:simulation
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:10 /* seconds think time */
   QUEUE:memory
      TYPE:passive
      TOKENS:4   /* partitions */
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:freememory
   QUEUE:comsysq
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:comsys
         WORK DEMANDS:1
      SERVER -
         RATES:11.22334/8 15.88912/8 18.04608/8 19.07506/8
   CHAIN:interactiv
      TYPE:closed
      POPULATION:30
      :terminals->getmemory->comsys->freememory->terminals
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:interactiv
      NODE LIST:terminals
         REGEN POP:30
         INIT POP:30
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:memory
         MEASURES:qt
         ALLOWED WIDTHS:10   /* percent */
   SAMPLING PERIOD GUIDELINES -
      QUEUES FOR DEPARTURE COUNTS:memory
         DEPARTURES:1000
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

Notice that the regenerative method is used to estimate confidence intervals. The initial state places all the users at the terminals. This is also the regeneration state. The simulation will be run until the relative confidence interval width of the mean queueing time at the MEMORY queue is less than or equal to 10 percent. The following results were obtained from the simulation.

```
                            Throughput
TERMINALSQ    2.26926( 2.24314, 2.29538)
MEMORY        2.26926( 2.24314, 2.29538)

                           Queue Length
TERMINALSQ   22.51093(22.19072,22.83113)
MEMORY        7.48906( 7.16886, 7.80927)

                           Queueing Time
TERMINALSQ    9.91995( 9.78835,10.05155)
MEMORY        3.30022( 3.13727, 3.46317)
```

It is difficult to compare the results from the various solution methods. We do not know the true values, and we do not know how accurate the results from the original simulation are because no confidence intervals were produced.

The main advantage of using a hybrid modeling approach is to reduce the number of events which must be simulated by replacing one or more portions of the model with resources which behave like the original model. The method employed to obtain the parameters of the substituted resource must be more efficient than the time necessary to solve the substituted portion of the original model. For a further discussion of hybrid modeling see Browne, Chandy, Brown, Keller, Towsley, and Dissley [33], Chiu and Chow [46], and Schwetman [168]. We will have more to say about hybrid modeling in Chapter 6 when we discuss decomposition.

5.8. FURTHER READING

For more information related to random numbers see the references by Knuth [97], Lewis, Goodman, and Miller [109], and Chapter 5 of Lavenberg [100]. There have been many books written about simulation. Some of the better ones are Banks and Carson [9], Fishman [64, 65], Gordon [71], Kleijnen [92, 93], Chapter 4 of Kobayashi [98], Chapters 6 and 7 of Lavenberg [100], Law and Kelton [106], Maisel and Gnugnoli [117], Pritsker and Pegdon [135], Schriber [164], and Shannon [171].

5.9. EXERCISES

5.1 Use the random number generator discussed in Section 5.1 with $m=32$, $b=9$ and $X_0 = 1$ to generate the next five numbers in the sequence.

5.2 Given the following five arrival times to a single service center (0.2, 0.6, 2.2, 2.6 and 3.0) and corresponding service times (0.8, 0.4, 1.2, 0.2, 0.2), calculate when service begins, the departure time, and the queue time of each of the five customers. If the total simulation time is 4.0 time units, calculate the utilization, the mean queueing time, the mean queue length, and the throughput of the service center.

5.3 Discuss the advantages and disadvantages of using simulation as a modeling tool.

5.4 Simulate an M/M/1 model like EX5.2 with the service time changed to 0.9. Run it for an increasing number of departures and notice how long it takes to approach the actual values.

5.5 Use a different seed for the random number generator for simulating the model discussed in Exercise 5.4.

5.6 Use a confidence interval method to produce confidence intervals for the model discussed in Exercise 5.4.

CHAPTER 6

MODEL STRUCTURE

A submodel is a portion of a model containing parameters which can be assigned values. A submodel may contain any subset of resources present in the model, and we may make one or more copies of the submodel. Submodels can be used to clarify the structure of a model, to avoid duplication of effort within a model, to permit sharing of parts of models, to introduce variability in the model structure and, with decomposition, to solve the submodel separately and replace the submodel with a flow equivalent server.

The structure of the model can be clarified by constructing submodels for the major subsystems to be represented. The submodels can be used to represent high-level abstractions of the subsystems which can be easily connected to form the overall system. If we have a model of a system which has a CPU and an I/O subsystem, we could construct a submodel representing the CPU and another submodel representing the I/O subsystem. The I/O submodel could also be decomposed into submodels nested within it representing each I/O device.

If a model contains similar subsystems, we could construct a submodel representing one copy of the subsystem with parameters which will capture the differences. Then the submodel can be duplicated for each subsystem with different values supplied for each copy of the submodel. In a communication network with several similar host computers, a submodel representing one host could be constructed and easily duplicated for each host needed in the model.

Many models contain subsystems which are similar to those used in other models. Submodels facilitate the use of portions of models. If a submodel of a CPU with round-robin scheduling has been constructed, this submodel can be used in many different models.

Very frequently, models have a requirement for having a variable number of resources. The number of resources can be specified as a model parameter, and a submodel can be built to represent one of the resources. Some modeling packages permit an arbitrary number of copies of the submodel to be created based on the value of a model parameter. A simple example of this is a model of an I/O subsystem that we want to evaluate for a variable number of I/O devices. This can be easily represented by using a submodel of one device and making any number of copies of it.

Hierarchical decomposition is a widely used technique for simplifying the solution of certain types of models. The model is decomposed into one or more submodels which are solved separately without the remainder of the model. Results from the submodel solution are used to characterize a flow equivalent server which is used in place of the submodel in an aggregate model. The flow equivalent server is usually a queue dependent server with appropriately chosen service rates. This decomposition and substitution can be accomplished by using any solution technique for solving the submodels and the aggregate model.

6.1. STRUCTURE CLARIFICATION

A good way to approach the modeling of a complicated system is to identify major subsystems and structure the model around these subsystems. A very high-level view of the system can be refined in a stepwise fashion by adding more and more details to the model. The model should consist of submodels which correspond to the major subsystems. The submodels can be defined at a high level. As time permits and as more accuracy is needed, these submodels can include more details and even have submodels defined within them. This block structuring helps to clarify how the model functions.

The following model shown in Figure 6.1 is a more detailed version of model EX4.2 which was diagramed in Figure 4.2.

```
MODEL:EX6.1
   METHOD:simulation
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.044
   QUEUE:drumq
      TYPE:fcfs
      CLASS LIST:drum
         SERVICE TIMES:.008
   SUBMODEL:rrqueue /*round robin queue*/
      NUMERIC PARAMETERS:mean_serve quantum overhead
      CHAIN PARAMETERS:chn
      QUEUE:q
         TYPE:fcfs
         CLASS LIST:cls
            SERVICE TIMES:constant(min(jv(0),quantum)+overhead)
      SET NODES:set_total
      ASSIGNMENT LIST:jv(0)=exponential(mean_serve)
      SET NODES:set_remain
      ASSIGNMENT LIST:jv(0)=jv(0)-min(jv(0),quantum)
      DUMMY NODES:dummy_out
      CHAIN:chn
         TYPE:external
         INPUT:set_total
         OUTPUT:dummy_out
         :set_total->cls->set_remain->cls dummy_out;if(jv(0)>0) if(t)
   END OF SUBMODEL RRQUEUE
   INVOCATION:cpu
      TYPE:rrqueue
      MEAN_SERVE:0.02
      QUANTUM:0.02
      OVERHEAD:0.0002
      CHN:chn
   CHAIN:chn
      TYPE:closed
      POPULATION:4
      :cpu->disk drum;.2 .8->cpu
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:chn
      NODE LIST:    disk  drum
         REGEN POP:    3     1
         INIT POP:     3     1
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:diskq
         MEASURES:qt
         ALLOWED WIDTHS:8   /* percent */
   SAMPLING PERIOD GUIDELINES -
      CYCLES:300
   LIMIT - CP SECONDS:90
   TRACE:no
END
```

Model EX4.2 was simple enough to solve numerically. Here we have replaced the CPU with a submodel which explicitly represents round-robin scheduling. In order to do this we will use simulation as the solution technique. The two queue definitions for the I/O devices are exactly the same as the previous model. The submodel for the RRQUEUE contains the modeling constructs necessary to depict the more complicated scheduling algorithm.

Numeric parameters in a submodel allow us to use symbolic names in the submodel definition which are given values at a later time. The chain parameter enables the chain defined in the submodel to be attached to a chain outside the submodel. When a customer enters the submodel, the total service demanded is assigned to a customer attribute. In RESQ the customer attributes are called job variables and identified by the keyword JV. Each customer has a vector of job variables which can be indexed starting at zero. For each customer, JV(0) is set equal to a sample from an exponential distribution which represents the total service demand. The service time

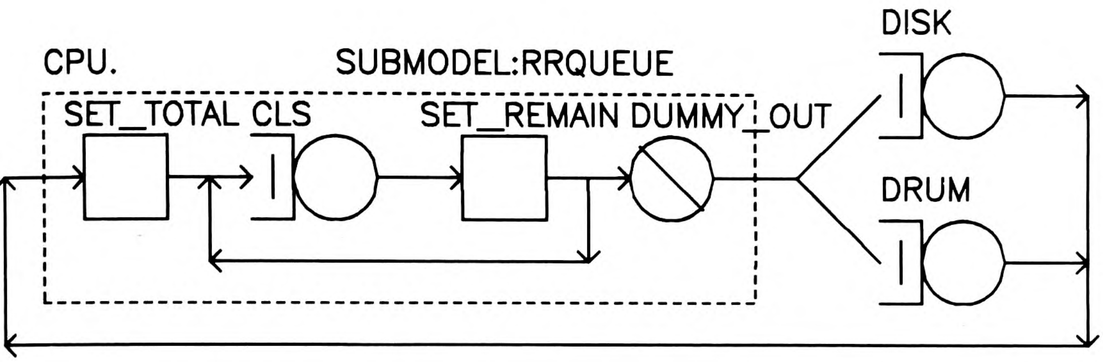


Figure 6.1. Round-robin CPU and Two I/O Devices

at the CPU will be equal to the round-robin quantum or the remaining service time, whichever is smaller, plus some overhead connected with system processing.

After receiving service at the CPU, the remaining service time is calculated by subtracting the smaller of the quantum or the remaining time. This again is stored in a customer attribute whose value is tested. If more service is necessary, the customer is routed back to the CPU. If the service is completed, the customer leaves the submodel.

In RESQ an invocation is a way of making a copy of a submodel. We will see later that there can be many invocations of the same submodel. When the submodel is invoked, the submodel parameters are assigned values. The parameter values given imply that the total service requested by each customer will be from an exponential distribution with mean 0.02, the quantum is 0.02 and the overhead, 0.0002. The routing statements in the model are exactly the same as in the previous version of the model.

This model uses the regenerative method for constructing confidence intervals. The regeneration state is the same as the initial state. There are three customers at the disk and one customer at the drum. The level of confidence is 90 percent. The sequential stopping rule is employed to check the accuracy of the confidence interval of the mean queueing time at the DISKQ. This condition will be checked at multiples of 300 regeneration cycles, and the simulation will be terminated when the accuracy condition is less than or equal to eight percent.

```
    SIMULATED TIME:      633.89233
          CPU TIME:          59.91
NUMBER OF EVENTS:            77186
NUMBER OF CYCLES:              900
```

These summary statistics show how much simulated time and processing time expired. The number of events is equal to the total number of completions at all queues; 900 regeneration cycles occurred for this run.

```
Element         Utilization                    Throughput
CPU.Q    0.96197(0.95860,0.96534)  74.79817(74.43318,75.16315)
DISKQ    0.41135(0.39475,0.42794)   9.32019( 9.03948, 9.60091)
DRUMQ    0.30078(0.29614,0.30542)  37.64676(37.22142,38.07210)

Element         Queue Length                  Queueing Time
CPU.Q    2.95755(2.92992,2.98517)   0.03954(0.03915,0.03993)
DISKQ    0.63162(0.59358,0.66967)   0.06777(0.06546,0.07008)
DRUMQ    0.41083(0.40217,0.41950)   0.01091(0.01073,0.01109)
```

Although this model is not the same as model EX4.2, many of these results are close to the results from the previous model. The CPU throughput is not close to the previous result because some customers go through the CPU more than once. The CPU mean queueing time is also different because of the round-robin scheduling.

We could now take this model and add further details. The CPU submodel could be refined more to represent other complexities. A new submodel could be defined to capture a more realistic I/O subsystem. This stepwise refinement could be continued to any level of detail desired.

6.2. EASE OF REPETITION

A frequent use of submodels is to repeat similar portions of a model. A submodel can be defined with parameters which can be assigned different values for each invocation. This ease of repetition simplifies the construction of many models.

To illustrate this feature, the following model is a simple representation of a manufacturing system with three tools. Figure 6.2 shows the model diagram. This model is based on one from Oates [129].

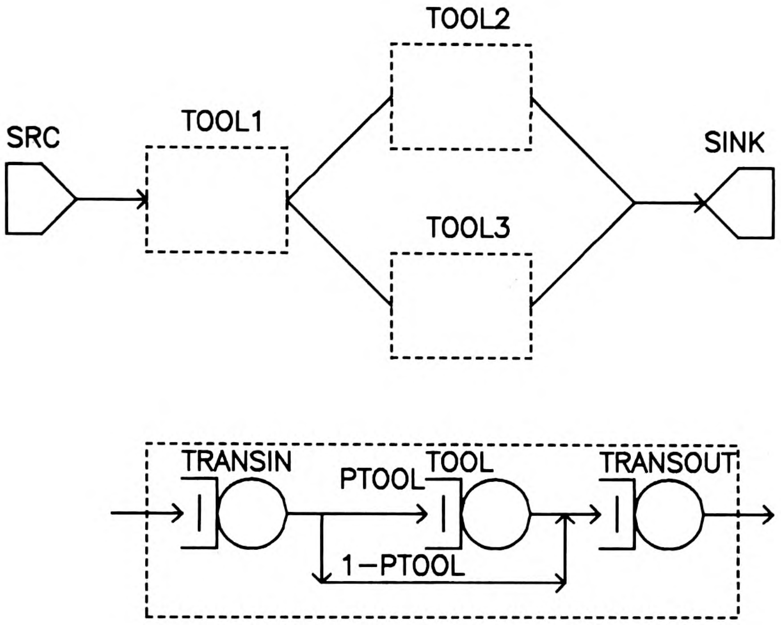


Figure 6.2. Three Invocations of a Tool Submodel

The TOOLSUB submodel contains three parameters. TRANSTIME will be used as the mean of an exponential distribution for the amount of time it takes to transfer a part to a tool, to bypass the tool, or to transfer the part out of the subsystem. TOOLTIME is the service time for parts which require service at the tool. PTOOL is a branching probability. Parts enter the submodel at the transfer input unit. With a probability of PTOOL they go to the tool. With one minus this probability, they bypass the tool. All parts go through the output transfer unit.

There are three invocations of the TOOLSUB submodel. Each one has a different set of parameter values. The routing shown at the bottom illustrates the order in which parts visit the tools.

```
MODEL:EX6.2
   METHOD:numerical
   SUBMODEL:toolsub
      NUMERIC PARAMETERS:transtime tooltime ptool
      CHAIN PARAMETERS:chn
      QUEUE:transinq
         TYPE:fcfs
         CLASS LIST:transin
```

```
            SERVICE TIMES:transtime
      QUEUE:toolq
         TYPE:fcfs
         CLASS LIST:tool
            SERVICE TIMES:tooltime
      QUEUE:transoutq
         TYPE:fcfs
         CLASS LIST:transout
            SERVICE TIMES:transtime
      CHAIN:chn
         TYPE:external
         INPUT:transin
         OUTPUT:transout
         :transin->tool transout;ptool 1-ptool
         :tool->transout
   END OF SUBMODEL TOOLSUB
   INVOCATION:tool1
      TYPE:toolsub
      TRANSTIME:4
      TOOLTIME:20
      PTOOL:.75
      CHN:line
   INVOCATION:tool2
      TYPE:toolsub
      TRANSTIME:5
      TOOLTIME:22
      PTOOL:.8
      CHN:line
   INVOCATION:tool3
      TYPE:toolsub
      TRANSTIME:3
      TOOLTIME:15
      PTOOL:.6
      CHN:line
   CHAIN:line
      TYPE:open
      SOURCE LIST:src
      ARRIVAL TIMES:25
      :src->tool1->tool2 tool3;.3 .7->sink
END
```

6.3. SHARING BETWEEN MODELS

Frequently, different models contain sections representing similar subsystems. It is convenient to be able to easily share portions of models in other models. Any part of a model can be placed in a separate file or as a member in a library. Then these files or members can be shared by many models.

```
   SUBMODEL:rrqueue /*round robin queue*/
```

```
      NUMERIC PARAMETERS:mean_serve quantum overhead
      CHAIN PARAMETERS:chn
      QUEUE:q
         TYPE:fcfs
         CLASS LIST:cls
            SERVICE TIMES:constant(min(jv(0),quantum)+overhead)
      SET NODES:set_total
      ASSIGNMENT LIST:jv(0)=exponential(mean_serve)
      SET NODES:set_remain
      ASSIGNMENT LIST:jv(0)=jv(0)-min(jv(0),quantum)
      DUMMY NODES:dummy_out
      CHAIN:chn
         TYPE:external
         INPUT:set_total
         OUTPUT:dummy_out
         :set_total->cls->set_remain->cls dummy_out;if(jv(0)>0) if(t)
   END OF SUBMODEL RRQUEUE
```

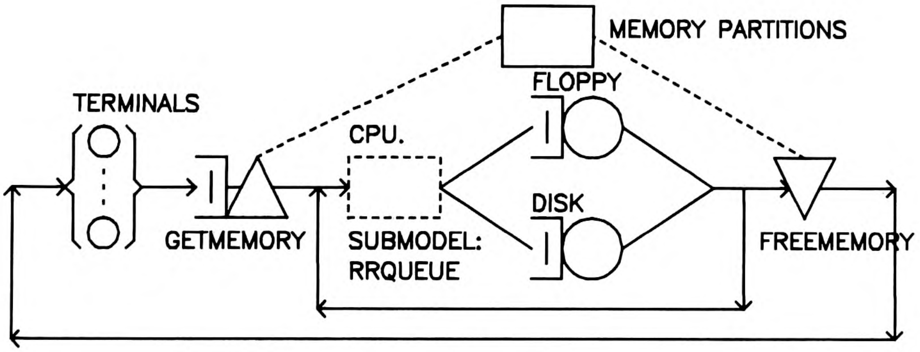


Figure 6.3. Model with Included Submodel

If we remove the RRQUEUE submodel from model EX6.1 and place it in a separate file, then, as shown in Figure 6.3, we can include it in any other model. The model that follows is similar to model EX4.3, but with the RRQUEUE submodel included and used in place of the original CPU queue definition.

```
MODEL:EX6.3
   METHOD:simulation
   QUEUE:floppyq
      TYPE:fcfs
      CLASS LIST:floppy
         SERVICE TIMES:0.22 /* seconds */
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:0.019 /* seconds */
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:10 /* seconds think time */
   QUEUE:memory
      TYPE:passive
      TOKENS:4   /* partitions */
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:freememory
   INCLUDE:rrqueue
   INVOCATION:cpu
      TYPE:rrqueue
      MEAN_SERVE:0.05
      QUANTUM:0.05
      OVERHEAD:0.0005
      CHN:interactiv
   CHAIN:interactiv
      TYPE:closed
      POPULATION:30   /* users at the terminals */
      :terminals->getmemory->cpu->floppy disk;.1 .9
      :floppy->freememory cpu;1/8 1-1/8
      :disk->freememory cpu;1/8 1-1/8
      :freememory->terminals
   CONFIDENCE INTERVAL METHOD:replications
   INITIAL STATE DEFINITION -
   CHAIN:interactiv
   NODE LIST:terminals
      INIT POP:30   /* users at the terminals */
   CONFIDENCE LEVEL:90
   NUMBER OF REPLICATIONS:5
   REPLIC LIMITS -
      QUEUES FOR DEPARTURE COUNTS:memory
         DEPARTURES:2000
   LIMIT - CP SECONDS:120
   TRACE:no
END
```

6.4. VARIABILITY IN MODEL STRUCTURE

The previous examples of using submodels have all involved a fixed number of invocations of each submodel. Often we would like a model to have a variable number of similar devices. Figure 6.4 and the following model illustrate an example of this type of approach.

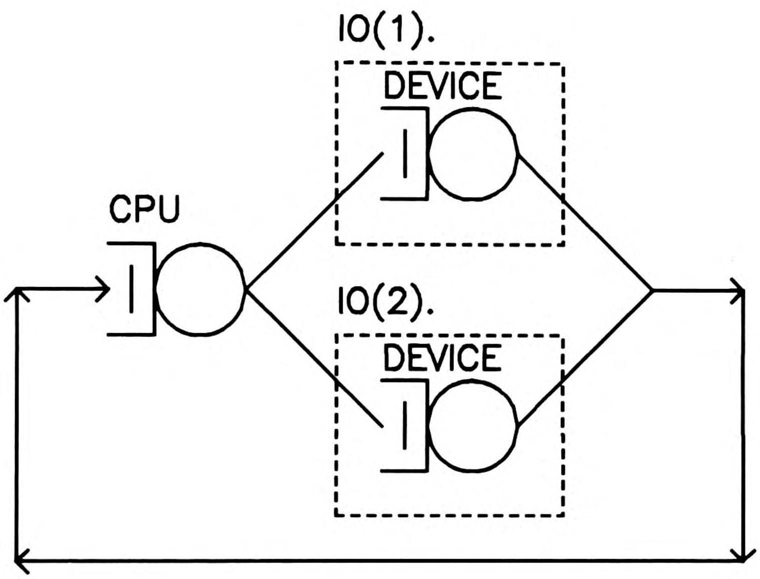


Figure 6.4. Model with a Variable Number of Devices

This is a model which is solved numerically. It contains a CPU and an arbitrary number of I/O devices. The number of I/O devices is specified as a numeric parameter named NIOS. The submodel definition contains a single queue for one of the I/O devices. The invocation uses a vector notation to produce NIOS copies of the submodel IOSUB. The routing in the model specifies that each I/O device will be branched to with the same probability.

```
MODEL:EX6.4
   METHOD:numerical
   NUMERIC PARAMETERS:nios
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:0.05
   SUBMODEL:iosub
```

```
      CHAIN PARAMETERS:chn
      QUEUE:deviceq
         TYPE:fcfs
         CLASS LIST:device
            SERVICE TIMES:0.10
      CHAIN:chn
         TYPE:external
         INPUT:device
         OUTPUT:device
   END OF SUBMODEL IOSUB
   INVOCATION:io(nios)
      TYPE:iosub
      CHN:chn
   CHAIN:chn
      TYPE:closed
      POPULATION:10
      :cpu->io(*).input;1/nios
      :io(*).output->cpu
END
```

When we solve this model and specify the number of I/Os as two and three, we obtain the following results.

```
NIOS  Element  Utilization  Throughput  Queue length  Queueing time
  2    CPU       0.83333     16.66666     3.33333        0.20000
       IO(1)     0.83333      8.33333     3.33333        0.40000
       IO(2)     0.83333      8.33333     3.33333        0.40000
  3    CPU       0.95078     19.01563     5.27969        0.27765
       IO(1)     0.63385      6.33858     1.57344        0.24823
       IO(2)     0.63385      6.33858     1.57344        0.24823
       IO(3)     0.63385      6.33858     1.57344        0.24823
```

6.5. DECOMPOSITION

Hierarchical decomposition is becoming a technique which is used in modeling certain types of systems. To decompose a model, the model is hierarchically structured with submodels which are solved individually. Results from each submodel solution are used to characterize a flow equivalent server which replaces the submodel. For models which cannot be solved analytically, decomposition may yield submodels and aggregate models which can all be solved analytically. An aggregate model is just a model with one or more submodels replaced by flow equivalent servers. Theoretical work which provides justification for using this approach can be found in Chandy, Herzog, and Woo [42] and Courtois [50, 52].

Even if the decomposed model does not produce submodels and aggregate models that can be solved analytically, decomposition could still be an attractive solution technique. If a model can be decomposed into a submodel which can be solved analytically and an aggregate model which must be simulated, the simulation of the aggregate model will frequently be more efficient than the original simulation. This is because the flow equivalent server requires only a single event in place of the many events required in the original submodel.

Unfortunately, there are some disadvantages to using decomposition. In Chapter 4 we mentioned that decomposition is an approximate solution as opposed to an exact representation of the system. Usually, only mean values of the performance measures may be obtained. It is difficult to calculate performance measures for the resources in the submodel. Commonly, only the results for the resources in the aggregate model are reported. It is difficult to determine the simulation run lengths and to generate confidence intervals. These last two issues are addressed in Blum, Donatiello, Heidelberger, Lavenberg, and MacNair [22].

There are several modeling situations which are particularly amenable to decomposition. When the submodel can be solved analytically, this is usually an ideal situation for using decomposition. An example is a model that contains only one queue which violates the restrictions for an analytic solution. In this case a submodel consisting of all but the queue which violates the analytic restrictions is constructed. This submodel can be solved analytically and the aggregate model solved by simulation. Decomposition will also be worthwhile when simulation has to be used to solve the submodel, and the aggregate model can be solved analytically if the amount of simulation run time to solve the submodel is not too large.

When the submodel to be solved has a fixed set of parameters, decomposition could be very fruitful. The results from the submodel solution could be substituted in an aggregate model, and a parametric study could be carried out on the aggregate model. This should reduce the time to do the parametric study. This is true even if the aggregate model must be simulated, because there will be fewer elements in the aggregate model and therefore fewer events to simulate.

If the time scale of the occurrence of the events in the aggregate model and the events in the submodel is very different, decomposition should be beneficial. This occurs when the model is nearly completely decomposable as discussed by Courtois [50, 51, 52]. We encountered this type of situation when modeling the simultaneous resource possession of the computer system with memory constraint in Section 4.4. The events which take place in the computer system after a job has been allocated memory occur much more frequently than the events outside the computer system.

If a model contains several identical submodels, decomposition could be particularly advantageous. The results from solving one of the submodels could be used as parameters for flow equivalent servers in the aggregate model. The more identical submodels there are in the model, the more efficient the decomposition will be.

6.5.1. Analytic Submodel

The following model will illustrate how a submodel which can be solved analytically can be used to reduce the amount of simulation time necessary to solve a decomposed model.

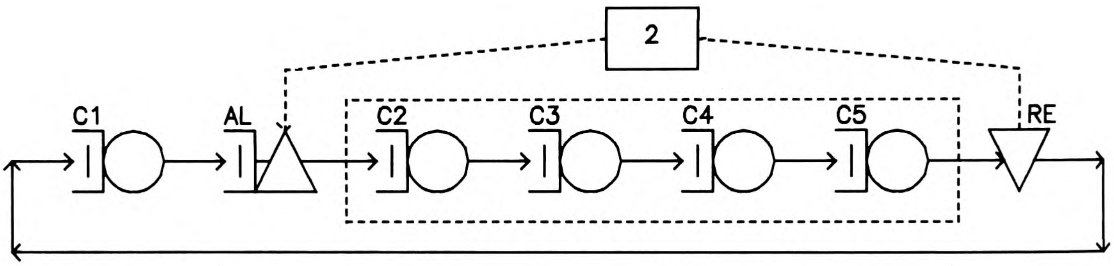


Figure 6.5. Model with Analytic Submodel

```
MODEL:EX6.5
   METHOD:simulation
   QUEUE:q1
      TYPE:fcfs
      CLASS LIST:c1
         SERVICE TIMES:.5
   QUEUE:pq
      TYPE:passive
      TOKENS:2
      DSPL:fcfs
      ALLOCATE NODE LIST:al
         NUMBER OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:re
   QUEUE:q2
      TYPE:fcfs
      CLASS LIST:c2
         SERVICE TIMES:.25
   QUEUE:q3
      TYPE:fcfs
      CLASS LIST:c3
         SERVICE TIMES:.25
   QUEUE:q4
      TYPE:fcfs
      CLASS LIST:c4
         SERVICE TIMES:.25
   QUEUE:q5
      TYPE:fcfs
      CLASS LIST:c5
         SERVICE TIMES:.25
   CHAIN:chn
      TYPE:closed
      POPULATION:10
      :c1->al->c2->c3->c4->c5->re->c1
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:chn
      NODE LIST:c1
         REGEN POP:10
         INIT POP:10
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:q1
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:50
   LIMIT - CP SECONDS:60
   TRACE:no
END
```

This model required 53.52 seconds of run time to reach the level of accuracy specified. The following decomposed version of this model, with the submodel solved analytically, required only 23.03 seconds to reach the same level of accuracy.

```
MODEL:EX6.5I    /* inner submodel */
   METHOD:numerical
   NUMERIC PARAMETERS:cpop    /* solved for values 1 and 2 */
   QUEUE:q2
      TYPE:fcfs
      CLASS LIST:c2
         SERVICE TIMES:.25
   QUEUE:q3
      TYPE:fcfs
      CLASS LIST:c3
         SERVICE TIMES:.25
   QUEUE:q4
      TYPE:fcfs
      CLASS LIST:c4
         SERVICE TIMES:.25
   QUEUE:q5
      TYPE:fcfs
      CLASS LIST:c5
         SERVICE TIMES:.25
   CHAIN:chn
      TYPE:closed
      POPULATION:cpop
      :c2->c3->c4->c5->c2
END
```

This submodel only has to be solved for chain populations of one and two since the passive queue enforces a maximum capacity of two in the submodel. The throughput is equal to 1.0 when the chain population is one and is equal to 1.6 for a population of two.

```
MODEL:EX6.50    /* Aggregate model */
   METHOD:simulation
   QUEUE:q1
      TYPE:fcfs
      CLASS LIST:c1
         SERVICE TIMES:.5
   QUEUE:pq
      TYPE:passive
      TOKENS:2
      DSPL:fcfs
      ALLOCATE NODE LIST:al
         NUMBER OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:re
   QUEUE:q2345
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:c2345
         WORK DEMANDS:1
      SERVER -            /* These come from the */
         RATES:1.0 1.6    /* submodel solution.  */
   CHAIN:chn
      TYPE:closed
      POPULATION:10
      :c1->al->c2345->re->c1
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:chn
      NODE LIST:c1
```

```
         REGEN POP:10
         INIT POP:10
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:q1 q2345
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:100
   LIMIT - CP SECONDS:60
   TRACE:no
END
```

The following results were obtained from a simulation of the original model and a simulation of the aggregate model using the flow equivalent server rates calculated analytically for the submodel.

```
        Q1           Simulation                   Decomposition
   Run length     53.52                        23.03
   Utilization     0.78806(0.78087,0.79526)   0.77600(0.76501,0.78700)
   Throughput      1.58841(1.58082,1.59601)   1.55380(1.53660,1.57100)
   Queue length    2.35685(2.26633,2.44736)   2.91300(2.78961,3.03638)
   Queueing time   1.48377(1.42656,1.54099)   1.87475(1.80213,1.94737)
```

6.5.2. *Simulation Submodel*

The model discussed in this subsection is taken from Sauer, MacNair, and Kurose [159, 160] and Sauer and MacNair [156]. It is a model of channel contention in an I/O subsystem. See any of these references for a description of the model.

Figure 6.6 is the model diagram. We will decompose the model so that the I/O subsystem is solved for multiprogramming levels of one through four. This submodel cannot be solved numerically, so simulation is used. The easiest way to solve the submodel is to replace the RRQUEUE submodel in the original model with a queue with zero service time. This is called the SHORTQ in the following submodel definition. The throughput through the SHORTQ is then used as a service rate in a flow equivalent server in the aggregate model. The following throughputs were produced by simulating this submodel for different multiprogramming levels: 42.7, 55.3, 61.3, and 65.7.

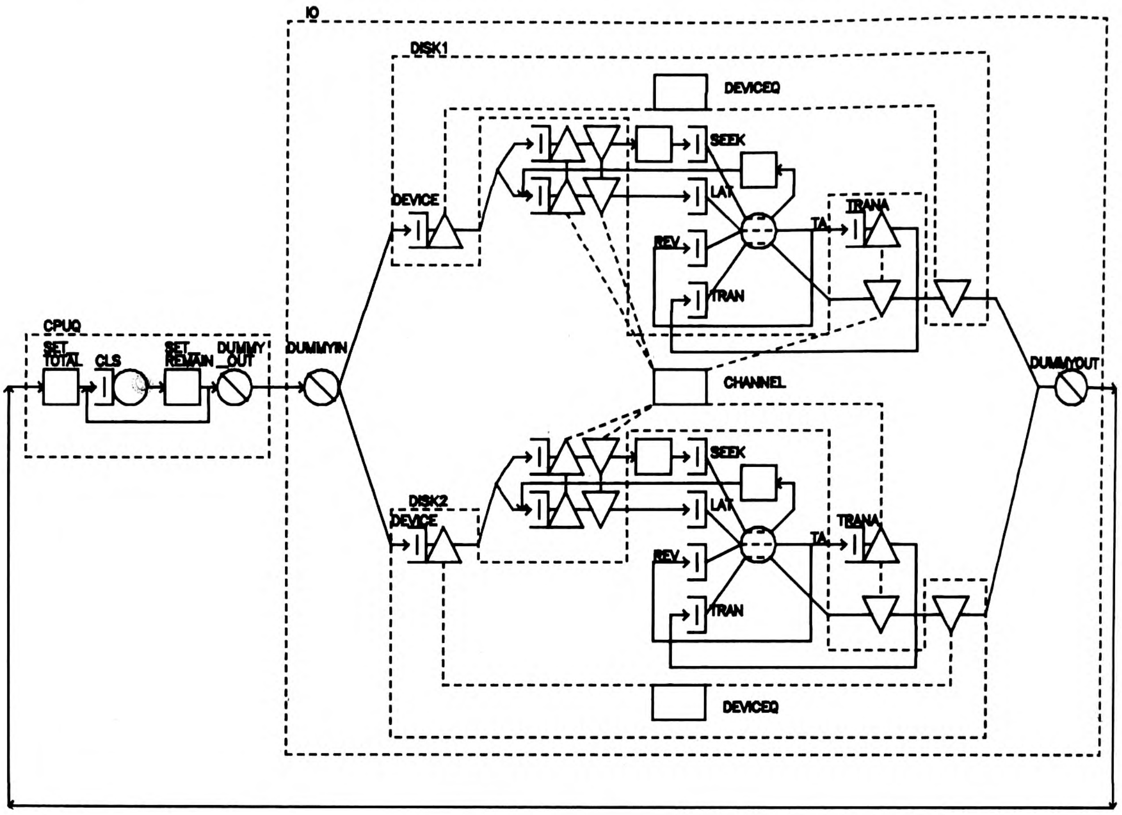


Figure 6.6. Model of Channel Contention

```
MODEL:EX6.6I   /* I/O submodel */
   METHOD:simulation
   NUMERIC PARAMETERS:mpl
   QUEUE:shortq
      TYPE:fcfs
      CLASS LIST:short
         SERVICE TIMES:constant(0)
   SUBMODEL:iosys /*subsystem with device contention for channel*/
      CHAIN PARAMETERS:c
      NUMERIC IDENTIFIERS:movearmp
         MOVEARMP:1/3
      QUEUE:channel
         TYPE:passive
         TOKENS:1
         DSPL:fcfs
         ALLOCATE NODE LIST:pos_s_a1 pos_l_a1 trana1
            NUMBERS OF TOKENS TO ALLOCATE:1
         ALLOCATE NODE LIST:pos_s_a2 pos_l_a2 trana2
            NUMBERS OF TOKENS TO ALLOCATE:1
         RELEASE NODE LIST:pos_s_r1 pos_l_r1 tranr1
         RELEASE NODE LIST:pos_s_r2 pos_l_r2 tranr2
      DUMMY NODES:dummyin dummyout
```

```
SUBMODEL:dasd /*individual device*/
   NUMERIC PARAMETERS:ncyl startarmt cylt revt trant
   NODE PARAMETERS:pos_s_a pos_s_r pos_l_a pos_l_r trana tranr
   CHAIN PARAMETERS:c
   GLOBAL VARIABLE IDENTIFIERS:oldcyl newcyl
      OLDCYL:ncyl/2
      NEWCYL:0
   QUEUE:deviceq
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:device
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:devicer
   QUEUE:timesq
      TYPE:fcfs
      CLASS LIST:seek
         SERVICE TIMES:standard(startarmt+abs(newcyl-oldcyl) ++
                                                        *cylt,0)
      CLASS LIST:lat rev
         SERVICE TIMES:uniform(0,revt,1) standard(revt,0)
      CLASS LIST:tran
         SERVICE TIMES:standard(trant,0)
   SET NODES:setnewcyl
   ASSIGNMENT LIST:++
        newcyl=ceil(uniform(0,oldcyl-1,(oldcyl-1)/(ncyl-1);++
                       oldcyl,ncyl,(ncyl-oldcyl)/(ncyl-1)))
   SET NODES:setoldcyl
   ASSIGNMENT LIST:oldcyl=newcyl
   CHAIN:c
      TYPE:external
      INPUT:device
      OUTPUT:devicer
      :device->pos_s_a pos_l_a;movearmp 1-movearmp
      :pos_s_a->pos_s_r->setnewcyl->seek->setoldcyl->pos_l_a
      :pos_l_a->pos_l_r->lat
      :lat->trana rev;if(ta>0) if(t)
      :rev->trana rev;if(ta>0) if(t)
      :trana->tran->tranr->devicer
END OF SUBMODEL DASD
INVOCATION:disk1
   TYPE:dasd
   NCYL:800
   STARTARMT:.01
   CYLT:.0001
   REVT:.0166667
   TRANT:.0029
   POS_S_A:pos_s_a1
   POS_S_R:pos_s_r1
   POS_L_A:pos_l_a1
   POS_L_R:pos_l_r1
   TRANA:trana1
   TRANR:tranr1
```

```
          C:c
       INVOCATION:disk2
          TYPE:dasd
          NCYL:800
          STARTARMT:.01
          CYLT:.0001
          REVT:.0166667
          TRANT:.0029
          POS_S_A:pos_s_a2
          POS_S_R:pos_s_r2
          POS_L_A:pos_l_a2
          POS_L_R:pos_l_r2
          TRANA:trana2
          TRANR:tranr2
          C:c
       CHAIN:c
          TYPE:external
          INPUT:dummyin
          OUTPUT:dummyout
          :dummyin->disk1.input disk2.input;.5 .5
          :disk1.output disk2.output->dummyout
    END OF SUBMODEL IOSYS
    INVOCATION:io
       TYPE:iosys
       C:c
    CHAIN:c
       TYPE:closed
       POPULATION:mpl
       :short->io->short
    CONFIDENCE INTERVAL METHOD:none
    INITIAL STATE DEFINITION -
    CHAIN:c
       NODE LIST:short
       INIT POP:mpl
    RUN LIMITS-
       NODES FOR DEPARTURE COUNTS:short
          DEPARTURES:5000
    LIMIT - CP SECONDS:30
    TRACE:no
END
```

The aggregate model consists of the flow equivalent server, representing the I/O subsystem, and the RRQUEUE submodel. This model must also be solved by simulation.

```
MODEL:EX6.60    /* Aggregate model */
    METHOD:simulation
    NUMERIC IDENTIFIERS:mean_serve quantum overhead
       MEAN_SERVE:.02
       QUANTUM:.02
       OVERHEAD:.0002
    QUEUE:ioq
```

```
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:ioc
         WORK DEMANDS:1
      SERVER -
         RATES:42.7 55.3 61.3 65.7
   SUBMODEL:rrqueue /*round robin queue*/
      NUMERIC PARAMETERS:mean_serve quantum overhead
      CHAIN PARAMETERS:chn
      QUEUE:q
         TYPE:fcfs
         CLASS LIST:cls
            SERVICE TIMES:standard(min(jv(0),quantum)+overhead,0)
      SET NODES:set_total
      ASSIGNMENT LIST:jv(0)=standard(mean_serve,1)
      SET NODES:set_remain
      ASSIGNMENT LIST:jv(0)=jv(0)-min(jv(0),quantum)
      DUMMY NODES:dummy_out
      CHAIN:chn
         TYPE:external
         INPUT:set_total
         OUTPUT:dummy_out
         :set_total->cls->set_remain->cls dummy_out;if(jv(0)>0) if(t)
   END OF SUBMODEL RRQUEUE
   INVOCATION:cpuq
      TYPE:rrqueue
      MEAN_SERVE:mean_serve
      QUANTUM:quantum
      OVERHEAD:overhead
      CHN:c
   CHAIN:c
      TYPE:closed
      POPULATION:4
      :cpuq.output->ioc->cpuq.input
   CONFIDENCE INTERVAL METHOD:replications
   INITIAL STATE DEFINITION -
   CHAIN:c
      NODE LIST:cpuq.set_total
      INIT POP:4
   CONFIDENCE LEVEL:90
   NUMBER OF REPLICATIONS:5
   INITIAL PORTION DISCARDED:10 /*percent*/
   REPLIC LIMITS-
      NODES FOR DEPARTURE COUNTS:cpuq.set_total
         DEPARTURES:10000
   LIMIT - CP SECONDS:300
   TRACE:no
END
```

The original model required 291 seconds of run time and the decomposed model, including the four submodel solutions, required only 152 seconds. The following is a comparison of some of the results produced by these two approaches.

```
CPUQ.Q        Simulation                        Decomposition
Run length    291.46                            151.67
Throughput     68.54985(68.37177,68.72792)       67.51341(67.32947,67.69736)
Utilization     0.87926( 0.87447, 0.88406)        0.86940( 0.86545, 0.87335)
Queue length    2.21399( 2.18089, 2.24708)        2.25151( 2.23616, 2.26686)
```

6.5.3. *Parametric Study with Submodel Parameters Fixed*

If we have a model which can be decomposed into a submodel whose parameters are fixed, then we can solve the submodel and use its results in the aggregate model. Then the aggregate model can be solved multiple times by varying parameters which affect only the aggregate model results. In this situation we can afford to spend a large amount of time solving the submodel. Figure 6.7 shows a model of a CPU, peripheral processors, and I/O units which can be found in Chandy and Sauer [45]. This model cannot be solved numerically. A simulation model is shown here.

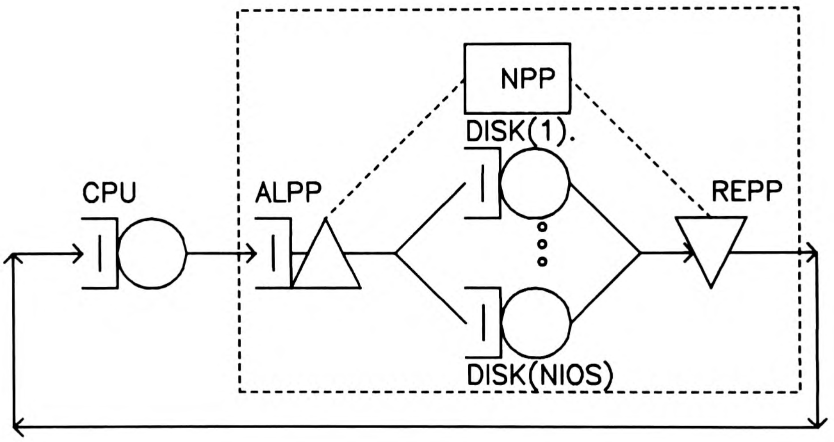


Figure 6.7. CPU, Peripheral Processors, and I/Os

```
MODEL:EX6.7
```

```
    METHOD:simulation
    NUMERIC PARAMETERS:npp mpl cpust diskst nios
    QUEUE:cpuq
       TYPE:ps
       CLASS LIST:cpu
          SERVICE TIMES:cpust
    QUEUE:ppq
       TYPE:passive
       TOKENS:npp
       DSPL:fcfs
       ALLOCATE NODE LIST:alpp
          NUMBERS OF TOKENS TO ALLOCATE:1
       RELEASE NODE LIST:repp
    SUBMODEL:io
       CHAIN PARAMETERS:ch1
       QUEUE:diskq
          TYPE:fcfs
          CLASS LIST:diskcl
             SERVICE TIMES:diskst
       CHAIN:ch1
          TYPE:external
          INPUT:diskcl
          OUTPUT:diskcl
    END OF SUBMODEL io
    INVOCATION:disk(nios)
       TYPE:io
       CH1:ch1
    CHAIN:ch1
       TYPE:closed
       POPULATION:mpl
       :cpu->alpp->disk(*).input;1/nios
       :disk(*).output->repp->cpu
    CONFIDENCE INTERVAL METHOD:regenerative
    REGENERATION STATE DEFINITION -
    CHAIN:ch1
       NODE LIST:cpu
          REGEN POP:mpl
          INIT POP:mpl
    CONFIDENCE LEVEL:90
    SEQUENTIAL STOPPING RULE:yes
       QUEUES TO BE CHECKED:disk(1).diskq
          MEASURES:qt
          ALLOWED WIDTHS:10
    SAMPLING PERIOD GUIDELINES -
       QUEUES FOR DEPARTURE COUNTS:disk(1).diskq
          DEPARTURES:1000
    LIMIT - CP SECONDS:120
    TRACE:no
END
```

If we fix the I/O subsystem parameters, we can decompose the model by setting the CPU service time to zero and removing the passive queue. This leaves us with a model that can be solved numerically. When we solve this with the multiprogramming level equal to one, two, three, and four, where four is the maximum number of peripheral processors, we obtain throughputs of 25.0, 41.7, 53.6, and 62.5.

```
MODEL:EX6.7I    /* Submodel */
   METHOD:numerical
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:0
   SUBMODEL:io
      CHAIN PARAMETERS:ch1
      QUEUE:diskq
         TYPE:fcfs
         CLASS LIST:diskcl
            SERVICE TIMES:.04
      CHAIN:ch1
         TYPE:external
         INPUT:diskcl
         OUTPUT:diskcl
   END OF SUBMODEL io
   INVOCATION:disk(5)
      TYPE:io
      CH1:ch1
   CHAIN:ch1
      TYPE:closed
      POPULATION:5
      :cpu->disk(*).input;1/5
      :disk(*).output->cpu
END
```

The aggregate model can also be solved numerically. Since the I/O subsystem parameters are fixed, we will solve the aggregate model for different CPU service times. The I/O submodel has been replaced by a flow equivalent server whose service rates come from the submodel solutions.

```
MODEL:EX6.7O    /* aggregate model */
   METHOD:numerical
   NUMERIC PARAMETERS:cpust
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:cpust
   QUEUE:ioq
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:ios
         WORK DEMANDS:1
      SERVER -
         RATES:25.0 41.7 53.6 62.5
```

```
    CHAIN:ch1
        TYPE:closed
        POPULATION:5
        :cpu->ios->cpu
END
```

The results from the aggregate model are shown here for different values of CPU service time.

```
 CPUST   Queue  Utilization   Throughput  Queue length  Queueing time
 0.005    CPUQ    0.30674      61.34865     0.42397        0.00691
          IOQ     0.99953      61.34865     4.57603        0.07459
 0.010    CPUQ    0.57158      57.15761     1.06795        0.01868
          IOQ     0.99065      57.15761     3.93205        0.06879
 0.020    CPUQ    0.86812      43.40623     2.43708        0.05615
          IOQ     0.90789      43.40623     2.56292        0.05905
 0.030    CPUQ    0.95905      31.96831     3.30363        0.10334
          IOQ     0.78280      31.96831     1.69637        0.05306
 0.040    CPUQ    0.98526      24.63155     3.77821        0.15339
          IOQ     0.67059      24.63155     1.22179        0.04960
 0.050    CPUQ    0.99386      19.87718     4.05608        0.20406
          IOQ     0.58112      19.87718     0.94392        0.04749
```

6.6. FURTHER READING

The following books and papers contain more information about submodels: Kobayashi [98], Lavenberg [100], Sauer and MacNair [154,156], Sauer, MacNair, and Kurose [158,159,160,161], and Sauer, MacNair, and Salza [162]. The following references should be consulted for additional information about decomposition: Avi-Itzhak and Heyman [6], Balbo and Bruell [7], Bard [10], Brandwajn [27], Browne, Chandy, Brown, Keller, Towsley, and Dissley [33], Chandy, Herzog, and Woo [42], Chandy and Sauer [45], Chiu and Chow [46], Courtois [50,51,52], Lavenberg [100], Sauer [149], Sauer and Chandy [151,152], Schwetman [168], and Thomasian and Nadj [181].

6.7. EXERCISES

6.1 Discuss the reasons for using submodels in model construction and solution.

6.2 Construct a model with at least one submodel where the submodel helps to clarify the structure of the model.

6.3 Build a submodel containing a CPU and several I/O devices. Use several copies of the submodel in a model containing terminals submitting transactions to multiple host systems represented by copies of the submodel.

6.4 Given the following submodel which assigns a normal random variable to a specified job variable, illustrate how it could be used in a model to produce service times which follow a normal distribution:

```
SUBMODEL:normal
/* Submodel to generate a normal random variable for a     */
/* distribution with mean MU and standard deviation SIGMA.*/
/* The normal r.v. is put in JV(NORMRV).                   */
   NUMERIC PARAMETERS:mu sigma normrv
   CHAIN PARAMETERS:chn1
   GLOBAL VARIABLES:u1 u2 x sg y
      U1:0
      U2:0
      X:0
      SG:0
      Y:0
   SET NODES:set1
   ASSIGNMENT LIST:u1=uniform(0,1,1) u2=uniform(0,1,1) ++
                   x=(-ln(u1)) ++
                   y=(x-1)*(x-1)/2 y=(-y) y=exp(y)
   SET NODES:set2
   ASSIGNMENT LIST:sg=uniform(0,1,1)
   SET NODES:set3
   ASSIGNMENT LIST:x=(-x)
   SET NODES:set4
   ASSIGNMENT LIST:x=(sigma*x)+mu jv(normrv)=x
   CHAIN:chn1
      TYPE:external
      INPUT:set1
      OUTPUT:set4
      :set1->set2 set1;if(u2<=y) if(t)
      :set2->set3 set4;if(sg>.5) if(t)
      :set3->set4
END OF SUBMODEL normal
```

6.5 Apply the decomposition techniques discussed in Section 6.5 to a model of a system you are familiar with.

6.6 Construct a submodel which represents four servers where the servers are picked at random from among those that are free.

CHAPTER 7

INTERPRETING RESULTS

We have discussed modeling constructs, solution methods, and model structures. Now we will describe how to interpret the results produced when we solve models. The various performance measures produced will be examined. The model results contain different types of errors, and the sources of these errors will be illustrated. Some information pertaining to how to determine whether the model is producing valid results will be presented. We will exhibit the results from a model constructed at different levels of detail. An analysis of the effects of modifying model parameters will illustrate how the behavior of different configurations can be studied. Some results are very sensitive to certain model parameters. An example of this will be given. Frequently, the results from a model solved using multiple parameter values are plotted. Some typical types of graphs will be described.

7.1. PERFORMANCE MEASURES

There are many different performance measures produced by modeling packages, but only a few of them are available when using an analytic solution. We will first focus on the results which are available when using both types of solution techniques. The utilization of a server at an active resource is the fraction of time the server is busy. At a passive resource, the utilization refers to the fraction of time a token is in use. The utilization gives a measure of the time a resource is busy. The throughput is a measure of the customer completion rate. It is the number of completions per unit of time. A simple relationship to keep in mind is that the utilization is equal to the throughput times the service time. The mean queue length is the average number of jobs waiting in line and in service. To find the average number of jobs waiting, subtract the utilization times the number of servers from the mean queue length. The mean queueing time is the average amount of time a job spends waiting in line and in service. To find the average waiting time, subtract the mean service time from the mean queueing time. Recall that Little's Rule also tells us that the mean queue length is equal to the throughput times the queueing time. For an open chain, the average chain population is equal to the average number of customers in the chain, and the mean response time is the average amount of time it takes a customer to go from a source to a sink. The queue length distribution can also be calculated when solving a model analytically. The queue length distribution values are the probabilities that the queue length is equal to each possible value.

These performance measures will be illustrated by displaying the corresponding results produced by solving a RESQ simulation model of an M/M/1 queue with a mean interarrival time of one and a mean service time of 0.5.

```
ELEMENT          UTILIZATION
MM1Q             0.49976

ELEMENT          THROUGHPUT
MM1Q             0.99957

ELEMENT          MEAN QUEUE LENGTH
MM1Q             0.98753

ELEMENT          MEAN QUEUEING TIME
MM1Q             0.98796

ELEMENT          OPEN CHAIN POPULATION
CH1              0.98753

ELEMENT          OPEN CHAIN RESPONSE TIME
CH1              0.98796

ELEMENT          QUEUE LENGTH DISTRIBUTION
MM1Q               0:0.50024
                   1:0.25063
                   2:0.12564
                   3:0.06355
                   4:0.03093
                   5:0.01539
                   6:7.2115E-03
                   7:3.2366E-03
                   8:1.5775E-03
                   9:8.0843E-04
                  10:3.9507E-04
```

Only the probabilities up to a queue length of ten have been shown for the queue length distribution.

When using simulation, there are many more performance measures available. Some measures which give an indication of variability in the results are the standard deviation of queue length, the standard deviation of queueing time, the maximum queue length, and the maximum queueing time. The queueing time distribution gives the probability that the queueing time is less than or equal to a specified value. The following results are for the same M/M/1 queue.

```
ELEMENT             STANDARD DEVIATION OF QUEUE LENGTH
MM1Q                1.38078

ELEMENT             STANDARD DEVIATION OF QUEUEING TIME
MM1Q                0.97274

ELEMENT             MAXIMUM QUEUE LENGTH
MM1Q                14

ELEMENT             MAXIMUM QUEUEING TIME
MM1Q                14.55514

ELEMENT             QUEUEING TIME DISTRIBUTION
MM1Q                5.00E-01:0.39183
                    1.00E+00:0.63186
                    1.50E+00:0.78011
                    2.00E+00:0.86905
                    2.50E+00:0.92311
                    3.00E+00:0.95440
```

The queueing time probabilities are given for a list of discrete points.

With simulation there are other results which are sometimes of use. The system state of where the jobs are left when the simulation stops is available. The mean service time at each active resource is the average value observed during the run. This is almost always different from the mean specified with the distribution input parameter. Since the simulation program is sampling from distributions, the average it observes is usually different from the specified mean. The number of departures from a resource is often helpful in debugging a model. Values of global variables and customer attributes are also available. Statistics related to the usage of tokens at passive resources can also be displayed.

In addition to the performance measures which are directly available, we are often interested in the response time between two arbitrary points in a model. This can be obtained by using a passive resource. An allocate node is placed where the response time is to start. A release node is placed at the point which ends the response time. To ensure that no waiting occurs at the allocate node, the number of tokens at the passive resource must be large enough so that it is never exhausted. Using a passive resource like this also provides queueing time distribution results.

7.2. SOURCES OF ERROR

When we examine the performance measures produced by solving a model, they may contain errors from several different sources. One particularly common source of error is caused by inaccurately estimating the input parameters of the model. To run the model we must supply values for service times distributions, arrival time distributions, routing probabilities, number of tokens at passive resources, chain populations, queueing disciplines, and other items. The values we supply are just estimates of the actual parameters. Some of the parameters are critical in producing the solution of the models. Slight errors in estimating their values can result in large errors in some of the performance measures.

This can be a particularly difficult problem to deal with. In designing a new system, it is often hard to obtain accurate estimates for input parameters.[1] Even when modeling an existing system for which measurements are available, frequently the measurements are inaccurate or missing necessary information. Software and hardware monitors used with computer systems and communication networks do not produce all the data necessary to run most models. Predicting parameters which represent future workloads is even more troublesome. There is usually no easy way to determine exactly how to predict future workload requirements.

There can also be errors in the model itself. The structure of the model can be incorrect. Some key resources can be missing from the model. The model may contain logical errors. These types of errors are normally not as difficult to deal with as the errors in the parameter estimation just described.

When solving a model with simulation, we must also be concerned with the statistical variability which exists in the results. Since there is randomness involved in sampling from distributions, the results are also random. We saw in Chapter 5 how confidence intervals can help in determining the accuracy of the results. We also described how the simulation can automatically stop when the desired level of accuracy is detected.

7.3. SIMULATION ACCURACY

Because of this statistical variability connected with simulation, in some sense the simulation results are inaccurate estimates for the model being solved. This is not true of results produced by an analytic solution. The results produced by solving an analytic model are exact for that model. However, the analytic model may not be an accurate representation of the actual system. Therefore, the analytic results may not agree with the system behavior. Since a simulation model may be made as detailed and realistic as we like, the results from the simulation model may be closer to the behavior of the system in spite of the statistical variability.

The accuracy of the simulation results can be controlled by the length of time we run the model. If we could run the simulation for an infinite amount of time, the results would be exact for the associated model. A model which has reached equilibrium will normally become more accurate as we increase the run length. If we have a steady state model and are using independent replications, we will have to increase the length of each replication. If the model is a transient solution, we will have to increase the number of replications.

We will illustrate this discussion of simulation accuracy with a simple model of a computer system. Figure 7.1 shows an interactive system with a memory constraint.

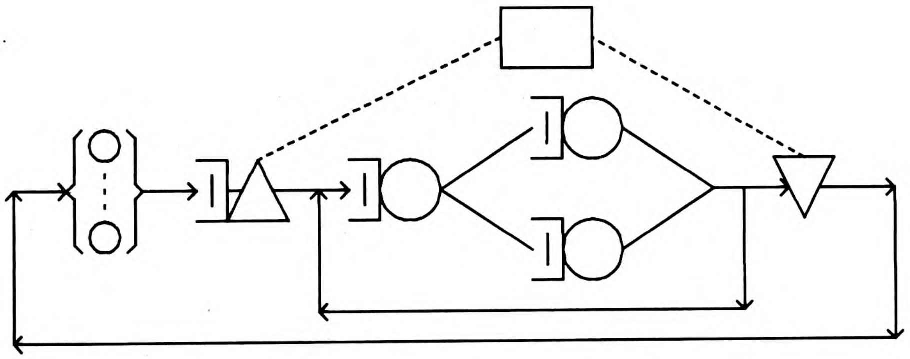


Figure 7.1. Model Diagram of an Interactive System

There are active service centers for the terminals, the CPU and two I/O devices. There is a passive resource to represent memory contention. The following is a RESQ model corresponding to the diagram in Figure 7.1.

```
MODEL:EX7.1
   METHOD:simulation
   NUMERIC PARAMETERS:thinktime users pageframes
   NUMERIC IDENTIFIERS:floppytime disktime cputime
      FLOPPYTIME:.22
      DISKTIME:.019
      CPUTIME:.05
   NUMERIC IDENTIFIERS:cpiocycles
      CPIOCYCLES:8
   QUEUE:floppyq
      TYPE:fcfs
      CLASS LIST:floppy
         SERVICE TIMES:floppytime
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:disktime
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:cputime
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:thinktime
   QUEUE:memory
      TYPE:passive
      TOKENS:pageframes
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:discrete(16,.25;32,.5;48,.25)
      RELEASE NODE LIST:freememory
   CHAIN:interactiv
      TYPE:closed
      POPULATION:users
      :terminals->getmemory->cpu->floppy disk;.1 .9
      :floppy->freememory cpu;1/cpiocycles 1-1/cpiocycles
      :disk->freememory cpu;1/cpiocycles 1-1/cpiocycles
      :freememory->terminals
   QUEUES FOR QUEUEING TIME DIST:memory
      VALUES:1 2 3 4 5 6 7 8
   QUEUES FOR QUEUE LENGTH DIST:memory
   MAX VALUE:users/2
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:interactiv
   NODE LIST:terminals
      REGEN POP:users
      INIT POP:users
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:memory cpuq
         MEASURES:qt qt
         ALLOWED WIDTHS:10 10
   SAMPLING PERIOD GUIDELINES -
      QUEUES FOR DEPARTURE COUNTS:memory
         DEPARTURES:2000
   LIMIT - CP SECONDS:300
   TRACE:no
END
```

This discussion will concentrate on the simulation accuracy of the results of this model. We are using the regenerative method to produce confidence intervals. All the customers in the closed chain are initialized at the terminals, and this same system state is used as the regeneration state. The confidence level is 90 percent. The sequential stopping rule is employed to check the accuracy of the confidence intervals for the mean queueing times at the MEMORY and CPUQ queues. The accuracy criteria are checked after every two thousand departures from the MEMORY queue.

The following simulation results were obtained for this model.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 13:42:53  DATE: 02/24/84
MODEL:EX7.1
THINKTIME:10
USERS:30
PAGEFRAMES:128
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                      SIMULATED TIME:     7683.34766
                            CPU TIME:         102.83
                    NUMBER OF EVENTS:         294452
                    NUMBER OF CYCLES:            323

WHAT:QTBO(MEMORY)


ELEMENT         MEAN QUEUEING TIME
MEMORY          3.39507(3.23577,3.55438) 9.4%

WHAT:
CONTINUE RUN:YES

EXTRA SAMPLING PERIODS:1

SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
SAMPLING PERIOD END: MEMORY DEPARTURE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:     8592.57813
                          CPU TIME:         114.97
                  NUMBER OF EVENTS:         329312
                  NUMBER OF CYCLES:            375


WHAT:ALLBO


ELEMENT          UTILIZATION
MEMORY           0.84943(0.84087,0.85800) 1.7%
FLOPPYQ          0.39982(0.39208,0.40756) 1.5%
DISKQ            0.30872(0.30642,0.31102) 0.5%
CPUQ             0.89862(0.89349,0.90375) 1.0%
TERMINALSQ       0.00000(0.00000,0.00000)


ELEMENT          THROUGHPUT
MEMORY           2.24636(2.22549,2.26722) 1.9%
FLOPPYQ          1.80830(1.78245,1.83416) 2.9%
DISKQ            16.23109(16.13292,16.32925) 1.2%
CPUQ             18.03940(17.93738,18.14142) 1.1%
TERMINALSQ       2.24636(2.22549,2.26722) 1.9%
FREEMEMORY       2.24636


ELEMENT          MEAN QUEUE LENGTH
MEMORY           7.57585(7.28393,7.86776) 7.7%
FLOPPYQ          0.58988(0.57167,0.60809) 6.2%
DISKQ            0.41949(0.41539,0.42358) 2.0%
CPUQ             2.47420(2.43792,2.51047) 2.9%
TERMINALSQ       22.42415(22.13223,22.71606) 2.6%


ELEMENT          STANDARD DEVIATION OF QUEUE LENGTH
MEMORY           3.95216
FLOPPYQ          0.85820
DISKQ            0.71946
CPUQ             1.37433
TERMINALSQ       3.95216


ELEMENT          MEAN QUEUEING TIME
MEMORY           3.37250(3.22446,3.52055) 8.8%
FLOPPYQ          0.32621(0.31905,0.33336) 4.4%
DISKQ            0.02584(0.02569,0.02600) 1.2%
CPUQ             0.13716(0.13558,0.13873) 2.3%
TERMINALSQ       9.98245(9.86628,10.09862) 2.3%
```

```
ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
MEMORY         2.39267
FLOPPYQ        0.31081
DISKQ          0.02506
CPUQ           0.15172
TERMINALSQ     9.91859


ELEMENT        MEAN TOKENS IN USE
MEMORY         108.72725(107.63087,109.82364) 2.0%


ELEMENT        MEAN TOTAL TOKENS IN POOL
MEMORY         127.99998


ELEMENT        QUEUE LENGTH DISTRIBUTION
MEMORY           0:0.01525(0.01255,0.01795) 0.5%
                 1:0.03285(0.02844,0.03726) 0.9%
                 2:0.05133(0.04575,0.05691) 1.1%
                 3:0.06762(0.06109,0.07415) 1.3%
                 4:0.07830(0.07213,0.08447) 1.2%
                 5:0.08662(0.08021,0.09302) 1.3%
                 6:0.09317(0.08730,0.09904) 1.2%
                 7:0.08688(0.08218,0.09158) 0.9%
                 8:0.08753(0.08243,0.09264) 1.0%
                 9:0.08465(0.07914,0.09016) 1.1%
                10:0.07819(0.07292,0.08346) 1.1%
                11:0.06698(0.06175,0.07221) 1.0%
                12:0.05524(0.04982,0.06067) 1.1%
                13:0.04026(0.03474,0.04579) 1.1%
                14:0.02788(0.02318,0.03258) 0.9%
                15:0.01908(0.01497,0.02318) 0.8%


ELEMENT        QUEUEING TIME DISTRIBUTION
MEMORY         1.00E+00:0.15776(0.14438,0.17113) 2.7%
               2.00E+00:0.32997(0.30835,0.35159) 4.3%
               3.00E+00:0.50622(0.48078,0.53165) 5.1%
               4.00E+00:0.65879(0.63516,0.68243) 4.7%
               5.00E+00:0.78241(0.76208,0.80273) 4.1%
               6.00E+00:0.86540(0.84836,0.88244) 3.4%
               7.00E+00:0.92032(0.90722,0.93342) 2.6%
               8.00E+00:0.95379(0.94494,0.96263) 1.8%


ELEMENT        DISTRIBUTION OF TOKENS IN USE


ELEMENT        DISTRIBUTION OF TOTAL TOKENS IN POOL
ELEMENT             MAXIMUM QUEUE LENGTH
MEMORY              21
FLOPPYQ             5
DISKQ               6
CPUQ                7
TERMINALSQ          30


ELEMENT             MAXIMUM QUEUEING TIME
MEMORY              20.62199
FLOPPYQ             2.64047
DISKQ               0.29332
CPUQ                1.94650
TERMINALSQ          111.93118
```

There were eight sampling periods. After each one of these the accuracy criteria were checked. In order to insure that the results were sufficiently accurate, we continued the simulation until the stopping criteria had been satisfied for two successive sampling periods. This occurred after one more sampling period. Extra sampling periods force the simulation to run longer and thus can help overcome some of the small sample problems of the sequential stopping rule. On a very short run, severe underestimates of the confidence interval widths may result in the criteria being accepted.

In some cases the accuracy criteria could be violated at the end of additional sampling periods. As you can see by comparing the mean queueing time of the MEMORY queue after eight and nine sampling periods, these results are fairly close. By continuing the run for the extra sampling period, we have ensured that the accuracy condition still holds.

7.4. VALIDATION

Validation is the process of ensuring that the model produces correct results. The model must be validated for the baseline case and for performance predictions. The baseline case is the solution of the model with parameters reflecting an existing system and a comparison of the results against measurements from the system. When designing a new system, actual measurements do not normally exist. In this case the analyst must convince himself or herself about the validity of the model based on intuition or discussions with system experts. If measurements exist, determining whether the model is producing correct results is normally a simple procedure. If the results do not agree with the measurements, the model or the input parameters must be adjusted. It is also possible that the measurement data are inaccurate. Jobs that have started before the measurement interval or ended after the measurement interval can cause errors in measurement data.

Validating performance predictions is much more difficult. One reason for doing the prediction is that the system is not available for measurements. So there is nothing to compare against. However, if a system is later installed based on the model results, a comparison should be performed to determine the validity of the model. If the system and the model do not agree, an investigation should be conducted to determine the cause of disagreement. The knowledge gained in determining the sources of error can be useful in future modeling studies.

The sources of error were discussed in Section 7.2. When the model results are incorrect, a determination of which types of errors are causing the problem must be undertaken. Logical errors in the model can normally be found by reviewing the model, or in the case of a simulation, by running a trace of the model. Most simulation programs provide a facility for obtaining detailed trace information as the simulation is in progress. The trace usually includes customer movement information from node to node, resource requests and completions at active and passive resources, event handling and event list processing, and periodic displays of the number of customers at each resource. Errors related to input parameters or model structure are more difficult to deal with but must also be resolved. Statistical variability of a simulation model is usually not a serious problem, because confidence intervals give us an indication of the accuracy of the results. The solution for increasing the accuracy in this case normally involves extending the length of the simulation run.

7.5. LEVEL OF DETAIL

The level of detail of a model is determined by its purpose. A model being used to design a new system or for performing capacity planning can be a gross, high-level representation of the system. A model used to tune a system must incorporate much more detail to capture all of the tuning effects.

A model used for capacity planning of a large computer system, like an MVS or VM type system, will normally contain a service center for each of the resources in the system whose utilization is at least five percent. These devices include the CPU, DASD devices, and tape drives. The channels, control units, and heads of strings are frequently not explicitly included. The time to use these devices is often included in the service demands of the DASD and tape service centers. Priority scheduling is permitted at the CPU, and memory contention can be depicted as a passive resource with a given multiprogramming level. Paging and swapping are represented by service demands at the appropriate DASD devices. Different workloads, like TSO, batch and data-base applications, can be included as multiple chains in the model. Interactive workloads with terminals are usually modeled as an infinite server in a closed chain whose chain population is equal to the number of terminals, or as a source of arrivals with a specified interarrival time distribution in an open chain.

For some modeling purposes, this level of detail is not adequate. The I/O subsystem is a very complicated subsystem. I/O path sharing, channel, control unit and head of string contention, rotational position sensing, buffered DASD, paging, and swapping are difficult to represent accurately. Loosely coupled and tightly coupled multiprocessors can also be an additional complexity. Obtaining an estimate of the network delays to remote terminals or other systems is also difficult. These more complex features require a more detailed model to represent them accurately.

Similar hierarchical structuring exists in communication network models. A high-level model might include service centers for communication lines, control units, a single service center for entire computer systems, and terminals. This would not be adequate for network designers working on new communication protocols. This task would require more detail including an item-by-item representation of all of the layers of protocols in the network.

Manufacturing models also can be constructed at different levels of detail. A high-level model might leave out machine failures and rework. A decision can be made as to how accurately batches of jobs should be represented. Some resources capable of parallel processing might simply be represented by a multiserver or a more realistic submodel might be used.

7.6. MODIFICATION ANALYSIS

After a baseline model is validated, the model is used to predict the future behavior of the system under various workloads or different configurations. The process of changing the parameters and structure of the baseline model to act as a predictive model is called modification analysis. The model and its parameters are modified to analyze alternative system behavior.

We will analyze the simple model illustrated in Figure 7.2. It contains a CPU and two I/O devices. This is an open model with transactions arriving from an external source according to a specified arrival rate. Other parameters of the model include the service times at the three service centers and visit ratios representing the average number of times a job visits a service center.

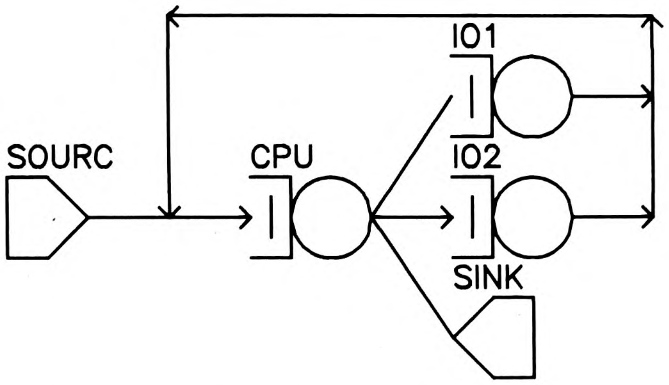


Figure 7.2. Computer System Model for Modification Analysis

A RESQ model, which includes symbolic numeric parameter names for the model parameters, is included here.

```
MODEL:EX7.2
   METHOD:numerical
   NUMERIC PARAMETERS:arrivlrate stcpu stio1 stio2
   NUMERIC PARAMETERS:           vrcpu vrio1 vrio2
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:stcpu
   QUEUE:io1q
      TYPE:fcfs
      CLASS LIST:io1
         SERVICE TIMES:stio1
   QUEUE:io2q
      TYPE:fcfs
      CLASS LIST:io2
         SERVICE TIMES:stio2
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:sourc
      ARRIVAL TIMES:1/arrivlrate
      :sourc->cpu->io1 io2 sink; ++
                   vrio1/vrcpu vrio2/vrcpu 1/vrcpu
      :io1 io2->cpu
END
```

This model can be solved analytically. There are three service centers, one for the CPU and one for each of the I/O devices. RESQ expects an interarrival time distribution, so the reciprocal of the arrival rate is used in the chain definition. RESQ uses branching probabilities for making routing decisions, so the visit ratios are combined in simple expressions to produce the probabilities. When we solve this model with the following set of parameters, we obtain the baseline performance measures.

```
ARRIVLRATE:5 /* TRANSACTIONS PER SECOND */
STCPU:.009 /* 9 MS PER VISIT */
STIO1:.040 /* 40 MS PER VISIT */
STIO2:.025 /* 25 MS PER VISIT */
VRCPU:6 /* VISITS PER TRANSACTION */
VRIO1:1 /* VISIT PER TRANSACTION */
VRIO2:4 /* VISITS PER TRANSACTION */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT        UTILIZATION
CPUQ           0.27000
IO1Q           0.20000
IO2Q           0.50000

ELEMENT        THROUGHPUT
CPUQ           29.99998
IO1Q           5.00000
IO2Q           19.99998

ELEMENT        MEAN QUEUE LENGTH
CPUQ           0.36986
IO1Q           0.25000
IO2Q           1.00000

ELEMENT        MEAN QUEUEING TIME
CPUQ           0.01233
IO1Q           0.05000
IO2Q           0.05000

ELEMENT        OPEN CHAIN POPULATION
CH1            1.61986

ELEMENT        OPEN CHAIN RESPONSE TIME
CH1            0.32397
```

Now we are in a position to perform a modification analysis. If we wish to determine the effect of doubling the speed of the CPU, one parameter which is affected is the CPU service time. This is probably the only parameter which will change, and its new value should be 4.5 ms. The following results are obtained with this new parameter value.

```
ARRIVLRATE:5
STCPU:.0045 /* 4.5 MS PER VISIT */
STIO1:.040
STIO2:.025
VRCPU:6
VRIO1:1
VRIO2:4
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT            UTILIZATION
CPUQ               0.13500
IO1Q               0.20000
IO2Q               0.50000

ELEMENT            THROUGHPUT
CPUQ               29.99998
IO1Q               5.00000
IO2Q               19.99998

ELEMENT            MEAN QUEUE LENGTH
CPUQ               0.15607
IO1Q               0.25000
IO2Q               1.00000

ELEMENT            MEAN QUEUEING TIME
CPUQ               5.2023E-03
IO1Q               0.05000
IO2Q               0.05000

ELEMENT            OPEN CHAIN POPULATION
CH1                1.40607

ELEMENT            OPEN CHAIN RESPONSE TIME
CH1                0.28121
```

Notice that the CPU utilization has been reduced by 50 percent. This is because the utilization is equal to the throughput times the service time.

The next modification will be to predict the result of replacing the I/O devices with disks which are 10 percent faster. This changes the values of the I/O service times, and we can obtain the following performance measures.

```
ARRIVLRATE:5
STCPU:.009
STIO1:.036  /* 36   MS PER VISIT */
STIO2:.0225 /* 22.5 MS PER VISIT */
VRCPU:6
```

```
VRIO1:1
VRIO2:4
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT             UTILIZATION
CPUQ                0.27000
IO1Q                0.18000
IO2Q                0.45000

ELEMENT             THROUGHPUT
CPUQ                29.99998
IO1Q                5.00000
IO2Q                19.99998

ELEMENT             MEAN QUEUE LENGTH
CPUQ                0.36986
IO1Q                0.21951
IO2Q                0.81818

ELEMENT             MEAN QUEUEING TIME
CPUQ                0.01233
IO1Q                0.04390
IO2Q                0.04091

ELEMENT             OPEN CHAIN POPULATION
CH1                 1.40756

ELEMENT             OPEN CHAIN RESPONSE TIME
CH1                 0.28151
```

Again notice the ten percent decrease in the utilization of the two I/O devices.

What will happen if we increase the number of terminals by 40 percent? To simplify this modification, we will assume that this will affect only the arrival rate of transactions to the system. This has the hidden assumption that the new terminals will be entering the same types of transactions and that the service times and visit ratios will remain the same as before. Increasing the arrival rate by 40 percent makes the transactions arrive at seven transactions per second.

```
ARRIVLRATE:7 /* TRANSACTIONS PER SECOND */
STCPU:.009
STIO1:.040
STIO2:.025
VRCPU:6
VRIO1:1
VRIO2:4
```

```
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT         UTILIZATION
CPUQ            0.37800
IO1Q            0.28000
IO2Q            0.70000

ELEMENT         THROUGHPUT
CPUQ            41.99997
IO1Q            7.00000
IO2Q            27.99998

ELEMENT         MEAN QUEUE LENGTH
CPUQ            0.60772
IO1Q            0.38889
IO2Q            2.33333

ELEMENT         MEAN QUEUEING TIME
CPUQ            0.01447
IO1Q            0.05556
IO2Q            0.08333

ELEMENT         OPEN CHAIN POPULATION
CH1             3.32993

ELEMENT         OPEN CHAIN RESPONSE TIME
CH1             0.47570
```

Here we see that both the utilizations and the throughputs increase by 40 percent.

Let's replace the two disk drives with a single I/O device which has twice the capacity as each of the previous I/O devices. This will make the visit ratio of the new device equal to five. However, it is not easy to predict what value should be used for the I/O service time. Temporarily, we will use a value of 30 milliseconds, which produces the following results.

```
ARRIVLRATE:5
STCPU:.009
STIO1:.030 /* 30 MS PER VISIT */
STIO2:0 /* THIS DEVICE WAS REMOVED */
VRCPU:6
VRIO1:5 /* VISITS PER TRANSACTION */
VRIO2:0 /* THIS DEVICE WAS REMOVED */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL
```

```
ELEMENT         UTILIZATION
CPUQ            0.27000
IO1Q            0.75000
IO2Q            0.00000

ELEMENT         THROUGHPUT
CPUQ            30.00000
IO1Q            25.00000
IO2Q            0.00000

ELEMENT         MEAN QUEUE LENGTH
CPUQ            0.36986
IO1Q            3.00000
IO2Q            0.00000

ELEMENT         MEAN QUEUEING TIME
CPUQ            0.01233
IO1Q            0.12000
IO2Q            0.00000

ELEMENT         OPEN CHAIN POPULATION
CH1             3.36986

ELEMENT         OPEN CHAIN RESPONSE TIME
CH1             0.67397
```

The utilization at IOQ1 is slightly higher than the sum of the two utilizations at the original I/O devices.

As a final modification, what happens when we eliminate one request per transaction for the second I/O device by making the index portion of one of the device's files resident in memory? This will change the visit ratios for the CPU and the second I/O device to five and three, respectively. Now we must determine what effect this change will have on the CPU service time. It certainly will increase because of the index searching. The exact amount of increase is difficult to determine without some measurement data, but we will use a value of 10.2 ms for the new service time. This produces the following performance measures.

```
ARRIVLRATE:5
STCPU:.0102 /* 10.2 MS PER VISIT */
STIO1:.040
STIO2:.025
VRCPU:5 /* VISITS PER TRANSACTION */
VRIO1:1
VRIO2:3 /* VISITS PER TRANSACTION */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL
```

```
ELEMENT         UTILIZATION
CPUQ            0.25500
IO1Q            0.20000
IO2Q            0.37500

ELEMENT         THROUGHPUT
CPUQ            24.99998
IO1Q            5.00000
IO2Q            15.00000

ELEMENT         MEAN QUEUE LENGTH
CPUQ            0.34228
IO1Q            0.25000
IO2Q            0.60000

ELEMENT         MEAN QUEUEING TIME
CPUQ            0.01369
IO1Q            0.05000
IO2Q            0.04000

ELEMENT         OPEN CHAIN POPULATION
CH1             1.19228

ELEMENT         OPEN CHAIN RESPONSE TIME
CH1             0.23846
```

This modification could also cause a change in the service time of the second I/O device. We have not used a different value for this parameter. Its new value is difficult to predict without some additional measurement data.

Table 7.1 contains a summary of the modification analysis for the baseline case and the five modifications. The parameter values used as input and some of the performance measures have been tabulated.

Parameter	Base	Case 1	Case 2	Case 3	Case 4	Case 5
ARRIVLRATE	5	5	5	7	5	5
STCPU	0.009	0.0045	0.009	0.009	0.009	0.0102
STIO1	0.040	0.040	0.036	0.040	0.030	0.040
STIO2	0.025	0.025	0.0225	0.025	0	0.025
VRCPU	6	6	6	6	6	5
VRIO1	1	1	1	1	5	1
VRIO2	4	4	4	4	0	3
UT(CPU)	0.27	0.135	0.27	0.378	0.27	0.255
UT(IO1)	0.20	0.20	0.18	0.28	0.75	0.20
UT(IO2)	0.50	0.50	0.45	0.70	0	0.375
QL(CPU)	0.37	0.156	0.37	0.608	0.37	0.342
QL(IO1)	0.25	0.25	0.22	0.389	3.00	0.25

QL(IO2)	1.00	1.00	0.818	2.33	0	0.60
Resp. Time	0.324	0.281	0.282	0.476	0.674	0.238

Table 7.1.

This modification analysis illustrates the skills needed by a performance analyst. A skill which is not necessary is an understanding of the mathematical techniques used to calculate the results. What is critical is how the model parameters should be changed to reflect changes in the workload or system configuration. Some parameter values are difficult to determine. This requires an in-depth understanding of the system and often a great deal of intuition.

7.7. SENSITIVITY ANALYSIS

Sometimes an input parameter value can have a significant effect on the results of a model. In this case it is important to perform a sensitivity analysis. This involves trying multiple values for a parameter or set of parameters and observing the magnitude of the change in the performance measures. If the results do not change much, this is not a critical parameter. If the results change quite a bit with small changes in the parameter, the value of this parameter is critical. This would indicate that a further investigation is necessary to obtain an accurate estimate for this parameter.

In the previous section, case four contained a parameter for the I/O service time which we said was temporarily estimated at 30 ms. We will now solve this model and vary the I/O service time from 30 to 40 ms. The following results are obtained.

```
ARRIVLRATE:5
STCPU:.009
STIO1:.03 /* 30 MS PER VISIT */
STIO2:0 /* DEVICE REMOVED */
VRCPU:6
VRIO1:5 /* VISITS PER TRANSACTION */
VRIO2:0 /* DEVICE REMOVED */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT        UTILIZATION
CPUQ           0.27000
IO1Q           0.75000

ELEMENT        THROUGHPUT
CPUQ           30.00000
IO1Q           25.00000
```

```
ELEMENT          MEAN QUEUE LENGTH
CPUQ             0.36986
IO1Q             3.00000

ELEMENT          MEAN QUEUEING TIME
CPUQ             0.01233
IO1Q             0.12000

ELEMENT          OPEN CHAIN POPULATION
CH1              3.36986

ELEMENT          OPEN CHAIN RESPONSE TIME
CH1              0.67397

WHAT:
ARRIVLRATE:5
STCPU:.009
STIO1:.035 /* 35 MS PER VISIT */
STIO2:0 /* DEVICE REMOVED */
VRCPU:6
VRIO1:5 /* VISITS PER TRANSACTION */
VRIO2:0 /* DEVICE REMOVED */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT          UTILIZATION
CPUQ             0.27000
IO1Q             0.87500

ELEMENT          THROUGHPUT
CPUQ             30.00000
IO1Q             25.00000

ELEMENT          MEAN QUEUE LENGTH
CPUQ             0.36986
IO1Q             7.00000

ELEMENT          MEAN QUEUEING TIME
CPUQ             0.01233
IO1Q             0.28000

ELEMENT          OPEN CHAIN POPULATION
CH1              7.36986

ELEMENT          OPEN CHAIN RESPONSE TIME
CH1              1.47397

WHAT:
ARRIVLRATE:5
STCPU:.009
```

```
STIO1:.039 /* 39 MS PER VISIT */
STIO2:0 /* DEVICE REMOVED */
VRCPU:6
VRIO1:5 /* VISITS PER TRANSACTION */
VRIO2:0 /* DEVICE REMOVED */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT        UTILIZATION
CPUQ           0.27000
IO1Q           0.97500

ELEMENT        THROUGHPUT
CPUQ           30.00000
IO1Q           25.00000

ELEMENT        MEAN QUEUE LENGTH
CPUQ           0.36986
IO1Q           39.00020

ELEMENT        MEAN QUEUEING TIME
CPUQ           0.01233
IO1Q           1.56001

ELEMENT        OPEN CHAIN POPULATION
CH1            39.37006

ELEMENT        OPEN CHAIN RESPONSE TIME
CH1            7.87401

WHAT:
ARRIVLRATE:5
STCPU:.009
STIO1:.04 /* 40 MS PER VISIT */
STIO2:0 /* DEVICE REMOVED */
VRCPU:6
VRIO1:5 /* VISITS PER TRANSACTION */
VRIO2:0 /* DEVICE REMOVED */
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT        UTILIZATION
CPUQ           0.27000
IO1Q           1.00000

ELEMENT        THROUGHPUT
CPUQ           30.00000
IO1Q           25.00000
```

```
ELEMENT         MEAN QUEUE LENGTH
CPUQ            0.36986
IO1Q            9.4520E+06

ELEMENT         MEAN QUEUEING TIME
CPUQ            0.01233
IO1Q            3.7808E+05

ELEMENT         OPEN CHAIN POPULATION
CH1             9.4520E+06

ELEMENT         OPEN CHAIN RESPONSE TIME
CH1             1.8904E+06
```

The results vary significantly with different values of the I/O service time parameter. This would indicate that the I/O service time is a critical parameter for this model. We should obtain an accurate estimate of it in order to obtain accurate performance measures.

7.8. PLOTTING OF RESULTS

Models are frequently solved for a large set of parameter values. One of the best ways of viewing a large collection of results from many different parameter values is using graphics. The graphical plots of model results can sometimes give further insight into the system behavior. In this section some typical plots will be illustrated. There are many other types of plots which analysts may find useful.

Let us start out by plotting some results from model EX7.2 which was discussed in the previous two sections. We have solved this model by varying the arrival rate (ARRIVLRATE) from 5.0 to 9.5 in increments of 0.5. The plots in Figure 7.3 illustrate the changes in utilization, throughput, mean queue length, and mean queueing time with the arrival rate.

Look back at the results listed for the M/M/1 queue model discussed in Section 7.1. The following two plots in Figure 7.4 show the queue length distribution and the queueing time distribution.

7.9. FURTHER READING

Additional information about performance measures and the model from Section 7.3 can be found in Sauer, MacNair, and Kurose [159, 160]. For a further discussion of the sequential stopping rule and the small sample

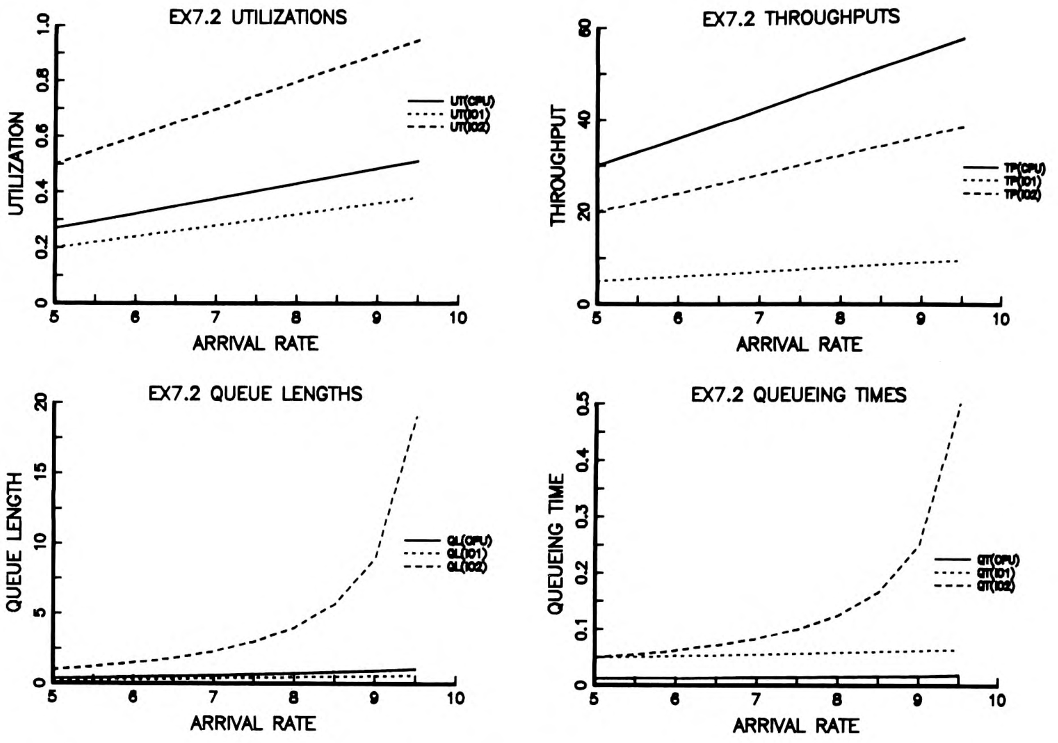


Figure 7.3. Utilization, Throughput, Queue Length, and Queueing Time

problems see Lavenberg and Sauer [102]. Buzen [40] contains a good discussion of modification analysis and the skills necessary for modeling. The model and parameters in Sections 7.6 and 7.7 are based on descriptions found in this same paper by Buzen. Lazowska, Zahorjan, Graham, and Sevcik [108] has additional information on modification analysis and validation of analytic models. Law and Kelton [106] has a good discussion of accuracy and validation of simulation models. Kobayashi [98] and Lavenberg [100] contain additional material related to the level of detail of models. MacNair and Sauer [115] illustrates some additional examples of graphical results.

7.10. EXERCISES

7.1 Discuss the various performance measures produced by modeling packages.

7.2 Discuss the sources of error that may exist in model results.

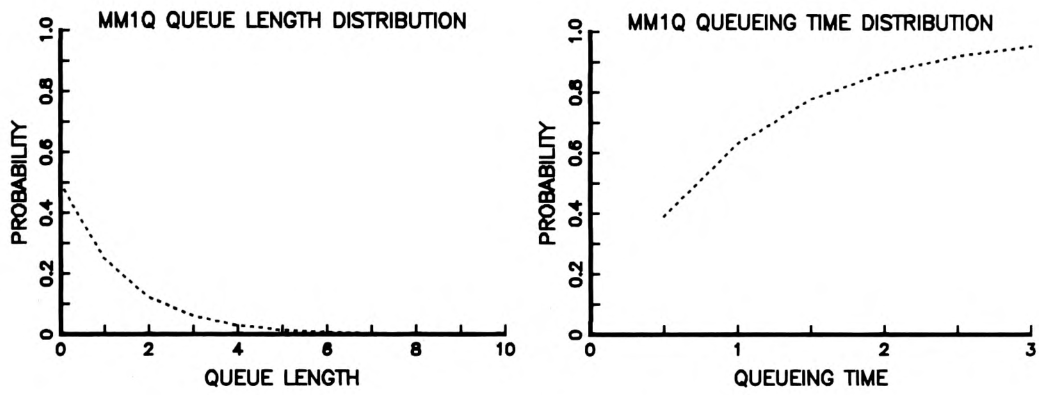


Figure 7.4. Queue Length and Queueing Time Distributions

7.3 Discuss the role confidence intervals play in determining the accuracy of simulation results.

7.4 Try to validate a simple model of a system you are familiar with.

7.5 Construct a model of a system at several different levels of detail and compare the results.

7.6 Perform a modification analysis and a sensitivity analysis on a model of your choice.

7.7 Plot some of the results obtained from a parametric study of a model of your choice.

CHAPTER 8

EVERYDAY LIFE SYSTEMS

This chapter contains models of simple systems people encounter in day to day activities. These models are being presented as a teaching aid to demonstrate how to formulate a model once we understand how a system behaves. The systems modeled in this chapter were chosen because most people are familiar with the way they function. The systems include a barber shop, a parking lot, a traffic light, use of a copier, a catalog store, and a supermarket.

8.1. BARBER SHOP

A barber shop is a very simple system. The barbers constitute the servers and the chairs can be represented by positions in a waiting line. Figure 8.1 is a schematic of a typical barber shop.

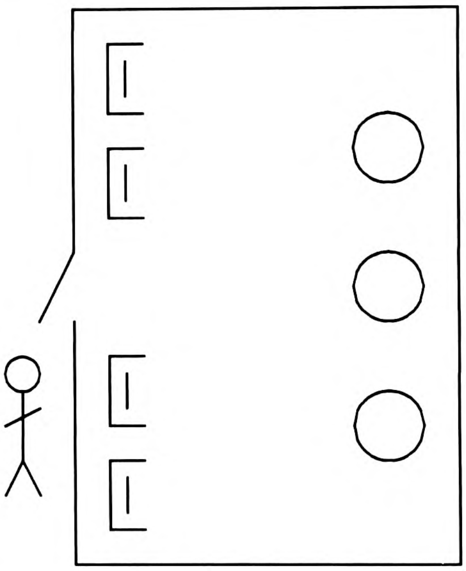


Figure 8.1. Schematic of a Barber Shop

It can be modeled as an open model with a source generating the arrivals of new customers. Figure 8.2 illustrates a model diagram of this system. The people arrive according to some interarrival time distribution. If there are any empty seats available, the new customer enters the barber shop. If the shop is full, the customer leaves. The customers are served in FCFS order by the first available barber.

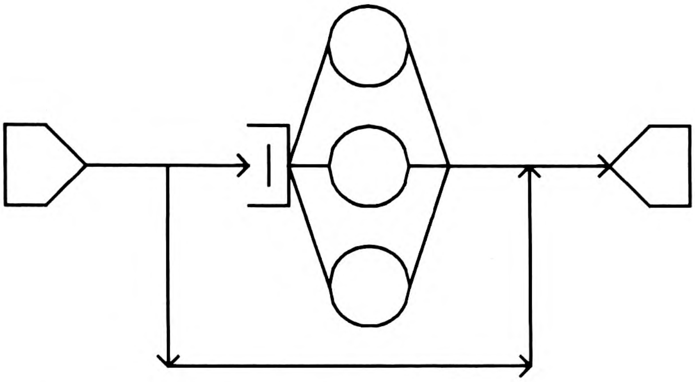


Figure 8.2. Model Diagram of a Barber Shop

The following listing is a simple RESQ model of this system. It will first be solved by simulation, and then we will compare it to an analytic solution. It contains symbolic input parameters for the rate of arrival of new customers, the time to cut a person's hair, the number of barbers, and the number of chairs. The arrival rate and the cutting time are mean values for exponential distributions.

```
MODEL:EX8.1
   METHOD:simulation
   NUMERIC PARAMETERS:arrivlrate cuttingtim numbarbers numchairs
      /* arrivlrate - arrival rate, people per hour */
      /* cuttingtim - cutting time in minutes       */
      /* numbarbers - number of barbers             */
      /* numchairs  - number of waiting chairs      */
   QUEUE:barbers
      TYPE:active
      SERVERS:numbarbers
      DSPL:fcfs
      CLASS LIST:chairs
         WORK DEMANDS:cuttingtim
      SERVER -
         RATES:1
   CHAIN:path
      TYPE:open
      SOURCE LIST:people
      ARRIVAL TIMES:60/arrivlrate   /* minutes between arrivals */
      :people->chairs sink;if(ql<numchairs+numbarbers) if(t)
      :chairs->sink
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   RUN LIMITS -
      QUEUES FOR DEPARTURE COUNTS:barbers
         DEPARTURES:1000
   LIMIT - CP SECONDS:5
   TRACE:no
END
```

The barbers and the chairs are modeled as a single service center with multiple servers. The model permits the arrival rate to be entered as the number of people per hour, which is converted to minutes between arrivals. If all of the chairs, and therefore the barbers, are occupied when a new customer arrives, the new customer leaves the model by going to the SINK. The simulation will not produce confidence intervals. It will be run until there are 1,000 departures from the shop.

We can solve this model and specify different arrival rates, cutting times, numbers of barbers, and numbers of chairs. The parameters for the first solution are 17 people per hour, ten minutes per haircut, five barbers, and ten chairs. We can obtain the following results from the simulation.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 09:23:17  DATE: 02/26/84
MODEL:EX8.1
ARRIVLRATE:17
CUTTINGTIM:10
NUMBARBERS:5
NUMCHAIRS:10
RUN END: BARBERS DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:      3553.03662
                          CPU TIME:            0.75
                  NUMBER OF EVENTS:            2004

WHAT:ALL

ELEMENT         UTILIZATION
BARBERS         0.54471
 SERVER    1     0.76568
 SERVER    2     0.64583
 SERVER    3     0.56723
 SERVER    4     0.41837
 SERVER    5     0.32642

ELEMENT         THROUGHPUT
BARBERS         0.28145
PEOPLE          0.28258
SINK            0.28145
```

```
ELEMENT        MEAN QUEUE LENGTH
BARBERS        2.88230

ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
BARBERS        1.87379

ELEMENT        MEAN QUEUEING TIME
BARBERS        10.21078

ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
BARBERS        10.05878

ELEMENT        MAXIMUM QUEUE LENGTH
BARBERS        10

ELEMENT        MAXIMUM QUEUEING TIME
BARBERS        67.45689

ELEMENT        OPEN CHAIN POPULATION
PATH           2.88230

ELEMENT        OPEN CHAIN RESPONSE TIME
PATH           10.24092

WHAT:
CONTINUE RUN:no
```

If the new customers were not leaving when all of the chairs are occupied, the actual throughput would be 17 people per hour divided by 60 minutes per hour which is 0.28333. This is a very short run, but the simulation throughput at the source (PEOPLE) is close to the actual value. The throughput at the barbers should be less than this value. The actual utilization is equal to the throughput times the service time divided by the number of servers. It should be less than 0.56667. Notice the server utilizations reported by the simulation program. When there are several servers free, the simulation always selects the first server to begin service. This is the reason for the decreasing server utilizations.

Now we will change the input parameters to a new set of values and solve the model again.

```
ARRIVLRATE:6
CUTTINGTIM:11
NUMBARBERS:3
NUMCHAIRS:4
RUN END: BARBERS DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.
```

```
                    SIMULATED TIME:        1.0060E+04
                          CPU TIME:              0.70
                  NUMBER OF EVENTS:              2002


WHAT:ALL

ELEMENT              UTILIZATION
BARBERS              0.35351
 SERVER    1          0.54293
 SERVER    2          0.33930
 SERVER    3          0.17829

ELEMENT              THROUGHPUT
BARBERS              0.09940
PEOPLE               0.09960
SINK                 0.09940

ELEMENT              MEAN QUEUE LENGTH
BARBERS              1.10741

ELEMENT              STANDARD DEVIATION OF QUEUE LENGTH
BARBERS              1.10274

ELEMENT              MEAN QUEUEING TIME
BARBERS              11.11303

ELEMENT              STANDARD DEVIATION OF QUEUEING TIME
BARBERS              11.05664

ELEMENT              MAXIMUM QUEUE LENGTH
BARBERS              6

ELEMENT              MAXIMUM QUEUEING TIME
BARBERS              74.20258

ELEMENT              OPEN CHAIN POPULATION
PATH                 1.10741

ELEMENT              OPEN CHAIN RESPONSE TIME
PATH                 11.14108
```

Again we could easily calculate the actual throughput at the source and approximate values for the utilization and throughput at the barbers. Instead we will change the model so that we can solve it analytically. The following model replaces the source with a FCFS service center and the open chain with a closed chain. The finite capacity of the barber shop is represented by the closed chain population. This is a cyclic queueing model because the customers keep cycling back through the service centers. The results produced at the BARBERS service center are comparable for both models.

```
MODEL:EX8.2
   METHOD:numerical
   NUMERIC PARAMETERS:arrivlrate cuttingtim numbarbers numchairs
   QUEUE:sourceq
      TYPE:fcfs
      CLASS LIST:people
         SERVICE TIMES:60/arrivlrate /* min. between arrivals */
   QUEUE:barbers
      TYPE:active
      SERVERS:numbarbers
      DSPL:fcfs
      CLASS LIST:chairs
         WORK DEMANDS:cuttingtim
      SERVER -
         RATES:1
   CHAIN:path
      TYPE:closed
      POPULATION:numchairs+numbarbers
      :people->chairs->people
END
```

Here are the performance measures from the analytic solution with the same set of parameter values.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 09:59:20  DATE: 02/26/84
MODEL:EX8.2
ARRIVLRATE:17
CUTTINGTIM:10
NUMBARBERS:5
NUMCHAIRS:10
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT        UTILIZATION
SOURCEQ        0.99971
BARBERS        0.56650

ELEMENT        THROUGHPUT
SOURCEQ        0.28325
BARBERS        0.28325

ELEMENT        MEAN QUEUE LENGTH
SOURCEQ        11.91461
BARBERS        3.08539

ELEMENT        MEAN QUEUEING TIME
SOURCEQ        42.06383
```

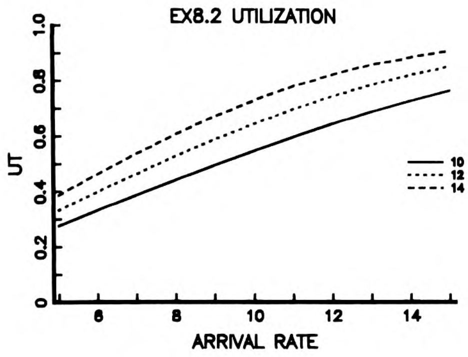
```
BARBERS          10.89277

WHAT:
ARRIVLRATE:6
CUTTINGTIM:11
NUMBARBERS:3
NUMCHAIRS:4
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT              UTILIZATION
SOURCEQ              0.99869
BARBERS              0.36619

ELEMENT              THROUGHPUT
SOURCEQ              0.09987
BARBERS              0.09987

ELEMENT              MEAN QUEUE LENGTH
SOURCEQ              5.83926
BARBERS              1.16074

ELEMENT              MEAN QUEUEING TIME
SOURCEQ              58.46945
BARBERS              11.62262
```

This technique of using a cyclic queueing model can be used to exactly represent a finite capacity, single-resource system. The population in the closed model is used to depict the finite capacity.

Next we will solve this model by varying the arrival rate from five to 15 people per hour and the cutting time from ten to 14 minutes with three barbers and four chairs. The following graphs in Figure 8.3 show the utilization of the barbers and the average amount of time spent in the barber shop.

8.2. PARKING LOT

We will build a simple model of a parking lot in order to illustrate how to use a passive resource. The passive resource contains a finite number of elements which are allocated to customers, held onto by the customers, and finally released by the customers. The finite number of elements will represent the number of spaces in the parking lot. Figure 8.4 shows a model diagram of the parking lot model.



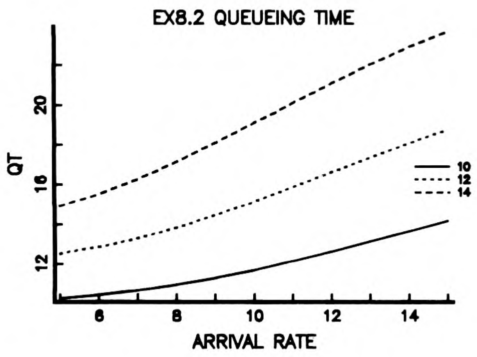


Figure 8.3. Graphs of Utilization and Queueing Time

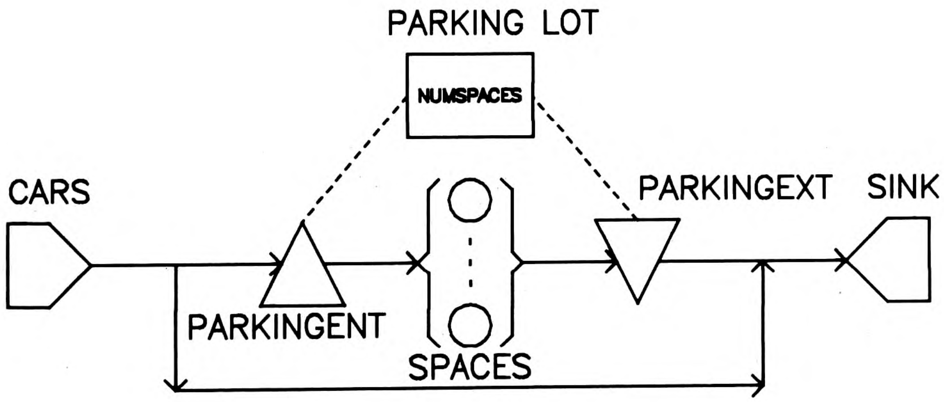


Figure 8.4. Model Diagram of a Parking Lot

There is a passive resource for the number of spaces in the lot. The number of tokens is equal to the number of spaces. One space is allocated at PARKINGENT and is held onto until it is released at PARKINGEXT. There is an active service center which represents the time a customer spends shopping while the car occupies a space in the parking lot. Cars are generated at a source and enter the parking lot if a space is available. A status function (TA) for the number of tokens available is checked to determine this condition. If all spaces are in use, the car leaves the model. After being allocated a space, the car holds onto the space until the service time at the SPACEQ service center is complete. The SPACEQ service center is modeled as an infinite server so that the occupants of the cars can be shopping at the same time. Then the car releases the space and leaves. The released space is then available to be allocated to new cars that arrive. This model contains numeric parameters for the number of spaces in the lot, the average amount of time a space is occupied, and the average rate of arrival.

```
MODEL:EX8.3
   METHOD:simulation
   NUMERIC PARAMETERS:numspaces spacetime arrivlrate
   /* numspaces  = the number of spaces in the parking lot */
   /* spacetime  = the average number of minutes a car spends */
   /*              in a space */
   /* arrivlrate = the average number of cars arriving per hour */
   QUEUE:spaceq
      TYPE:is
      CLASS LIST:spaces
         SERVICE TIMES:spacetime
   QUEUE:parkinglot
      TYPE:passive
      TOKENS:numspaces
      DSPL:fcfs
      ALLOCATE NODE LIST:parkingent
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:parkingext
   CHAIN:carpath
      TYPE:open
      SOURCE LIST:cars
      ARRIVAL TIMES:60/arrivlrate  /* inter-arrival time, minutes */
      :cars->parkingent sink;if(ta>0) if(t)
      :parkingent->spaces->parkingext->sink
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   RUN LIMITS -
      SIMULATED TIME:480
      QUEUES FOR DEPARTURE COUNTS:parkinglot
         DEPARTURES:1000
   LIMIT - CP SECONDS:5
   TRACE:no
END
```

When we simulate this model with 100 spaces, an average of 20 minutes occupying a space and 270 cars per hour, we obtain the following results. On the average only 79 spaces are occupied. Notice that the mean queueing times are less than the 20 minutes which was specified as an input parameter. This is a very short run of the simulation, and the results are not representative of the actual performance measures. The simulation would have to be continued to obtain more accurate results.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 19:20:15  DATE: 02/29/84
MODEL:PARKINGL
NUMSPACES:100
SPACETIME:20
ARRIVLRATE:270
RUN END: PARKINGLOT DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:       242.20998
                          CPU TIME:            1.44
                  NUMBER OF EVENTS:            2080


WHAT:ALL

ELEMENT         UTILIZATION
PARKINGLOT      0.78691
SPACEQ          0.00000

ELEMENT         THROUGHPUT
PARKINGLOT      4.12865
SPACEQ          4.12865
PARKINGEXT      4.12865
CARS            4.45894
SINK            4.15342

ELEMENT         MEAN QUEUE LENGTH
PARKINGLOT      78.69113
SPACEQ          78.69113

ELEMENT         STANDARD DEVIATION OF QUEUE LENGTH
PARKINGLOT      15.46044
SPACEQ          15.46044

ELEMENT         MEAN QUEUEING TIME
PARKINGLOT      17.76599
SPACEQ          17.76599

ELEMENT         STANDARD DEVIATION OF QUEUEING TIME
PARKINGLOT      18.32513
SPACEQ          18.32513

ELEMENT         MEAN TOKENS IN USE
PARKINGLOT      78.69113
```

```
ELEMENT          MEAN TOTAL TOKENS IN POOL
PARKINGLOT       100.00000

ELEMENT          MAXIMUM QUEUE LENGTH
PARKINGLOT       100
SPACEQ           100

ELEMENT          MAXIMUM QUEUEING TIME
PARKINGLOT       134.91374
SPACEQ           134.91374

ELEMENT          OPEN CHAIN POPULATION
CARPATH          78.69113

ELEMENT          OPEN CHAIN RESPONSE TIME
CARPATH          18.94609
```

8.3. TRAFFIC LIGHT

We will construct a simple model of an intersection with traffic flowing in one direction. Figure 8.5 shows a model diagram of this system. Cars are generated at a source and wait at an allocate node if the light is red. When the light becomes green, cars go through the intersection one at a time. The light is changed to red by allocating the token representing the light to a special customer, which holds onto it for a time period representing the red light. Then the token is released by this special customer, and the special customer spends time at a service center for the amount of time the light stays green.

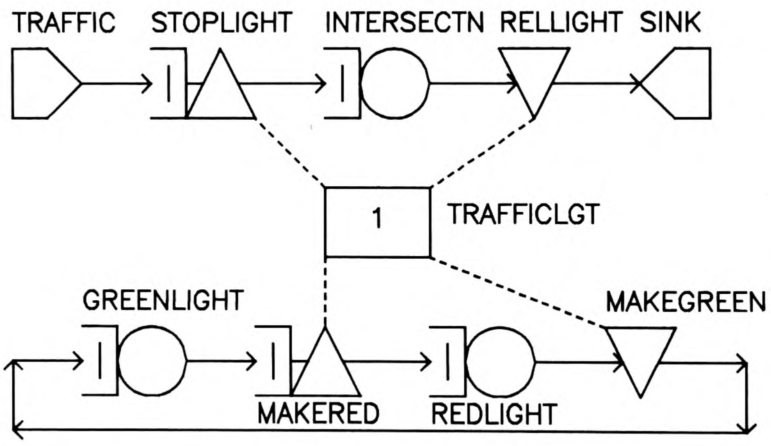


Figure 8.5. Model Diagram of a Traffic Light

The model contains numeric parameters for the average amount of time it takes a car to get through the intersection, the average length of a green light, the average number of cars to arrive per minute, and the average length of a red light. The passive resource representing the traffic light has one token, two allocate nodes, and two release nodes. One pair of allocate and release nodes (STOPLIGHT and RELLIGHT) is for the cars, and the other pair of allocate and release nodes (MAKERED and MAKEGREEN) is for the special customer that controls the light. The passive resource has a priority queueing discipline, with the special customer having priority over the cars. There are service centers for the intersection time, the green light and the red light. The lengths of the green and red lights are specified to be constants. The intersection time and the interarrival time between cars are specified as exponential distributions. This simple technique of synchronizing customers with a passive resource will be very useful in many other modeling situations.

```
MODEL:EX8.4
   METHOD:simulation
   NUMERIC PARAMETERS:intersectm greentime arrivlrate redtime
   /* intersectm = average amount of time a car spends in the */
   /*              intersection, in seconds */
   /* greentime  = length of a green light, in seconds */
   /* arrivlrate = average number of cars which arrive in 1 minute */
   /* redtime    = length of a red light, in seconds */
   QUEUE:trafficlgt
      TYPE:passive
      TOKENS:1
      DSPL:prty
      ALLOCATE NODE LIST:stoplight makered
         NUMBERS OF TOKENS TO ALLOCATE:1 1
         PRIORITIES:2 1
      RELEASE NODE LIST:rellight makegreen
   QUEUE:intersectq
      TYPE:fcfs
      CLASS LIST:intersectn
         SERVICE TIMES:intersectm
   QUEUE:greenq
      TYPE:fcfs
      CLASS LIST:greenlight
         SERVICE TIMES:constant(greentime)
   QUEUE:redq
      TYPE:fcfs
      CLASS LIST:redlight
         SERVICE TIMES:constant(redtime)
   CHAIN:trafpath
      TYPE:open
      SOURCE LIST:traffic
      ARRIVAL TIMES:60/arrivlrate
      :traffic->stoplight->intersectn->rellight->sink
      :greenlight->makered->redlight->makegreen->greenlight
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   CHAIN:trafpath
   NODE LIST:greenlight
      INIT POP:1
   RUN LIMITS -
      QUEUES FOR DEPARTURE COUNTS:intersectq
         DEPARTURES:1000
   LIMIT - CP SECONDS:5
   TRACE:no
END
```

The following results were produced by assigning the intersection time to two seconds, the green time to 20 seconds, the red time to 30 seconds, and the arrival rate to ten cars per minute. Again we are running the simulation for a very short amount of time. On the average there are about four cars waiting at a red light or in the intersection if the light is green. The average amount of time a car spends waiting for a red light is about 22.5 (24.5 minus 1.95) seconds.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 18:57:29  DATE: 02/28/84
MODEL:EX8.4
INTERSECTM:2
GREENTIME:20
ARRIVLRATE:10
REDTIME:30
RUN END: INTERSECTQ DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:     6040.91406
                          CPU TIME:           0.99
                  NUMBER OF EVENTS:           2241

WHAT:ALL

ELEMENT        UTILIZATION
TRAFFICLGT     0.90840
 STOPLIGHT      0.32240
 MAKERED        0.58600
INTERSECTQ     0.32240
GREENQ         0.39292
REDQ           0.58600

ELEMENT        THROUGHPUT
TRAFFICLGT     0.18507
 STOPLIGHT      0.16554
 MAKERED        0.01953
INTERSECTQ     0.16554
GREENQ         0.01953
REDQ           0.01953
RELLIGHT       0.16554
```

```
MAKEGREEN       0.01953
TRAFFIC         0.16637
SINK            0.16554

ELEMENT         MEAN QUEUE LENGTH
TRAFFICLGT      4.67230
 STOPLIGHT       4.06522
 MAKERED         0.60708
INTERSECTQ      0.32240
GREENQ          0.39292
REDQ            0.58600

ELEMENT         STANDARD DEVIATION OF QUEUE LENGTH
TRAFFICLGT      3.85954
 STOPLIGHT       3.81547
 MAKERED         0.48840
INTERSECTQ      0.46739
GREENQ          0.48840
REDQ            0.49255

ELEMENT         MEAN QUEUEING TIME
TRAFFICLGT      25.22060
 STOPLIGHT       24.52933
 MAKERED         31.07884
INTERSECTQ      1.94756
GREENQ          20.00000
REDQ            30.00000

ELEMENT         STANDARD DEVIATION OF QUEUEING TIME
TRAFFICLGT      18.64174
 STOPLIGHT       19.58708
 MAKERED         1.69751
INTERSECTQ      1.99970

ELEMENT         MEAN TOKENS IN USE
TRAFFICLGT      0.90840

ELEMENT         MEAN TOTAL TOKENS IN POOL
TRAFFICLGT      1.00000

ELEMENT         MAXIMUM QUEUE LENGTH
TRAFFICLGT      23
 STOPLIGHT       22
 MAKERED         1
INTERSECTQ      1
GREENQ          1
REDQ            1

ELEMENT         MAXIMUM QUEUEING TIME
TRAFFICLGT      110.01151
 STOPLIGHT       110.01151
 MAKERED         39.51199
```

```
INTERSECTQ        13.49137
GREENQ            20.00000
REDQ              30.00000

ELEMENT           OPEN CHAIN POPULATION
TRAFPATH          5.06522

ELEMENT           OPEN CHAIN RESPONSE TIME
TRAFPATH          30.59857
```

8.4. COPIER

This section discusses a model of using a copier. Figure 8.6 shows a model diagram. People arrive at a source according to some interarrival time distribution. There is a passive resource with one token to insure that only one person can use the copier at a time. There is a set node to assign the number of copies to a customer attribute by sampling from a distribution. If the number of blank sheets remaining in the copier is greater than or equal to the number of copies, the sheets are allocated to the person and time is spent copying the previously assigned number of copies. A destroy node discards the copied sheets so that the number of blank sheets remaining is correct. Remember that a customer which visits a destroy node discards all the tokens it is holding. Then any customer waiting can use the copier when the previous person leaves. If the number of copies exceeds the number of sheets remaining, the person loads the copier with new blank sheets before proceeding to perform the copying.

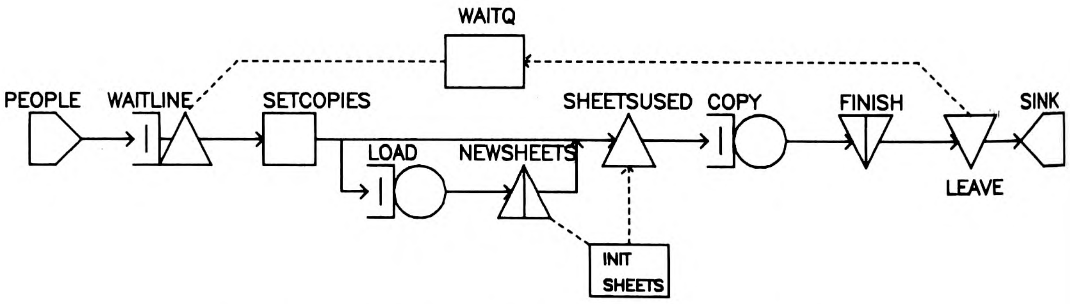


Figure 8.6. Model Diagram of Using a Copier

The model contains numeric parameters for the rate of arrival of the people, the mean number of copies made, the average amount of time to load the copier with blank sheets, the number of blank sheets initially in the copier, the amount of time to copy one sheet, and the number of simulation minutes per sequential sampling period. The passive resource for restricting the use of the copier defines an allocate node, a release node, and one token. The time to load the blank sheets is assumed to be from an exponential distribution. The passive resource controlling the use of the blank sheets contains a release node, a destroy node, a create node, and a number of tokens equal to a previously defined numeric parameter (INITSHEETS). The number of sheets allocated to a customer is equal to a customer attribute (in RESQ, it is a job variable (JV(0)) which is assigned as a sample from an exponential distribution at a set node. The number of sheets created when the copier is about to be empty is a sample from a discrete distribution. Either 100, 200, or 300 sheets are loaded with the given probabilities. The copying time is equal to the number of copies times the amount of time necessary to copy one sheet. We are using the regenerative method to produce confidence intervals and the sequential sampling procedure to detect when the accuracy criteria are satisfied.

```
MODEL:EX8.5
   METHOD:simulation
   NUMERIC PARAMETERS:peoplear mnumcopies loadtime
      /* peoplear   = number of people per minute */
      /* mnumcopies = mean number of copies made  */
      /* loadtime   = average amount of seconds to load */
   NUMERIC PARAMETERS:initsheets copytime stperperid
      /* initsheets = initial number of sheets in copier */
      /* copytime   = number of seconds to copy 1 sheet */
      /* stperperid = simulation minutes per period */
   QUEUE:waitq
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:waitline
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:leave
   QUEUE:loadq
      TYPE:fcfs
      CLASS LIST:load
         SERVICE TIMES:loadtime
   QUEUE:sheetsq
      TYPE:passive
      TOKENS:initsheets
      DSPL:fcfs
      ALLOCATE NODE LIST:sheetsused
         NUMBERS OF TOKENS TO ALLOCATE:jv(0)
      DESTROY NODE LIST:finish
      CREATE NODE LIST:newsheets
```

```
         NUMBERS OF TOKENS TO CREATE:discrete(100,.4;200,.3;300,.3)
   QUEUE:copyq
      TYPE:fcfs
      CLASS LIST:copy
         SERVICE TIMES:constant(jv(0)*copytime)
   SET NODES:setcopies
      ASSIGNMENT LIST:jv(0)=exponential(mnumcopies)
      /* jv(0)       = number of copies for each person */
   CHAIN:peoplepath
      TYPE:open
      SOURCE LIST:people
      ARRIVAL TIMES:60/peoplear   /* seconds between arrivals */
      :people->waitline->setcopies
      :setcopies->sheetsused load;if(jv(0)<ta) if(t)
      :load->newsheets->sheetsused->copy->finish->leave->sink
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:sheetsq
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      SIMULATED TIME:60*stperperid  /* seconds */
   LIMIT - CP SECONDS:30
   TRACE:no
END
```

The following results are produced for the parameter values shown. This is a very short simulation run, and some of the confidence interval widths are very large. There is not much contention with this set of parameter values. The maximum queueing time of more than five minutes is caused by a long load time. Notice that it was necessary to load the copier 53 times during this run.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 07:54:53  DATE: 03/07/84
MODEL:EX8.5
PEOPLEAR:0.5
MNUMCOPIES:10
LOADTIME:60
INITSHEETS:300
COPYTIME:0.5
STPERPERID:400
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.
```

```
                   SIMULATED TIME:     1.2024E+05
                         CPU TIME:           2.62
                 NUMBER OF EVENTS:           2045
                 NUMBER OF CYCLES:            939

WHAT:ALLBO

ELEMENT        UTILIZATION
WAITQ          0.06531(0.05670,0.07393) 1.7%
LOADQ          0.02322(-0.00801,0.05444) 6.2%
COPYQ          0.04210(0.03906,0.04514) 0.6%

ELEMENT        THROUGHPUT
WAITQ          8.2834E-03(7.8471E-03,8.7196E-03)  10.5%
SHEETSQ        7.8842E-03(7.4562E-03,8.3122E-03)  10.9%
LOADQ          4.4078E-04(2.9473E-05,8.5209E-04)  186.6%
COPYQ          8.2834E-03(7.8471E-03,8.7196E-03)  10.5%
LEAVE          8.2834E-03
FINISH         8.2834E-03
NEWSHEETS      4.4078E-04
SETCOPIES      8.2834E-03
PEOPLE         8.2834E-03
SINK           8.2834E-03

ELEMENT        MEAN QUEUE LENGTH
WAITQ          0.07718(0.06279,0.09157) 37.3%
SHEETSQ        0.04205(0.03894,0.04516) 14.8%
LOADQ          0.02322(-0.00801,0.05444) 269.0%
COPYQ          0.04210(0.03906,0.04514) 14.4%

ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
WAITQ          0.31875
SHEETSQ        0.20070
LOADQ          0.15059
COPYQ          0.20081

ELEMENT        MEAN QUEUEING TIME
WAITQ          9.31745(7.67765,10.95725) 35.2%
SHEETSQ        5.33341(5.06968,5.59714) 9.9%
LOADQ          52.67407(40.19261,65.15552) 47.4%
COPYQ          5.08190(4.82670,5.33709) 10.0%

ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
WAITQ          21.55467
SHEETSQ        4.89252
LOADQ          55.17625
COPYQ          4.90234

ELEMENT        MEAN TOKENS IN USE
WAITQ          0.06531(0.05670,0.07393) 26.4%
SHEETSQ        0.82559(0.72304,0.92814) 24.8%
```

```
ELEMENT          MEAN TOTAL TOKENS IN POOL
WAITQ            1.00000
SHEETSQ          109.07230(103.04735,115.09724) 11.0%

ELEMENT          MAXIMUM QUEUE LENGTH
WAITQ            4
SHEETSQ          1
LOADQ            1
COPYQ            1

ELEMENT          MAXIMUM QUEUEING TIME
WAITQ            304.10132
SHEETSQ          36.62265
LOADQ            301.45630
COPYQ            36.62265

ELEMENT          OPEN CHAIN POPULATION
PEOPLEPATH       0.07718(0.06279,0.09157) 37.3%

ELEMENT          OPEN CHAIN RESPONSE TIME
PEOPLEPATH       9.31745(7.67765,10.95725) 35.2%

WHAT:ND(*)

ELEMENT          NUMBER OF DEPARTURES
WAITQ            996
SHEETSQ          948
LOADQ            53
COPYQ            996
LEAVE            996
FINISH           996
NEWSHEETS        53
SETCOPIES        996
PEOPLE           996
SINK             996

WHAT:
CONTINUE RUN:no
```

If we triple the rate at which people arrive to use the copier, we obtain the following results. There is more contention exhibited in these results compared to the ones with the previous arrival rate.

```
PEOPLEAR:1.5
MNUMCOPIES:10
LOADTIME:60
INITSHEETS:300
COPYTIME:0.5
STPERPERID:400
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
SAMPLING PERIOD END: SIMULATED TIME GUIDELINE
```

```
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:      4.8739E+04
                          CPU TIME:            3.05
                  NUMBER OF EVENTS:            2503
                  NUMBER OF CYCLES:             974

WHAT:ALLBO

ELEMENT        UTILIZATION
WAITQ          0.20563(0.17595,0.23532) 5.9%
LOADQ          0.07768(-0.02337,0.17873) 20.2%
COPYQ          0.12795(0.11921,0.13669) 1.7%

ELEMENT        THROUGHPUT
WAITQ          0.02505(0.02384,0.02626) 9.7%
SHEETSQ        0.02382(0.02262,0.02502) 10.1%
LOADQ          1.2516E-03(2.3253E-04,2.2706E-03) 162.8%
COPYQ          0.02505(0.02384,0.02626) 9.7%
LEAVE          0.02505
FINISH         0.02505
NEWSHEETS      1.2516E-03
SETCOPIES      0.02505
PEOPLE         0.02505
SINK           0.02505

ELEMENT        MEAN QUEUE LENGTH
WAITQ          0.41951(0.25452,0.58450) 78.7%
SHEETSQ        0.12780(0.11888,0.13672) 14.0%
LOADQ          0.07768(-0.02337,0.17873) 260.2%
COPYQ          0.12795(0.11921,0.13669) 13.7%

ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
WAITQ          1.21415
SHEETSQ        0.33387
LOADQ          0.26767
COPYQ          0.33403

ELEMENT        MEAN QUEUEING TIME
WAITQ          16.74559(10.30549,23.18570) 76.9%
SHEETSQ        5.36516(5.11409,5.61622) 9.4%
LOADQ          62.06848(46.62164,77.51532) 49.8%
COPYQ          5.10742(4.86712,5.34772) 9.4%

ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
WAITQ          42.05983
SHEETSQ        5.07177
LOADQ          65.92419
COPYQ          5.07391

ELEMENT        MEAN TOKENS IN USE
WAITQ          0.20563(0.17595,0.23532) 28.9%
```

```
SHEETSQ          2.59484(2.26255,2.92713) 25.6%

ELEMENT          MEAN TOTAL TOKENS IN POOL
WAITQ            1.00000(1.00000,1.00000) 0.0%
SHEETSQ          105.14462(99.44360,110.84564) 10.8%

ELEMENT          MAXIMUM QUEUE LENGTH
WAITQ            13
SHEETSQ          1
LOADQ            1
COPYQ            1

ELEMENT          MAXIMUM QUEUEING TIME
WAITQ            448.45972
SHEETSQ          37.66333
LOADQ            405.26050
COPYQ            37.66333

ELEMENT          OPEN CHAIN POPULATION
PEOPLEPATH       0.41951(0.25452,0.58450) 78.7%

ELEMENT          OPEN CHAIN RESPONSE TIME
PEOPLEPATH       16.74559(10.30548,23.18568) 76.9%

WHAT:ND(*)

ELEMENT          NUMBER OF DEPARTURES
WAITQ            1221
SHEETSQ          1161
LOADQ            61
COPYQ            1221
LEAVE            1221
FINISH           1221
NEWSHEETS        61
SETCOPIES        1221
PEOPLE           1221
SINK             1221
```

8.5. CATALOG STORE

The model discussed in this section will be more complicated than the ones discussed in the previous sections in this chapter. It is a model of a catalog store which sells items from a catalog. People call the store or come in to place orders or pick them up when they are ready. In addition to answering the telephone calls and waiting on the people who come in, the clerks who work at the store must place orders which are ready for delivery in bins in anticipation of customer pickup. This is called a binning operation. The store is in operation for nine hours, but the desk is only open for the last seven hours. No new requests are accepted after nine hours, but the ones present are completed.

Figure 8.7 shows four sources. Three of them are for three different types of requests: a phone call, a person arriving at the desk, and a binning request. Each customer is assigned a number to identify the type of request. There are four clerks available for handling the requests. The phone and desk requests take precedence over the binning requests, until eight hours have transpired. After eight hours, the binning requests are processed the same as the other types of requests. Any binning requests which are waiting after the eight hours at an allocate node with lower priority than the phone and desk requests are moved to the line with the other requests by creating a sufficient number of tokens to allocate to the waiting binning requests.

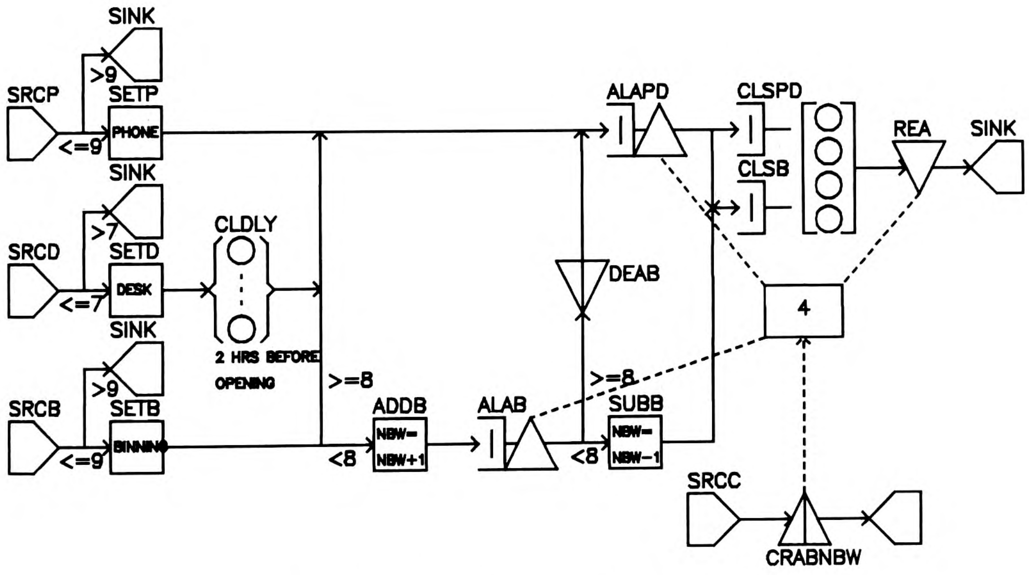


Figure 8.7. Model Diagram of a Catalog Store

The model contains numeric parameters for the mean interarrival times for the phone, desk, and binning requests and their mean service times. The phone and desk requests have the same mean service times. Numeric identifiers are defined to make the model easier to read. The symbolic names are used in place of the numbers. Two global variables are defined. One is used as a counter for the number of binning requests that are waiting, and the other is the simulation clock used for timing purposes. There is a passive resource which assigns a higher priority to the phone and desk requests over the binning requests until eight hours have passed. There is an active service center with four servers. Binning operations take a different amount of time to process than the phone and desk requests. The infinite server queue delays the arrival of the desk requests by two hours. The desk is only open after two hours have elapsed. The first three set nodes are used to identify the types of requests. The final two set nodes keep track of how many binning requests are waiting to be processed. The number waiting have their priority increased after eight hours of the store operation. Any requests that arrive after nine hours of store operation are turned away. The type of request is checked to determine the amount of processing required.

```
MODEL:EX8.6
   METHOD:simulation
   /* Time unit is in hours. */
   NUMERIC PARAMETERS: atp atd atb stpd stb
   NUMERIC IDENTIFIERS:request phone desk binning
      REQUEST:0
      PHONE:1
      DESK:2
      BINNING:3
   GLOBAL VARIABLES:nbw clock
      NBW:0
      CLOCK:0
   QUEUE:pqassoc
      TYPE:passive
      TOKENS:4
      DSPL:prty
      ALLOCATE NODE LIST:alapd alab
         NUMBERS OF TOKENS TO ALLOCATE:1 1
         PRIORITIES:1 2
      RELEASE NODE LIST:rea
      DESTROY NODE LIST:deab
      CREATE NODE LIST:crabnbw
         NUMBERS OF TOKENS TO CREATE:nbw
   QUEUE:aqassoc
      TYPE:active
      SERVERS:4
      DSPL:fcfs
      CLASS LIST:clspd clsb
         WORK DEMANDS:stpd stb
      SERVER -
         RATES:1
   QUEUE:delay2
      TYPE:is
      CLASS LIST:cldly
         SERVICE TIMES:standard(2,0)
   SET NODES:setp
      ASSIGNMENT LIST:jv(request)=phone
   SET NODES:setd
      ASSIGNMENT LIST:jv(request)=desk
   SET NODES:setb
      ASSIGNMENT LIST:jv(request)=binning
   SET NODES:addb subb
      ASSIGNMENT LIST:nbw=nbw+1 nbw=nbw-1
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:srcp srcd srcb srcc
      ARRIVAL TIMES:atp atd atb standard(8,0)
      :srcp->setp sink;if(clock<=9) if(clock>9)
      :setp->alapd->clspd clsb; ++
            if(jv(request)=phone or jv(request)=desk) ++
            if(jv(request)=binning)
      :clspd->rea->sink
      :srcd->setd sink;if(clock<=7) if(clock>7)
      :setd->cldly->alapd
      :srcb->setb sink;if(clock<=9) if(clock>9)
      :setb->alapd addb;if(clock>=8) if(clock<8)
      :addb->alab->deab subb;if(clock>=8) if(clock<8)
      :deab->alapd
      :subb->clsb->rea
      :srcc->crabnbw->sink
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   RUN LIMITS -
      SIMULATED TIME:10
   LIMIT - CP SECONDS:10
   TRACE:no
END
```

The simulation is run for ten hours of store operation time to make sure any requests present after nine hours can be finished. The utilization of the clerks is very low. The idle time would include time for breaks, lunch, and possibly other activities. However, the four clerks could probably handle more requests. The model parameters used would represent a light day. At the end of the simulation we display the number of requests remaining (LNG). It is zero, which indicates that all of the requests were completed.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 10:17:02  DATE: 03/10/84
MODEL:EX8.6
ATP:.25
ATD:.2
ATB:.5
STPD:.1
STB:.3
RUN END: SIMULATED TIME LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                    SIMULATED TIME:      10.02981
                          CPU TIME:          0.18
                  NUMBER OF EVENTS:           261
```

```
WHAT:ALL

ELEMENT           UTILIZATION
PQASSOC           0.34423
 ALAPD             0.23751
 ALAB              0.10672
AQASSOC           0.34423
 SERVER    1       0.49106
 SERVER    2       0.42434
 SERVER    3       0.24138
 SERVER    4       0.22013
 CLSPD             0.17888
 CLSB              0.16534
DELAY2            0.00000

ELEMENT           THROUGHPUT
PQASSOC           9.57147
 ALAPD             7.97623
 ALAB              1.59524
AQASSOC           9.57147
 CLSPD             7.67712
 CLSB              1.89435
DELAY2            4.38692
REA               9.57147
CRABNBW           0.09970
SETP              3.29019
SETD              4.38692
SETB              1.89435
ADDB              1.59524
SUBB              1.59524
SRCP              3.88841
SRCD              5.98217
SRCB              2.09376
SRCC              0.09970
SINK              12.06404

ELEMENT           MEAN QUEUE LENGTH
PQASSOC           1.38402
 ALAPD             0.95715
 ALAB              0.42686
AQASSOC           1.37691
 CLSPD             0.71553
 CLSB              0.66138
DELAY2            8.77385

ELEMENT           STANDARD DEVIATION OF QUEUE LENGTH
PQASSOC           1.27967
 ALAPD             1.04610
 ALAB              0.59364
AQASSOC           1.26223
 CLSPD             0.90236
 CLSB              0.87530
DELAY2            5.30245
```

```
ELEMENT          MEAN QUEUEING TIME
PQASSOC          0.14460
 ALAPD            0.12000
 ALAB             0.26758
AQASSOC          0.14386
 CLSPD            0.09320
 CLSB             0.34913
DELAY2           2.00000

ELEMENT          STANDARD DEVIATION OF QUEUEING TIME
PQASSOC          0.21596
 ALAPD            0.18058
 ALAB             0.31409
AQASSOC          0.21476
 CLSPD            0.08741
 CLSB             0.38670

ELEMENT          MEAN TOKENS IN USE
PQASSOC          1.37691

ELEMENT          MEAN TOTAL TOKENS IN POOL
PQASSOC          4.00000

ELEMENT          MAXIMUM QUEUE LENGTH
PQASSOC          5
 ALAPD            5
 ALAB             2
AQASSOC          4
 CLSPD            4
 CLSB             4
DELAY2           17

ELEMENT          MAXIMUM QUEUEING TIME
PQASSOC          1.37963
 ALAPD            1.32846
 ALAB             1.37963
AQASSOC          1.37963
 CLSPD            0.43161
 CLSB             1.37963
DELAY2           2.00000

ELEMENT          OPEN CHAIN POPULATION
CH1              10.15786

ELEMENT          OPEN CHAIN RESPONSE TIME
CH1              0.84200

WHAT:ND(*)

ELEMENT          NUMBER OF DEPARTURES
PQASSOC          96
 ALAPD            80
 ALAB              16
AQASSOC           96
 CLSPD             77
 CLSB              19
DELAY2            44
REA               96
CRABNBW           1
SETP              33
SETD              44
SETB              19
ADDB              16
SUBB              16
SRCP              39
SRCD              60
SRCB              21
SRCC              1
SINK              121

WHAT:LNG

ELEMENT           FINAL LENGTHS
PQASSOC           0
AQASSOC           0
DELAY2            0
```

8.6. SUPERMARKET

A simple model of a supermarket will contain a deli section with three servers and a checkout area with four registers. Shoppers arrive and go to a set node to determine how many items they are going to purchase. Some of the customers go to the deli section. All customers spend time shopping for the number of items they need. If they buy ten or less items, there is a special checkout counter for them. Otherwise a customer picks the shortest line available at the other registers.

The model contains a numeric parameter for the mean interarrival time of customers. There are numeric identifiers defined for mean service times at the deli section and the registers and for the number of deli servers. The deli section is modeled as a multiserver queue with three classes. The shopping time is spent at an infinite server resource. The checkout counters are single server queues. The number of items to purchase is determined by sampling from a uniform distribution from one to 30. Since the uniform distribution is a continuous distribution, the sample is converted to an integer using the ceiling function at set node SETNUM. The number of items is saved as a customer attribute. Some customers go to the deli section, and all customers then spend time shopping. The first register (REG1) is for ten items or less. The customer attribute (JV(0)) is checked to deter-

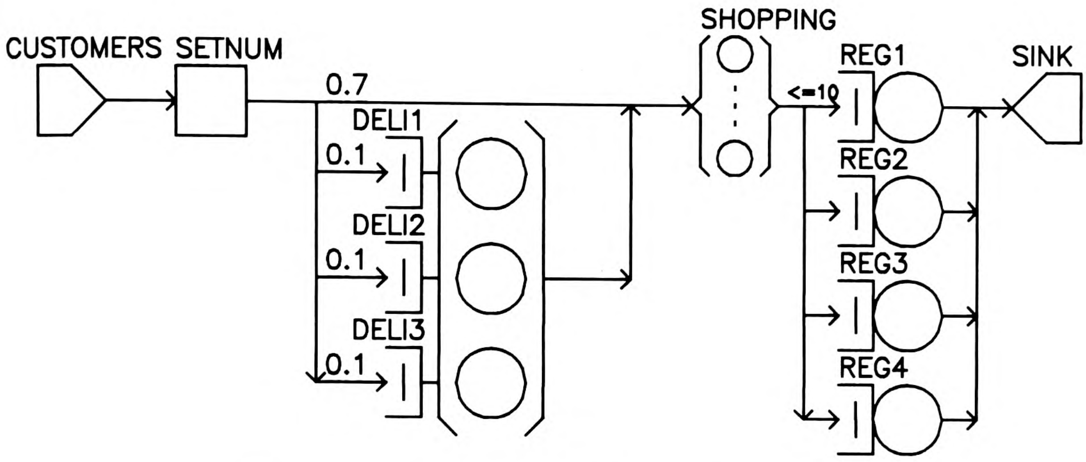


Figure 8.8. Model Diagram of a Supermarket

mine the number of items. The line lengths at the remaining checkouts are examined to determine which one is the shortest. The regenerative method is used to construct the confidence intervals. Since this is an open model, the regeneration state is the empty store.

```
MODEL:EX8.7
   METHOD:SIMULATION
   NUMERIC PARAMETERS:CUSTARRIVT
   NUMERIC IDENTIFIERS:DELI1TIME DELI2TIME DELI3TIME NUMDELI
      DELI1TIME:3
      DELI2TIME:5
      DELI3TIME:3
      NUMDELI:3
   NUMERIC IDENTIFIERS:REG1TIME REG2TIME REG3TIME REG4TIME
      REG1TIME:3
      REG2TIME:8
      REG3TIME:6
      REG4TIME:8
   QUEUE:DELIQ
      TYPE:ACTIVE
      SERVERS:NUMDELI
      DSPL:FCFS
      CLASS LIST:DELI1
         WORK DEMANDS:DELI1TIME
      CLASS LIST:DELI2
         WORK DEMANDS:DELI2TIME
      CLASS LIST:DELI3
         WORK DEMANDS:DELI3TIME
      SERVER -
         RATES:1
   QUEUE:SHOPPINGQ
      TYPE:IS
      CLASS LIST:SHOPPING
         SERVICE TIMES:JV(0)*0.5
   QUEUE:CKOUTQ1
      TYPE:FCFS
      CLASS LIST:REG1
         SERVICE TIMES:REG1TIME
   QUEUE:CKOUTQ2
      TYPE:FCFS
      CLASS LIST:REG2
         SERVICE TIMES:REG2TIME
   QUEUE:CKOUTQ3
      TYPE:FCFS
      CLASS LIST:REG3
         SERVICE TIMES:REG3TIME
   QUEUE:CKOUTQ4
      TYPE:FCFS
      CLASS LIST:REG4
         SERVICE TIMES:REG4TIME
   SET NODES:SETNUM
      ASSIGNMENT LIST:JV(0)=ceil(uniform(1,30,1))
   CHAIN:shoppingc
      TYPE:open
      SOURCE LIST:customers
      ARRIVAL TIMES:custarrivt
      :customers->setnum->deli1 deli2 deli3 shopping;.1 .1 .1 .7
      :deli1 deli2 deli3->shopping
      :shopping->reg1; if(jv(0)<=10)
      :shopping->reg2; if(ql(reg2)<=ql(reg3) and ql(reg2)<=ql(reg4))
      :shopping->reg3; if(ql(reg3)<=ql(reg4))
      :shopping->reg4
      :reg1 reg2 reg3 reg4->sink
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:ckoutq1 ckoutq2 ckoutq3 ckoutq4
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:30
   LIMIT - CP SECONDS:50
   TRACE:no
END
```

The sequential stopping procedure was used, and the simulation automatically stopped when the accuracy criteria were detected. The utilization of the deli servers is low, so there are probably too many deli servers. It takes an average of about 17 minutes for a customer to complete all of his or her shopping. The service times at the active service centers is displayed to see how close they are to the values specified in the model. They are very close, which is another indication of the accuracy of the results.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 08:39:19  DATE: 03/11/84
MODEL:EX8.7
CUSTARRIVT:3
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                     SIMULATED TIME:      6.1064E+04
                           CPU TIME:           33.62
                  NUMBER OF EVENTS:           66697
                  NUMBER OF CYCLES:              90

WHAT:ALLBO

ELEMENT            UTILIZATION
DELIQ              0.12655(0.12197,0.13113) 0.9%
 SERVER     1       0.27297(0.26546,0.28049) 1.5%
 SERVER     2       0.08845(0.08252,0.09438) 1.2%
 SERVER     3       0.01822(0.01526,0.02118) 0.6%
 DELI1              0.03486(0.03295,0.03677) 0.4%
 DELI2              0.05727(0.05353,0.06102) 0.7%
 DELI3              0.03442(0.03242,0.03641) 0.4%
SHOPPINGQ          0.00000(0.00000,0.00000)
CKOUTQ1            0.31559(0.30460,0.32657) 2.2%
CKOUTQ2            0.71813(0.70894,0.72732) 1.8%
CKOUTQ3            0.50173(0.48977,0.51368) 2.4%
CKOUTQ4            0.38382(0.36710,0.40055) 3.3%

ELEMENT            THROUGHPUT
DELIQ              0.10093(0.09886,0.10299) 4.1%
 DELI1              0.03342(0.03228,0.03456) 6.8%
 DELI2              0.03377(0.03234,0.03520) 8.5%
 DELI3              0.03374(0.03245,0.03502) 7.6%
SHOPPINGQ          0.33044(0.32641,0.33447) 2.4%
CKOUTQ1            0.10510(0.10264,0.10756) 4.7%
CKOUTQ2            0.09107(0.08941,0.09273) 3.7%
CKOUTQ3            0.08478(0.08302,0.08654) 4.1%
CKOUTQ4            0.04949(0.04783,0.05115) 6.7%
SETNUM             0.33044
CUSTOMERS          0.33044
SINK               0.33044

ELEMENT            MEAN QUEUE LENGTH
DELIQ              0.38070(0.36681,0.39458) 7.3%
 DELI1              0.10492(0.09918,0.11066) 10.9%
 DELI2          0.17225(0.16094,0.18356) 13.1%
 DELI3          0.10353(0.09757,0.10949) 11.5%
SHOPPINGQ      2.64157(2.58583,2.69731) 4.2%
CKOUTQ1        0.45832(0.43246,0.48418) 11.3%
CKOUTQ2        0.99057(0.96205,1.01910) 5.8%
CKOUTQ3        0.62758(0.60303,0.65214) 7.8%
CKOUTQ4        0.47015(0.44276,0.49753) 11.6%

ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
DELIQ          0.62001
 DELI1          0.32487
 DELI2          0.41783
 DELI3          0.32104
SHOPPINGQ      1.64753
CKOUTQ1        0.80065
CKOUTQ2        0.81149
CKOUTQ3        0.72752
CKOUTQ4        0.67279

ELEMENT        MEAN QUEUEING TIME
DELIQ          3.77201(3.67677,3.86724) 5.0%
 DELI1          3.13898(3.03003,3.24792) 6.9%
 DELI2          5.10101(4.91438,5.28763) 7.3%
 DELI3          3.06891(2.94338,3.19445) 8.2%
SHOPPINGQ      7.99406(7.85018,8.13793) 3.6%
CKOUTQ1        4.36063(4.18497,4.53628) 8.1%
CKOUTQ2        10.87720(10.47463,11.27977) 7.4%
CKOUTQ3        7.40245(7.17382,7.63107) 6.2%
CKOUTQ4        9.49997(9.09466,9.90528) 8.5%

ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
DELIQ          4.02950
 DELI1          3.10487
 DELI2          5.16425
 DELI3          3.11209
SHOPPINGQ      10.09827
CKOUTQ1        4.35481
CKOUTQ2        10.38003
CKOUTQ3        7.11697
CKOUTQ4        9.59193

ELEMENT        MAXIMUM QUEUE LENGTH
DELIQ          5
 DELI1          3
 DELI2          4
 DELI3          3
SHOPPINGQ      11
CKOUTQ1        6
CKOUTQ2        5
CKOUTQ3        5
CKOUTQ4        5
```

```
ELEMENT             MAXIMUM QUEUEING TIME
DELIQ               41.93370
 DELI1               35.06032
 DELI2               41.93370
 DELI3               30.24907
SHOPPINGQ           123.56955
CKOUTQ1             35.84251
CKOUTQ2             91.95149
CKOUTQ3             53.99690
CKOUTQ4             99.01994

ELEMENT             OPEN CHAIN POPULATION
SHOPPINGC           5.56889(5.45590,5.68187) 4.1%

ELEMENT             OPEN CHAIN RESPONSE TIME
SHOPPINGC           16.85286(16.62274,17.08296) 2.7%

WHAT:ST(*)

ELEMENT             MEAN SERVICE TIMES
DELIQ               3.76153
 DELI1               3.12891
 DELI2               5.08806
 DELI3               3.06050
SHOPPINGQ           7.99406
CKOUTQ1             3.00264
CKOUTQ2             7.88558
CKOUTQ3             5.91797
CKOUTQ4             7.75571

WHAT:ND(*)

ELEMENT             NUMBER OF DEPARTURES
DELIQ               6163
 DELI1               2041
 DELI2               2062
 DELI3               2060
SHOPPINGQ           20178
CKOUTQ1             6418
CKOUTQ2             5561
CKOUTQ3             5177
CKOUTQ4             3022
SETNUM              20178
CUSTOMERS           20178
SINK                20178
```

8.7. FURTHER READING

The models discussed in the first four sections were constructed to illustrate the use of some of the basic model elements. The catalog store model presented in Section 8.5 is based on discussions with Wessels [186]. The supermarket model is a simplified version of a model developed by Freireich [66]. Many books on simulation contain simple models like these which are encountered in everyday situations. The following books may be helpful in describing other models of similar systems: Gordon [71], Law and Kelton [106], Maisel and Gnugnoli [117], Pritsker and Pegden [135], Russell [146], and Schriber [164].

8.8. EXERCISES

8.1 Construct and solve some models of systems you might encounter in day-to-day activities.

8.2 What modeling element was used in model EX8.1 to limit the capacity of the barber shop? Why are the server utilizations decreasing from server 1 to server 5?

8.3 Explain how the passive queue is used to control the traffic light in model EX8.4.

8.4 Explain how the allocate, destroy, and create nodes were used in model EX8.5 to control the depletion and loading of sheets in the copier.

8.5 Explain how the shortest queue was selected in model EX8.7.

CHAPTER 9

COMPUTER SYSTEM MODELS

This chapter discusses three models of computer systems. The first is a simple version of a capacity planning model for an MVS type system. The second model is of a system with multiple processors and multiple memory units. The last one is a model of a mass storage subsystem.

9.1. CAPACITY PLANNING MODEL

Frequently, a very gross model of a computer system is all that is necessary for a capacity planning study. Capacity planning involves a baseline model which must be validated and some forecasts of future workloads and alternative configurations. Since it is very difficult to correctly project the future workload requirements accurately, a detailed model of the system is not necessary.

Figure 9.1 illustrates a model diagram of an MVS type of system. It depicts three different types of workloads. One is an interactive workload called TSO. The second one is a batch workload. The last one is a database workload called IMS. The TSO workload is represented as a closed chain with an infinite server for the terminals. There is a passive resource which restricts the multiprogramming level. The batch workload is a closed chain. The multiprogramming level is equal to the number of customers in the closed chain. Using this approach, we are assuming that when a batch job finishes, it is immediately replaced by another batch job. An open chain is used for the IMS workload. Transactions arrive at a certain rate of arrival, and there is also a maximum multiprogramming level for IMS transactions. Each workload contends for the CPU and the I/O devices. The CPU has a priority queueing discipline.

The first model of this system is a simple model that can be solved analytically. To do this, we will make some simplifying assumptions. The priority scheduling at the CPU is replaced by processor sharing. The model does not contain any memory constraints for the TSO or IMS workloads. The service times at each I/O device are the total service demanded for all visits to each device. Since this can be different for each workload, FCFS scheduling cannot be used. These simplifications can be avoided by using a queueing network package containing approximation techniques or simulation. The simulation approach will be used later in this section. This model contains only ten I/O devices. Most large systems contain many more I/O

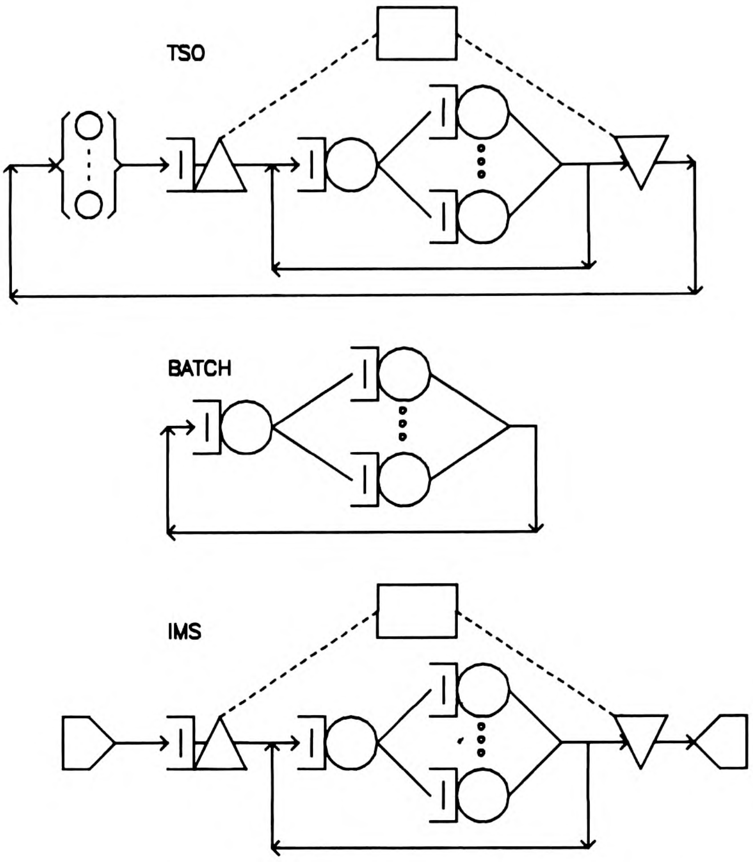


Figure 9.1. Model Diagram of an MVS Type System

devices. The I/O path contention and rotational position sensing are not being explicitly modeled. Some of this is captured in the measurement data. The model contains different parameters for the three different workloads. The parameters for the TSO workload include the think time (Z), the number of TSO terminals, the total service demand at the CPU for an average TSO transaction, and the total service demand at each of the I/O devices. The service demands at the I/O devices are defined as vectors with one element for each I/O device. The batch parameters are the number of batch jobs and the total of service demands at the CPU and I/O devices. The IMS workload parameters are the transaction arrival rate and the total service demands at all devices. In the real system, a transaction or job visits the CPU and an I/O device several times before completing. Because it is difficult to obtain measurement data for individual visits to I/O devices by workload, the total of service demands for all visits is used. This also simplifies the routing statements so that each device is branched to only once for each transaction or job.

```
MODEL:EX9.1
   METHOD:numerical
   NUMERIC PARAMETERS:   z ntso dcputso dtso(10)
   NUMERIC PARAMETERS:     nbat dcpubat dbat(10)
   NUMERIC PARAMETERS:lambdaims dcpuims dims(10)
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:z
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:      cputso    cpubat    cpuims
         SERVICE TIMES:dcputso   dcpubat   dcpuims
   QUEUE:disk1q
      TYPE:ps
      CLASS LIST:      disk1tso  disk1bat  disk1ims
         SERVICE TIMES:dtso(1)   dbat(1)   dims(1)
   QUEUE:disk2q
      TYPE:ps
      CLASS LIST:      disk2tso  disk2bat  disk2ims
         SERVICE TIMES:dtso(2)   dbat(2)   dims(2)
   QUEUE:disk3q
      TYPE:ps
      CLASS LIST:      disk3tso  disk3bat  disk3ims
         SERVICE TIMES:dtso(3)   dbat(3)   dims(3)
   QUEUE:disk4q
      TYPE:ps
      CLASS LIST:      disk4tso  disk4bat  disk4ims
         SERVICE TIMES:dtso(4)   dbat(4)   dims(4)
   QUEUE:disk5q
      TYPE:ps
      CLASS LIST:      disk5tso  disk5bat  disk5ims
         SERVICE TIMES:dtso(5)   dbat(5)   dims(5)
   QUEUE:disk6q
      TYPE:ps
      CLASS LIST:      disk6tso  disk6bat  disk6ims
         SERVICE TIMES:dtso(6)   dbat(6)   dims(6)
   QUEUE:disk7q
      TYPE:ps
      CLASS LIST:      disk7tso  disk7bat  disk7ims
         SERVICE TIMES:dtso(7)   dbat(7)   dims(7)
   QUEUE:disk8q
      TYPE:ps
      CLASS LIST:      disk8tso  disk8bat  disk8ims
         SERVICE TIMES:dtso(8)   dbat(8)   dims(8)
   QUEUE:disk9q
      TYPE:ps
      CLASS LIST:      disk9tso  disk9bat  disk9ims
         SERVICE TIMES:dtso(9)   dbat(9)   dims(9)
   QUEUE:disk10q
      TYPE:ps
      CLASS LIST:      disk10tso disk10bat disk10ims
         SERVICE TIMES:dtso(10)  dbat(10)  dims(10)
   CHAIN:chtso
      TYPE:closed
      POPULATION:ntso
      :terminals->cputso->++
       disk1tso->disk2tso->disk3tso->disk4tso->disk5tso->++
       disk6tso->disk7tso->disk8tso->disk9tso->disk10tso->++
       terminals
   CHAIN:chbat
      TYPE:closed
      POPULATION:nbat
      :cpubat->++
       disk1bat->disk2bat->disk3bat->disk4bat->disk5bat->++
       disk6bat->disk7bat->disk8bat->disk9bat->disk10bat->++
       cpubat
   CHAIN:chims
      TYPE:open
      SOURCE LIST:srcims
         ARRIVAL TIMES:1/lambdaims
      :srcims->cpuims->++
       disk1ims->disk2ims->disk3ims->disk4ims->disk5ims->++
       disk6ims->disk7ims->disk8ims->disk9ims->disk10ims->++
       sink
END
```

The parameter values given are for a system under a very heavy load. It could be a peak period on prime shift or it could be a benchmark with the processor driven very hard. All times are in seconds. All but one of the ten disks has a utilization greater than 15 percent. Most of the 16 TSO jobs are at the CPU. The CPU is certainly the bottleneck for the TSO workload. Remember that we are not using priority scheduling in the analytic solution. The batch workload is using the I/O devices quite a bit. The IMS transactions use DISK2 heavily, but not much else. Recall that the analytic solution does not place any memory constraint on the TSO or IMS workloads.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 19:14:31  DATE: 03/23/84
MODEL:EX9.1
Z:12.12
NTSO:16
DCPUTSO:6.37
DTSO:.039 .021 .25 .462 .501 .00005 .764 .12 .264 .072
NBAT:7
DCPUBAT:.0385
DBAT:.291 .343 .168 .147 .118 0 .34 .178 .177 .099
LAMBDAIMS:1.0
DCPUIMS:.00335
DIMS:.464 .034 .039 0 0 .097 0 0 .04 0
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL
```

```
ELEMENT          UTILIZATION
CPUQ             1.00000
 CPUTSO           0.93972
 CPUBAT           0.05693
 CPUIMS           3.3500E-03
DISK1Q           0.90003
 DISK1TSO         5.7534E-03
 DISK1BAT         0.43027
 DISK1IMS         0.46400
DISK2Q           0.54426
 DISK2TSO         3.0980E-03
 DISK2BAT         0.50716
 DISK2IMS         0.03400
DISK3Q           0.32429
 DISK3TSO         0.03688
 DISK3BAT         0.24841
 DISK3IMS         0.03900
DISK4Q           0.28551
 DISK4TSO         0.06816
 DISK4BAT         0.21736
DISK5Q           0.24838
 DISK5TSO         0.07391
 DISK5BAT         0.17448
DISK6Q           0.09701
 DISK6TSO         7.3762E-06
 DISK6IMS         0.09700
DISK7Q           0.61543
 DISK7TSO         0.11271
 DISK7BAT         0.50273
DISK8Q           0.28089
 DISK8TSO         0.01770
 DISK8BAT         0.26319
DISK9Q           0.34066
 DISK9TSO         0.03895
 DISK9BAT         0.26171
 DISK9IMS         0.04000
DISK10Q          0.15700
 DISK10TSO        0.01062
 DISK10BAT        0.14638

ELEMENT          THROUGHPUT
TERMINALSQ       0.14752
CPUQ             2.62613
 CPUTSO           0.14752
 CPUBAT           1.47861
 CPUIMS           1.00000

ELEMENT          MEAN QUEUE LENGTH
TERMINALSQ       1.78798
CPUQ             14.49063
 CPUTSO           13.55920
 CPUBAT           0.87954
 CPUIMS           0.05189
```

```
DISK1Q          5.17508
 DISK1TSO        0.03584
 DISK1BAT        2.27400
 DISK1IMS        2.86523
DISK2Q          1.03836
 DISK2TSO        6.3411E-03
 DISK2BAT        0.96271
 DISK2IMS        0.06930
DISK3Q          0.46646
 DISK3TSO        0.05417
 DISK3BAT        0.35510
 DISK3IMS        0.05719
DISK4Q          0.39101
 DISK4TSO        0.09492
 DISK4BAT        0.29609
DISK5Q          0.32576
 DISK5TSO        0.09808
 DISK5BAT        0.22768
DISK6Q          0.10743
 DISK6TSO        8.1686E-06
 DISK6IMS        0.10742
DISK7Q          1.36596
 DISK7TSO        0.26790
 DISK7BAT        1.09806
DISK8Q          0.37796
 DISK8TSO        0.02443
 DISK8BAT        0.35353
DISK9Q          0.50065
 DISK9TSO        0.05854
 DISK9BAT        0.38208
 DISK9IMS        0.06003
DISK10Q         0.18379
 DISK10TSO       0.01258
 DISK10BAT       0.17121


ELEMENT         MEAN QUEUEING TIME
TERMINALSQ      12.12001
CPUQ            5.51786
 CPUTSO          91.91231
 CPUBAT          0.59484
 CPUIMS          0.05189
DISK1Q          1.97061
 DISK1TSO        0.24293
 DISK1BAT        1.53793
 DISK1IMS        2.86523
DISK2Q          0.39540
 DISK2TSO        0.04298
 DISK2BAT        0.65109
 DISK2IMS        0.06930
DISK3Q          0.17762
 DISK3TSO        0.36718
 DISK3BAT        0.24016
```

```
  DISK3IMS          0.05719
DISK4Q            0.14889
  DISK4TSO          0.64345
  DISK4BAT          0.20025
DISK5Q            0.12405
  DISK5TSO          0.66484
  DISK5BAT          0.15398
DISK6Q            0.04091
  DISK6TSO          5.5371E-05
  DISK6IMS          0.10742
DISK7Q            0.52014
  DISK7TSO          1.81601
  DISK7BAT          0.74263
DISK8Q            0.14392
  DISK8TSO          0.16561
  DISK8BAT          0.23910
DISK9Q            0.19064
  DISK9TSO          0.39682
  DISK9BAT          0.25841
  DISK9IMS          0.06003
DISK10Q           0.06998
  DISK10TSO         0.08529
  DISK10BAT         0.11579

ELEMENT           OPEN CHAIN POPULATION
CHIMS             3.21107

ELEMENT           OPEN CHAIN RESPONSE TIME
CHIMS             3.21107
```

Now we will turn to a simulation model which explicitly includes memory constraints for the TSO and IMS workloads and priority scheduling at the CPU. Several new numeric parameters are included for these features. There are also two numeric parameters for controlling the simulation run length. The CPU is represented as a preemptive priority active queue. The ten disks are exactly the same as the previous model. There are two passive queues for the memory constraints for TSO and IMS. The routing is very similar to the last model. We are using independent replications to generate confidence intervals. Since the TSO and batch workloads are modeled as closed chains, the number of jobs in these chains must be initialized somewhere. We will run each replication until there are a specified number of departures from the CPUQ.

```
MODEL:EX9.2
   METHOD:simulation
   NUMERIC PARAMETERS:   z ntso mctso dcputso prtso dtso(10)
   NUMERIC PARAMETERS:          nbat  dcpubat prbat dbat(10)
   NUMERIC PARAMETERS:lambdaims mcims dcpuims prims dims(10)
   NUMERIC PARAMETERS:cpudep cpulim
   QUEUE:terminalsq
```

```
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:z
   QUEUE:cpuq
      TYPE:prtypr
      PREEMPT DIST:1
      CLASS LIST:       cputso    cpubat    cpuims
         SERVICE TIMES:dcputso    dcpubat   dcpuims
         PRIORITIES:    prtso     prbat     prims
   QUEUE:disk1q
      TYPE:fcfs
      CLASS LIST:       disk1tso  disk1bat  disk1ims
         SERVICE TIMES:dtso(1)    dbat(1)   dims(1)
   ...
   QUEUE:mctsoq
      TYPE:passive
      TOKENS:mctso
      DSPL:fcfs
      ALLOCATE NODE LIST:memtso
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:reltso
   QUEUE:mcimsq
      TYPE:passive
      TOKENS:mcims
      DSPL:fcfs
      ALLOCATE NODE LIST:memims
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:relims
   CHAIN:chtso
      TYPE:closed
      POPULATION:ntso
      :terminals->memtso->cputso->++
       disk1tso->disk2tso->disk3tso->disk4tso->disk5tso->++
       disk6tso->disk7tso->disk8tso->disk9tso->disk10tso->++
       reltso->terminals
   CHAIN:chbat
      TYPE:closed
      POPULATION:nbat
      :cpubat->++
       disk1bat->disk2bat->disk3bat->disk4bat->disk5bat->++
       disk6bat->disk7bat->disk8bat->disk9bat->disk10bat->++
       cpubat
   CHAIN:chims
      TYPE:open
      SOURCE LIST:srcims
         ARRIVAL TIMES:1/lambdaims
      :srcims->memims->cpuims->++
       disk1ims->disk2ims->disk3ims->disk4ims->disk5ims->++
       disk6ims->disk7ims->disk8ims->disk9ims->disk10ims->++
       relims->sink
   CONFIDENCE INTERVAL METHOD:replications
   INITIAL STATE DEFINITION -
   CHAIN:chtso
```

```
      NODE LIST:terminals
         INIT POP:ntso
   CHAIN:chbat
      NODE LIST:cpubat
         INIT POP:nbat
   CONFIDENCE LEVEL:90
   NUMBER OF REPLICATIONS:5
   REPLIC LIMITS -
      QUEUES FOR DEPARTURE COUNTS:cpuq
         DEPARTURES:cpudep
   LIMIT - CP SECONDS:cpulim
   TRACE:no
END
```

The parameter values are very similar to the ones used for the analytic solution. The memory constraint for TSO transactions is seven and for IMS it is three. IMS has the highest priority, followed by TSO and batch. Each replication is run until there are 15,000 departures from the CPUQ. Because of the priority scheduling, we can notice a difference in the utilizations at the CPU of the different workloads. There is very little batch work being completed because of the low priority. Notice the difference in mean queue lengths, partly because of the priority scheduling and partly because of the memory constraints. The TSO memory constraint is having a considerable effect on the TSO transactions. Most of the batch confidence intervals are very large because there were so few batch completions. The batch workload is probably not of much interest under these circumstances, so it is probably not worthwhile running the simulation longer. The IMS workload receives very good service since it has the highest priority.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 06:10:13  DATE: 03/24/84
MODEL:EX9.2
Z:12.12
NTSO:16
MCTSO:7
DCPUTSO:6.37
PRTSO:2
DTSO:.039 .021 .25 .462 .501 .00005 .764 .12 .264 .072
NBAT:7
DCPUBAT:.0385
PRBAT:3
DBAT:.291 .343 .168 .147 .118 0 .34 .178 .177 .099
LAMBDAIMS:1.0
MCIMS:3
DCPUIMS:.00335
PRIMS:1
DIMS:.464 .034 .039 0 0 .097 0 0 .04 0
CPUDEP:15000
CPULIM:900
REPLICATION    1: CPUQ DEPARTURE LIMIT
REPLICATION    2: CPUQ DEPARTURE LIMIT
REPLICATION   3: CPUQ DEPARTURE LIMIT
REPLICATION   4: CPUQ DEPARTURE LIMIT
REPLICATION   5: CPUQ DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.

  SIMULATED TIME PER REPLICATION:      1.2997E+04
                       CPU TIME:          246.55
NUMBER OF EVENTS PER REPLICATION:          179976
          NUMBER OF REPLICATIONS:               5

WHAT:ALLBO


ELEMENT             UTILIZATION
MCTSOQ              0.99969(0.99959,0.99980) 0.0%
MCIMSQ              0.45499(0.45003,0.45996) 1.0%
CPUQ                0.99995(0.99985,1.00004) 0.0%
 CPUTSO              0.99660(0.99650,0.99670) 0.0%
 CPUBAT              1.5670E-05(2.8273E-06,2.8512E-05) 0.0%
 CPUIMS              3.3335E-03(3.2980E-03,3.3691E-03) 0.0%
DISK1Q              0.46822(0.46451,0.47193) 0.7%
 DISK1TSO            6.2445E-03(6.1150E-03,6.3740E-03) 0.0%
 DISK1BAT            9.5024E-05(-7.3376E-06,1.9739E-04) 0.0%
 DISK1IMS            0.46188(0.45802,0.46574) 0.8%
DISK2Q              0.03708(0.03671,0.03745) 0.1%
 DISK2TSO            3.2965E-03(3.2002E-03,3.3927E-03) 0.0%
 DISK2BAT            1.1534E-04(3.0895E-05,1.9978E-04) 0.0%
 DISK2IMS            0.03367(0.03327,0.03406) 0.1%
DISK3Q              0.07800(0.07713,0.07886) 0.2%
 DISK3TSO            0.03909(0.03804,0.04014) 0.2%
 DISK3BAT            7.7618E-05(-2.8081E-06,1.5804E-04) 0.0%
 DISK3IMS            0.03882(0.03852,0.03913) 0.1%
DISK4Q              0.07285(0.07189,0.07381) 0.2%
 DISK4TSO            0.07277(0.07178,0.07376) 0.2%
 DISK4BAT            7.7736E-05(-8.9084E-06,1.6438E-04) 0.0%
DISK5Q              0.07851(0.07614,0.08087) 0.5%
 DISK5TSO            0.07848(0.07610,0.08085) 0.5%
 DISK5BAT            2.8736E-05(-9.1216E-06,6.6594E-05) 0.0%
DISK6Q              0.09626(0.09570,0.09683) 0.1%
 DISK6TSO            7.9240E-06(7.7194E-06,8.1287E-06) 0.0%
 DISK6IMS            0.09626(0.09569,0.09682) 0.1%
DISK7Q              0.12069(0.11700,0.12438) 0.7%
 DISK7TSO            0.12058(0.11691,0.12426) 0.7%
 DISK7BAT            1.0515E-04(-1.0682E-05,2.2097E-04) 0.0%
DISK8Q              0.01928(0.01891,0.01966) 0.1%
 DISK8TSO            0.01921(0.01885,0.01957) 0.1%
 DISK8BAT            7.5204E-05(-1.8727E-05,1.6913E-04) 0.0%
DISK9Q              0.08143(0.08013,0.08273) 0.3%
 DISK9TSO            0.04158(0.04031,0.04285) 0.3%
 DISK9BAT            9.4735E-05(-5.5351E-06,1.9501E-04) 0.0%
 DISK9IMS            0.03976(0.03941,0.04011) 0.1%
DISK10Q             0.01150(0.01114,0.01186) 0.1%
 DISK10TSO           0.01146(0.01107,0.01185) 0.1%
 DISK10BAT       4.2854E-05(4.1202E-06,8.1587E-05) 0.0%

ELEMENT         THROUGHPUT
MCTSOQ          0.15824(0.15547,0.16102) 3.5%
MCIMSQ          0.99523(0.98751,1.00295) 1.6%
TERMINALSQ      0.15929(0.15654,0.16204) 3.5%
CPUQ            1.15413(1.14923,1.15904) 0.9%
 CPUTSO          0.15827(0.15549,0.16105) 3.5%
 CPUBAT          4.3021E-04(4.9240E-05,8.1118E-04) 177.1%
 CPUIMS          0.99543(0.98776,1.00310) 1.5%
DISK1Q          1.15398(1.14902,1.15893) 0.9%
 DISK1TSO        0.15826(0.15548,0.16104) 3.5%
 DISK1BAT        4.3021E-04(4.9240E-05,8.1118E-04) 177.1%
 DISK1IMS        0.99529(0.98757,1.00301) 1.6%
SRCIMS          0.99550
SINK            0.99523

ELEMENT         MEAN QUEUE LENGTH
MCTSOQ          14.05746(13.97369,14.14124) 1.2%
MCIMSQ          1.75329(1.73402,1.77255) 2.2%
TERMINALSQ      1.94254(1.85876,2.02631) 8.6%
CPUQ            13.50794(13.49593,13.51996) 0.2%
 CPUTSO          6.50596(6.49417,6.51775) 0.4%
 CPUBAT          6.99855(6.99689,7.00021) 0.0%
 CPUIMS          3.4315E-03(3.4021E-03,3.4609E-03) 1.7%
DISK1Q          0.83577(0.82441,0.84714) 2.7%
 DISK1TSO        0.06371(0.06135,0.06608) 7.4%
 DISK1BAT        3.4609E-04(-9.4231E-05,7.8641E-04) 254.5%
 DISK1IMS        0.77171(0.76121,0.78222) 2.7%
DISK2Q          0.03972(0.03927,0.04017) 2.3%
 DISK2TSO        4.5709E-03(4.4653E-03,4.6766E-03) 4.6%
 DISK2BAT        2.4011E-04(6.7467E-05,4.1276E-04) 143.8%
 DISK2IMS        0.03491(0.03447,0.03535) 2.5%
DISK3Q          0.09141(0.09034,0.09248) 2.3%
 DISK3TSO        0.04345(0.04246,0.04444) 4.6%
 DISK3BAT        1.0766E-04(3.4833E-06,2.1184E-04) 193.5%
 DISK3IMS        0.04785(0.04740,0.04830) 1.9%
DISK4Q          0.11808(0.11548,0.12068) 4.4%
 DISK4TSO        0.08040(0.07891,0.08190) 3.7%
 DISK4BAT        9.8829E-05(-8.0906E-06,2.0575E-04) 216.4%
 DISK4IMS        0.03758(0.03615,0.03901) 7.6%
DISK5Q          0.15572(0.15030,0.16114) 7.0%
 DISK5TSO        0.08619(0.08318,0.08920) 7.0%
 DISK5BAT        2.9216E-05(-9.6176E-06,6.8049E-05) 265.8%
 DISK5IMS        0.06950(0.06678,0.07223) 7.8%
DISK6Q          0.11202(0.11117,0.11287) 1.5%
 DISK6TSO        3.5283E-04(2.8838E-04,4.1728E-04) 36.5%
 DISK6IMS        0.11167(0.11083,0.11250) 1.5%
DISK7Q          0.28534(0.27347,0.29721) 8.3%
 DISK7TSO        0.13796(0.13295,0.14298) 7.3%
 DISK7BAT        3.0532E-04(-2.1362E-04,8.2427E-04) 339.9%
 DISK7IMS        0.14707(0.14005,0.15409) 9.5%
DISK8Q          0.04385(0.04359,0.04412) 1.2%
```

```
  DISK8TSO          0.01966(0.01929,0.02002) 3.7%
  DISK8BAT          1.1025E-04(-4.0960E-05,2.6146E-04) 274.3%
  DISK8IMS          0.02409(0.02387,0.02430) 1.8%
 DISK9Q            0.15275(0.14802,0.15749) 6.2%
  DISK9TSO          0.04403(0.04250,0.04555) 6.9%
  DISK9BAT          1.6318E-04(-2.5070E-05,3.5142E-04) 230.7%
  DISK9IMS          0.10856(0.10512,0.11201) 6.3%
 DISK10Q           0.02022(0.01928,0.02117) 9.3%
  DISK10TSO         0.01157(0.01117,0.01198) 7.0%
  DISK10BAT         4.7168E-05(4.6567E-06,8.9679E-05) 180.3%
  DISK10IMS         8.6028E-03(8.0028E-03,9.2028E-03) 13.9%

 ELEMENT           MEAN QUEUEING TIME
 MCTSOQ            88.53023(86.49715,90.56329) 4.6%
 MCIMSQ            1.76138(1.73422,1.78853) 3.1%
 TERMINALSQ        12.17930(11.82077,12.53783) 5.9%
 CPUQ              5.63178(5.61408,5.64948) 0.6%
  CPUTSO            41.05673(40.27191,41.84154) 3.8%
  CPUBAT            0.08451(0.03258,0.13645) 122.9%
  CPUIMS            3.4473E-03(3.4356E-03,3.4589E-03) 0.7%
 DISK1Q            0.72419(0.71537,0.73301) 2.4%
  DISK1TSO          0.40260(0.38931,0.41590) 6.6%
  DISK1BAT          0.53513(0.16237,0.90789) 139.3%
  DISK1IMS          0.77530(0.76699,0.78360) 2.1%
 DISK2Q            0.03442(0.03410,0.03474) 1.9%
  DISK2TSO          0.02889(0.02825,0.02952) 4.4%
  DISK2BAT          0.56923(0.22259,0.91587) 121.8%
  DISK2IMS          0.03508(0.03477,0.03538) 1.7%
 DISK3Q            0.07922(0.07799,0.08045) 3.1%
  DISK3TSO          0.27455(0.27117,0.27793) 2.5%
  DISK3BAT          0.19876(0.04646,0.35107) 153.3%
  DISK3IMS          0.04808(0.04758,0.04859) 2.1%
 DISK4Q            0.10233(0.09996,0.10470) 4.6%
  DISK4TSO          0.50817(0.49808,0.51825) 4.0%
  DISK4BAT          0.20135(0.04256,0.36015) 157.7%
  DISK4IMS          0.03776(0.03628,0.03924) 7.8%
 DISK5Q            0.13499(0.13014,0.13983) 7.2%
  DISK5TSO          0.54462(0.52965,0.55959) 5.5%
  DISK5BAT          0.13265
  DISK5IMS          0.06984(0.06700,0.07269) 8.2%
 DISK6Q            0.09709(0.09643,0.09775) 1.4%
  DISK6TSO          2.2303E-03(1.8246E-03,2.6361E-03) 36.4%
  DISK6IMS          0.11220(0.11124,0.11316) 1.7%
 DISK7Q            0.24732(0.23605,0.25860) 9.1%
  DISK7TSO          0.87167(0.85091,0.89244) 4.8%
  DISK7BAT          0.41968(-0.00340,0.84277) 201.6%
  DISK7IMS          0.14783(0.13980,0.15585) 10.9%
 DISK8Q            0.03800(0.03769,0.03832) 1.6%
  DISK8TSO          0.12424(0.12165,0.12683) 4.2%
  DISK8BAT          0.16149(0.03651,0.28647) 154.8%
  DISK8IMS          0.02420(0.02395,0.02445) 2.0%
 DISK9Q            0.13239(0.12807,0.13671) 6.5%
  DISK9TSO          0.27820(0.27068,0.28572) 5.4%
 DISK9BAT        0.25993(0.06574,0.45412) 149.4%
 DISK9IMS        0.10909(0.10542,0.11277) 6.7%
DISK10Q         0.01753(0.01667,0.01839) 9.8%
 DISK10TSO       0.07313(0.07160,0.07466) 4.2%
 DISK10BAT       0.11474(0.06041,0.16908) 94.7%
 DISK10IMS       8.6465E-03(8.0115E-03,9.2814E-03) 14.7%

ELEMENT         MEAN TOKENS IN USE
MCTSOQ          6.99786(6.99714,6.99858) 0.0%
MCIMSQ          1.36498(1.35010,1.37986) 2.2%

ELEMENT         MEAN TOTAL TOKENS IN POOL
MCTSOQ          7.00000(7.00000,7.00000) 0.0%
MCIMSQ          3.00000

ELEMENT         MAXIMUM QUEUE LENGTH
MCTSOQ          16
MCIMSQ          18
TERMINALSQ      16
CPUQ            17
 CPUTSO          7
 CPUBAT          7
 CPUIMS          3
DISK1Q          8
 DISK1TSO        5
 DISK1BAT        6
 DISK1IMS        3
DISK2Q          6
 DISK2TSO        4
 DISK2BAT        4
 DISK2IMS        3
DISK3Q          7
 DISK3TSO        4
 DISK3BAT        3
 DISK3IMS        3
DISK4Q          8
 DISK4TSO        5
 DISK4BAT        2
 DISK4IMS        3
DISK5Q          7
 DISK5TSO        4
 DISK5BAT        1
 DISK5IMS        3
DISK6Q          5
 DISK6TSO        2
 DISK6BAT        1
 DISK6IMS        3
DISK7Q          9
 DISK7TSO        5
 DISK7BAT        7
 DISK7IMS        3
DISK8Q          7
 DISK8TSO        3
 DISK8BAT        4
 DISK8IMS        3
DISK9Q          8
 DISK9TSO        3
 DISK9BAT        4
 DISK9IMS        3
DISK10Q         5
 DISK10TSO       2
 DISK10BAT       2
 DISK10IMS       3

ELEMENT         MAXIMUM QUEUEING TIME
MCTSOQ          190.42751
MCIMSQ          14.78805
TERMINALSQ      119.49144
CPUQ            129.55780
 CPUTSO          129.55780
 CPUBAT          0.36434
 CPUIMS          0.03762
DISK1Q          6.23365
 DISK1TSO        5.50544
 DISK1BAT        1.52777
 DISK1IMS        6.23365
DISK2Q          1.27344
 DISK2TSO        0.61008
 DISK2BAT        1.27344
 DISK2IMS        0.94008
DISK3Q          3.67241
 DISK3TSO        3.67241
 DISK3BAT        0.75374
 DISK3IMS        1.78574
DISK4Q          4.45479
 DISK4TSO        4.45479
 DISK4BAT        0.69891
 DISK4IMS        4.44128
DISK5Q          6.01475
 DISK5TSO        6.01475
 DISK5BAT        0.47490
 DISK5IMS        4.60192
DISK6Q          1.15101
 DISK6TSO        0.63058
 DISK6IMS        1.15101
DISK7Q          7.35371
 DISK7TSO        7.35371
 DISK7BAT        2.44769
 DISK7IMS        6.83558
DISK8Q          1.17878
 DISK8TSO        1.17878
 DISK8BAT        1.06251
 DISK8IMS        1.06251
DISK9Q          2.66187
 DISK9TSO        2.61701
 DISK9BAT        1.17748
 DISK9IMS        2.66187
DISK10Q         0.77064
 DISK10TSO       0.77064
 DISK10BAT       0.42730
 DISK10IMS       0.76915

ELEMENT         OPEN CHAIN POPULATION
CHIMS           1.75329(1.73402,1.77255) 2.2%

ELEMENT         OPEN CHAIN RESPONSE TIME
CHIMS           1.76183(1.73457,1.78909) 3.1%
```

The parameters used for these solutions would probably indicate that this system is approaching its capacity limit during peak periods. They would have been derived from measurements and would represent a baseline case. Capacity planning involves predicting future system behavior under changing workloads and new configurations. The performance analyst must predict how the workloads will change and try various configurations which will handle the increased workloads. New parameter values must be determined to represent the growth in the workloads and the alternate configurations to handle the additional work. The model can be solved for many different parameter sets, and an adequate solution can be found.

9.2. SYSTEM MEMORY MODEL

The system consists of multiple processors attached by several buses to a number of shared memory units. A bus can provide a path from any processor to any memory unit. Depending on the number of processors, buses and memory units, there can be contention at the buses and the memory units. A job is first allocated a processor and accesses memory. If the required information is present in the memory unit, the job cycles back to the CPU. If the information is not present, the processor is released, and an access is made to an I/O device.

The multiple processor memory model is shown in Figure 9.2. It includes a passive queue representing the number of processors, a multiserver queue for the CPUs, a separate passive queue for each memory unit, and a passive queue for the buses. I/O devices are depicted by parallel single server queues. A job will wait at the processor passive queue when the number of jobs at the processors and the system memory units is equal to the number of processors. When a job is allocated a processor, it proceeds to the CPU queue. Since this is a multiserver queue with the number of servers equal to the number of processors, there is no contention at the CPUs. After an exponential service time, the job randomly chooses one of the memory units. If the memory unit is busy, the job waits in a queue.

When the memory unit is available, the job queues for a bus. After being allocated a bus, there is a constant service time to use the bus and transfer information from the memory unit. The job then either proceeds back to the CPU or releases the processor and performs I/O. The I/O service time is also exponential. This is a closed model with the number of jobs equal to the multiprogramming level in the computer system.

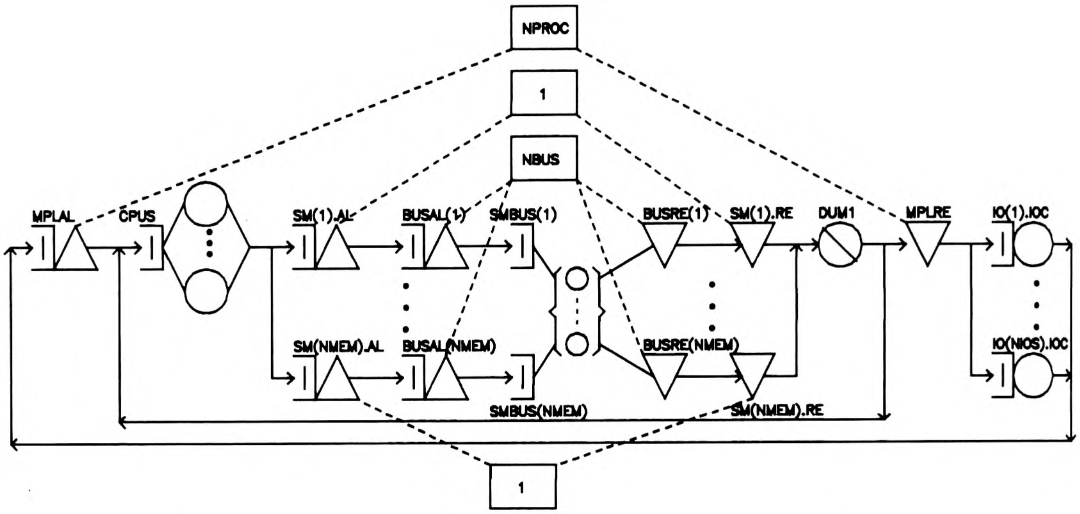


Figure 9.2. Model Diagram of System Memory Model

The model contains numeric parameters for the CPU service time for each visit, the bus time, the number of processors, the number of buses, the number of system memory units, the number of I/O devices, the I/O service time, the multiprogramming level, the number of cycles through the CPU and memory subsystem, the seed for the random number generator, and CPU run limit. Some of these parameters will determine the number of elements in the model. The values for the number of memory units (NMEM) and the number of I/O units (NIOS) will determine the number of queues and nodes in the model.

There is a submodel defined for one system memory unit and a submodel for one I/O device. These submodels are invoked a variable number of times based on the values of numeric parameters. The spectral method is used to construct the confidence intervals. All of the jobs in the closed chain are initialized at the passive queue to be allocated a processor.

```
MODEL:EX9.3
   METHOD:simulation
   NUMERIC PARAMETERS:cputime bustime nproc nbus nmem
   NUMERIC PARAMETERS:nios iost mpl ncycles replnum cpulim
   NODE ARRAYS:busal(nmem) smbus(nmem) busre(nmem)
   QUEUE:mplq
      TYPE:passive
      TOKENS:nproc
      DSPL:fcfs
      ALLOCATE NODE LIST:mplal
          NUMBER OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:mplre
   QUEUE:cpusq
      TYPE:active
      SERVERS:nproc
      DSPL:fcfs
      CLASS LIST:cpus
         WORK DEMANDS:cputime
      SERVER -
         RATES:1
   QUEUE:busq
      TYPE:passive
      TOKENS:nbus
      DSPL:fcfs
      ALLOCATE NODE LIST:busal(*)
          NUMBER OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:busre(*)
   QUEUE:smbusq
      TYPE:active
      SERVERS:nbus
      DSPL:fcfs
      CLASS LIST:smbus(*)
         WORK DEMANDS:constant(bustime)
      SERVER -
         RATES:1
   DUMMY NODES:dum1
   SUBMODEL:subsm
      CHAIN PARAMETERS:network
      QUEUE:smq
         TYPE:passive
         TOKENS:1
         DSPL:fcfs
         ALLOCATE NODE LIST: al
            NUMBER OF TOKENS TO ALLOCATE:1
         RELEASE NODE LIST: re
      CHAIN:network
         TYPE:external
         INPUT:al
         OUTPUT:re
   END OF SUBMODEL subsm
   SUBMODEL:subio
      CHAIN PARAMETERS:network
      QUEUE:ioq
```

```
            TYPE:fcfs
            CLASS LIST:ioc
               SERVICE TIMES:iost
         CHAIN:network
            TYPE:external
            INPUT:ioc
            OUTPUT:ioc
      END OF SUBMODEL subio
      INVOCATION:sm(nmem)
         TYPE:subsm
         NETWORK:network
      INVOCATION:io(nios)
         TYPE:subio
         NETWORK:network
      CHAIN:network
         TYPE:closed
         POPULATION:mpl
         :mplal->cpus
         :cpus->sm(*).al;1/nmem
         :sm(*).al->busal(*)->smbus(*)->busre(*)->sm(*).re
         :sm(*).re->dum1
         :dum1->mplre;1/ncycles
         :dum1->cpus;1-1/ncycles
         :mplre->io(*).ioc;1/nios
         :io(*).ioc->mplal
      CONFIDENCE INTERVAL METHOD:spectral
      INITIAL STATE DEFINITION -
      CHAIN:network
         NODE LIST:mplal
            INIT POP:mpl
      CONFIDENCE LEVEL:90
      SEQUENTIAL STOPPING RULE:yes
         CONFIDENCE INTERVAL QUEUES:mplq cpusq busq smbusq
            MEASURES:qt
            ALLOWED WIDTHS:10
         CONFIDENCE INTERVAL QUEUES:sm(1).smq io(1).ioq
            MEASURES:qt
            ALLOWED WIDTHS:10
         CONFIDENCE INTERVAL QUEUES:sm(2).smq io(2).ioq
            MEASURES:qt
            ALLOWED WIDTHS:10
         CONFIDENCE INTERVAL QUEUES:sm(3).smq io(3).ioq
            MEASURES:qt
            ALLOWED WIDTHS:10
         CONFIDENCE INTERVAL QUEUES:sm(4).smq io(4).ioq
            MEASURES:qt
            ALLOWED WIDTHS:10
      CONFIDENCE INTERVAL QUEUES:io(5).ioq io(6).ioq io(7).ioq io(8).ioq
         MEASURES:qt
         ALLOWED WIDTHS:10
      CONFIDENCE INTERVAL QUEUES:io(9).ioq io(10).ioq io(11).ioq
         MEASURES:qt
         ALLOWED WIDTHS:10
```

```
  CONFIDENCE INTERVAL QUEUES:io(12).ioq io(13).ioq io(14).ioq
      MEASURES:qt
      ALLOWED WIDTHS:10
  CONFIDENCE INTERVAL QUEUES:io(15).ioq io(16).ioq io(17).ioq
      MEASURES:qt
      ALLOWED WIDTHS:10
  CONFIDENCE INTERVAL QUEUES:io(18).ioq io(19).ioq io(20).ioq
      MEASURES:qt
      ALLOWED WIDTHS:10
   INITIAL PERIOD LIMITS -
      SIMULATED TIME:1000
      EVENTS:6000
   LIMIT - CP SECONDS:cpulim
   SEED:replnum
   TRACE:no
END
```

The model is solved with the following parameter values: 0.1 ms for the CPU service time, 0.02 ms for the bus time, four processors, one bus, four memory units, 20 I/O devices, 24 milliseconds for I/O service time, a multiprogramming level of 20, 100 cycles through the CPU and memory subsystem, the third seed for the random number generator, and 240 seconds for the model solution.

The model stopped after 240 CPU seconds. This occurred before the accuracy criteria were satisfied. The bus utilization is about 60 percent, and the I/O devices vary between 26 and 50 percent. The average queue length at the CPU and memory subsystem is 8.54. There are 3.86 jobs at the CPU and memory units and 4.68 jobs waiting. This leaves 11.46 jobs at the I/O devices. The throughput for the processor passive queue is 0.31 jobs per ms. The confidence intervals for the mean queueing times at the processor passive queue and the I/O devices are very large. The model should be run longer if these performance measures are critical.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 18:02:57  DATE: 03/30/84
MODEL:EX9.3
CPUTIME:.1
BUSTIME:.02
NPROC:4
NBUS:1
NMEM:4
NIOS:20
IOST:24
MPL:20
NCYCLES:100
REPLNUM:3
CPULIM:240
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
SAMPLING PERIOD END: EVENT LIMIT
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                  SIMULATED TIME:     5754.20313
                        CPU TIME:         240.22
                NUMBER OF EVENTS:         350524

WHAT:ALLBO

INVOCATION     ELEMENT        UTILIZATION
               MPLQ           0.96606
               BUSQ           0.60607
                BUSAL(1)       0.15143
                BUSAL(2)       0.15235
                BUSAL(3)       0.15122
                BUSAL(4)       0.15107
               CPUSQ          0.75510
                SERVER    1    0.86714
                SERVER    2    0.81193
                SERVER    3    0.73384
                SERVER    4    0.60749
               SMBUSQ         0.60607
                SMBUS(1)       0.15143
                SMBUS(2)       0.15235
                SMBUS(3)       0.15122
                SMBUS(4)       0.15107
SM(1)          SMQ            0.19393
SM(2)          SMQ            0.19550
SM(3)          SMQ            0.19376
SM(4)          SMQ            0.19378
IO(1)          IOQ            0.39011
IO(2)          IOQ            0.37937
IO(3)          IOQ            0.33730
IO(4)          IOQ            0.28396
IO(5)          IOQ            0.26692
IO(6)          IOQ            0.41879
IO(7)          IOQ            0.34019
IO(8)          IOQ            0.41775
IO(9)          IOQ            0.36413
IO(10)         IOQ            0.43473
IO(11)         IOQ            0.39990
IO(12)         IOQ            0.29864
IO(13)         IOQ            0.37315
IO(14)         IOQ            0.38230
IO(15)         IOQ            0.49748
IO(16)          IOQ             0.40157
IO(17)          IOQ             0.38410
IO(18)          IOQ             0.41368
IO(19)          IOQ             0.44312
IO(20)          IOQ             0.33350

INVOCATION      ELEMENT         THROUGHPUT
                MPLQ            0.31108
                BUSQ            30.30341
                 BUSAL(1)        7.57133
                 BUSAL(2)        7.61756
                 BUSAL(3)        7.56091
                 BUSAL(4)        7.55361
                CPUSQ           30.30357
                SMBUSQ          30.30341
                 SMBUS(1)        7.57133
                 SMBUS(2)        7.61756
                 SMBUS(3)        7.56091
                 SMBUS(4)        7.55361
SM(1)           SMQ             7.57133
SM(2)           SMQ             7.61756
SM(3)           SMQ             7.56091
SM(4)           SMQ             7.55361
IO(1)           IOQ             0.01616
IO(2)           IOQ             0.01442
IO(3)           IOQ             0.01442
IO(4)           IOQ             0.01217
IO(5)           IOQ             0.01390
IO(6)           IOQ             0.01720
IO(7)           IOQ             0.01269
IO(8)           IOQ             0.02085
IO(9)           IOQ             0.01616
IO(10)          IOQ             0.01599
IO(11)          IOQ             0.01477
IO(12)          IOQ             0.01251
IO(13)          IOQ             0.01442
IO(14)          IOQ             0.01303
IO(15)          IOQ             0.01894
IO(16)          IOQ             0.01564
IO(17)          IOQ             0.01825
IO(18)          IOQ             0.01634
IO(19)          IOQ             0.01634
IO(20)          IOQ             0.01495
                BUSRE(1)        7.57133
                BUSRE(2)        7.61756
                BUSRE(3)        7.56091
                BUSRE(4)        7.55361
                MPLRE           0.31108
                DUM1            30.30341
SM(1)           RE              7.57133
SM(2)           RE              7.61756
SM(3)           RE              7.56091
SM(4)           RE              7.55361
```

```
INVOCATION     ELEMENT        MEAN QUEUE LENGTH
               MPLQ           8.54315
               BUSQ           0.77696
                BUSAL(1)       0.19393
                BUSAL(2)       0.19550
                BUSAL(3)       0.19376
                BUSAL(4)       0.19378
               CPUSQ          3.02039
               SMBUSQ         0.60607
                SMBUS(1)       0.15143
                SMBUS(2)       0.15235
                SMBUS(3)       0.15122
                SMBUS(4)       0.15107
SM(1)          SMQ            0.21062
SM(2)          SMQ            0.21263
SM(3)          SMQ            0.21036
SM(4)          SMQ            0.21022
IO(1)          IOQ            0.60027
IO(2)          IOQ            0.51460
IO(3)          IOQ            0.53429
IO(4)          IOQ            0.37152
IO(5)          IOQ            0.34390
IO(6)          IOQ            0.73393
IO(7)          IOQ            0.47829
IO(8)          IOQ            0.68385
IO(9)          IOQ            0.53187
IO(10)         IOQ            0.66912
IO(11)         IOQ            0.60433
IO(12)         IOQ            0.38066
IO(13)         IOQ            0.55475
IO(14)         IOQ            0.58337
IO(15)         IOQ            0.95825
IO(16)         IOQ            0.67384
IO(17)         IOQ            0.52480
IO(18)         IOQ            0.53846
IO(19)         IOQ            0.70951
IO(20)         IOQ            0.46723

INVOCATION     ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
               MPLQ           3.62395
               BUSQ           0.73634
                BUSAL(1)       0.39537
                BUSAL(2)       0.39658
                BUSAL(3)       0.39524
                BUSAL(4)       0.39526
               CPUSQ          0.90662
               SMBUSQ         0.48862
                SMBUS(1)       0.35846
                SMBUS(2)       0.35936
                SMBUS(3)       0.35826
                SMBUS(4)       0.35812
SM(1)          SMQ            0.44806
SM(2)          SMQ            0.45056
SM(3)           SMQ            0.44764
SM(4)           SMQ            0.44734
IO(1)           IOQ            0.89312
IO(2)           IOQ            0.75876
IO(3)           IOQ            0.91099
IO(4)           IOQ            0.65971
IO(5)           IOQ            0.63399
IO(6)           IOQ            1.11966
IO(7)           IOQ            0.75321
IO(8)           IOQ            0.98036
IO(9)           IOQ            0.84805
IO(10)          IOQ            0.95530
IO(11)          IOQ            0.87561
IO(12)          IOQ            0.65095
IO(13)          IOQ            0.84111
IO(14)          IOQ            0.87061
IO(15)          IOQ            1.27903
IO(16)          IOQ            1.01358
IO(17)          IOQ            0.76722
IO(18)          IOQ            0.73233
IO(19)          IOQ            0.97948
IO(20)          IOQ            0.82061

INVOCATION      ELEMENT        MEAN QUEUEING TIME
                MPLQ           27.36353(23.45761,31.26944) 28.5%
                BUSQ           0.02564(0.02554,0.02574) 0.8%
                 BUSAL(1)       0.02561
                 BUSAL(2)       0.02566
                 BUSAL(3)       0.02563
                 BUSAL(4)       0.02565
                CPUSQ          0.09967(0.09917,0.10017) 1.0%
                SMBUSQ         0.02000(0.02000,0.02000) 0.0%
                 SMBUS(1)       0.02000
                 SMBUS(2)       0.02000
                 SMBUS(3)       0.02000
                 SMBUS(4)       0.02000
SM(1)           SMQ            0.02782(0.02767,0.02796) 1.0%
SM(2)           SMQ            0.02791(0.02776,0.02806) 1.1%
SM(3)           SMQ            0.02782(0.02765,0.02800) 1.2%
SM(4)           SMQ            0.02783(0.02766,0.02800) 1.2%
IO(1)           IOQ            37.14030(29.60065,44.67995) 40.6%
IO(2)           IOQ            35.67574(23.71260,47.63889) 67.1%
IO(3)           IOQ            37.04091(12.23162,61.85019) 134.0%
IO(4)           IOQ            30.53976(18.90219,42.17734) 76.2%
IO(5)           IOQ            24.47348(18.94743,29.99953) 45.2%
IO(6)           IOQ            42.65845(-4.26932,89.58623) 220.0%
IO(7)           IOQ            37.52390(23.05740,51.99037) 77.1%
IO(8)           IOQ            32.34151(19.63872,45.04428) 78.6%
IO(9)           IOQ            32.80647(16.21382,49.39912) 101.2%
IO(10)          IOQ            41.85065(26.15631,57.54500) 75.0%
IO(11)          IOQ            40.54945(31.45183,49.64708) 44.9%
IO(12)          IOQ            30.42253(18.63383,42.21123) 77.5%
IO(13)          IOQ            38.45975(30.29814,46.62137) 42.4%
IO(14)          IOQ             44.75768(11.54152,77.97385) 148.4%
IO(15)          IOQ             50.58678(27.33376,73.83981) 91.9%
IO(16)          IOQ             42.75700(18.30812,67.20587) 114.4%
IO(17)          IOQ             28.55008(22.88754,34.21262) 39.7%
IO(18)          IOQ             32.92131(25.28915,40.55345) 46.4%
IO(19)          IOQ             42.47963(14.61713,70.34215) 131.2%
IO(20)          IOQ             31.26234(-8.86049,71.38518) 256.7%

INVOCATION      ELEMENT         STANDARD DEVIATION OF QUEUEING TIME
                MPLQ            17.44107
                BUSQ            8.6671E-03
                 BUSAL(1)        8.6405E-03
                 BUSAL(2)        8.6829E-03
                 BUSAL(3)        8.6682E-03
                 BUSAL(4)        8.6767E-03
                CPUSQ           0.09938
SM(1)           SMQ             0.01107
SM(2)           SMQ             0.01108
SM(3)           SMQ             0.01106
SM(4)           SMQ             0.01106
IO(1)           IOQ             38.68065
IO(2)           IOQ             32.66183
IO(3)           IOQ             36.93301
IO(4)           IOQ             28.99449
IO(5)           IOQ             23.91405
IO(6)           IOQ             45.34605
IO(7)           IOQ             38.63686
IO(8)           IOQ             30.07637
IO(9)           IOQ             33.18271
IO(10)          IOQ             37.97098
IO(11)          IOQ             37.65411
IO(12)          IOQ             31.06099
IO(13)          IOQ             30.52859
IO(14)          IOQ             45.55791
IO(15)          IOQ             48.89714
IO(16)          IOQ             41.39999
IO(17)          IOQ             24.65466
IO(18)          IOQ             30.21254
IO(19)          IOQ             43.01801
IO(20)          IOQ             31.07422

INVOCATION      ELEMENT         MEAN TOKENS IN USE
                MPLQ            3.86423
                BUSQ            0.60607
SM(1)           SMQ             0.19393
SM(2)           SMQ             0.19550
SM(3)           SMQ             0.19376
SM(4)           SMQ             0.19378

INVOCATION      ELEMENT         MEAN TOTAL TOKENS IN POOL
                MPLQ            4.00000
                BUSQ            1.00000
SM(1)           SMQ             1.00000
SM(2)          SMQ            1.00000
SM(3)          SMQ            1.00000
SM(4)          SMQ            1.00000

INVOCATION     ELEMENT        MAXIMUM QUEUE LENGTH
               MPLQ           20
               BUSQ           4
                BUSAL(1)       1
                BUSAL(2)       1
                BUSAL(3)       1
                BUSAL(4)       1
               CPUSQ          4
               SMBUSQ         1
                SMBUS(1)       1
                SMBUS(2)       1
                SMBUS(3)       1
                SMBUS(4)       1
SM(1)          SMQ            4
SM(2)          SMQ            4
SM(3)          SMQ            4
SM(4)          SMQ            4
IO(1)          IOQ            5
IO(2)          IOQ            3
IO(3)          IOQ            5
IO(4)          IOQ            3
IO(5)          IOQ            3
IO(6)          IOQ            5
IO(7)          IOQ            3
IO(8)          IOQ            5
IO(9)          IOQ            5
IO(10)         IOQ            4
IO(11)         IOQ            4
IO(12)         IOQ            4
IO(13)         IOQ            5
IO(14)         IOQ            4
IO(15)         IOQ            6
IO(16)         IOQ            4
IO(17)         IOQ            4
IO(18)         IOQ            4
IO(19)         IOQ            5
IO(20)         IOQ            4

INVOCATION     ELEMENT        MAXIMUM QUEUEING TIME
               MPLQ           115.04950
               BUSQ           0.07899
                BUSAL(1)       0.07594
                BUSAL(2)       0.07792
                BUSAL(3)       0.07899
                BUSAL(4)       0.07859
               CPUSQ          1.06927
               SMBUSQ         0.02000
                SMBUS(1)       0.02000
                SMBUS(2)       0.02000
               SMBUS(3)        0.02000
               SMBUS(4)        0.02000
SM(1)          SMQ             0.10306
SM(2)          SMQ             0.11747
SM(3)          SMQ             0.11853
SM(4)          SMQ             0.11949
IO(1)          IOQ             206.14177
IO(2)          IOQ             139.16139
IO(3)          IOQ             127.89719
IO(4)          IOQ             112.80254
IO(5)          IOQ             129.50917
IO(6)          IOQ             231.51268
IO(7)          IOQ             218.30869
IO(8)          IOQ             123.88608
IO(9)          IOQ             153.15895
IO(10)         IOQ             167.86011
IO(11)         IOQ             173.59288
IO(12)         IOQ             160.68028
IO(13)         IOQ             133.22328
IO(14)         IOQ             244.72989
IO(15)         IOQ             224.45024
IO(16)         IOQ             174.84987
IO(17)         IOQ             137.76170
IO(18)         IOQ             155.20480
IO(19)         IOQ             211.46941
IO(20)         IOQ             158.90723
```

This model can be decomposed in several different fashions. One way is to put the processor passive queue and all resources within it in a submodel and solve it separately. The submodel solution can be accomplished by simulation and replaced by a flow equivalent server in the higher level model. This model with the flow equivalent server and the I/O devices can be solved analytically.

9.3. MASS STORAGE SUBSYSTEM

A mass storage subsystem (MSS) is mainly used as an archival storage device. It is sometimes used to replace a tape library. It consists of very slow, high-capacity storage areas. The storage medium is a roll of tape wrapped around a cylinder. Many of these cylinders are stored in a honeycomb area and must be retrieved when the data are requested.

The MSS model accepts four types of requests: (1) cylinder faults, (2) stage operations, (3) demount commands, and (4) mount requests. The different types of requests are identified by setting a job variable at a set node when a job first arrives. The different requests have different arrival rates, priorities at certain devices, and distinguishable service times at some resources. The interarrival times for the requests are assumed to follow an exponential distribution. We can specify the number of Data Recording Devices (DRDs), accessors, Data Recording Controllers (DRCs), and staging adapters. The cylinder fault requests have priority over the other requests for acquiring a DRD. After being allocated a DRD, a request must obtain an accessor. Then the request experiences a delay for moving the accessor. We are assuming there is no contention between the accessors. The accessor is released, and a service time representing a cartridge load is taken. The cartridge load operations can take place in parallel for each request which has moved the accessor and has a DRD. After loading the cartridge, the request must be allocated a DRC and a staging adapter. Then there is a delay representing the seek and data transfer operation. We are permitting each type of request to have a different type of distribution for seek and data transfer. Cylinder fault and mount requests have a constant distribution for this time. We are assuming there is no multiplexing of the DRCs. The DRC and staging adapter are released, and the model records the host response time. The DRD is rewound, with a constant service time, and an accessor is reacquired and moved to reposition the cartridge. Finally, the accessor is released, and the DRD is freed.

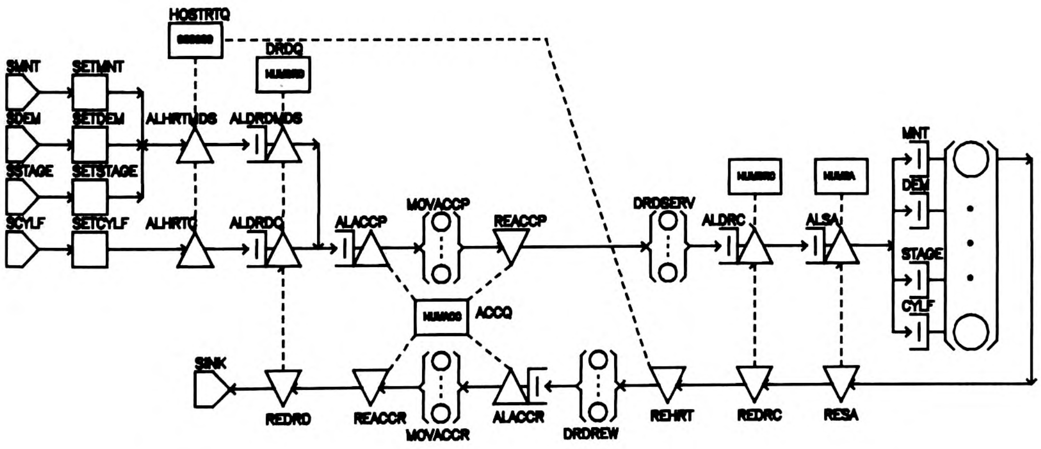


Figure 9.3. Model Diagram of a Mass Storage Subsystem

There are numeric parameters for the four different interarrival times, for the number of accessors, DRCs, staging adapters, and DRDs, for the service times for moving the accessor, the cartridge load time, the seek and data transfer for the four types of requests, the DRD rewind time, and the coefficient of variation of moving the accessor. There is a passive queue for recording the host response time. There are passive queues for the DRDs, the accessors, the DRCs, and the staging adapters. There are active queues for moving the accessors, loading the cartridge, doing the seek and data transfer, and rewinding the DRD. There are four sources, one for each type of request. The regenerative method is being used to calculate the confidence intervals along with the sequential stopping rule.

```
MODEL:EX9.4
   METHOD:simulation
   NUMERIC PARAMETERS:iatmnt iatdem iatstage iatcylf
   NUMERIC PARAMETERS:numacc numdrc numsa numdrd
   NUMERIC PARAMETERS:stmovacc stdrd
   NUMERIC PARAMETERS:stmnt stdem ststage stcylf stdrdrew
   NUMERIC PARAMETERS:cvmovacc
   NUMERIC IDENTIFIERS:mntni demni stageni cylfni
      MNTNI:1
      DEMNI:2
      STAGENI:3
      CYLFNI:4
   QUEUE:hostrtq
      TYPE:passive
      TOKENS:999999
      DSPL:fcfs
      ALLOCATE NODE LIST:alhrtmds alhrtc
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:rehrt
   QUEUE:drdq
      TYPE:passive
      TOKENS:numdrd
      DSPL:prty
      ALLOCATE NODE LIST:aldrdmds aldrdc
         NUMBERS OF TOKENS TO ALLOCATE:1
         PRIORITIES:2 1
      RELEASE NODE LIST:redrd
   QUEUE:accq
      TYPE:passive
      TOKENS:numacc
      DSPL:fcfs
      ALLOCATE NODE LIST:alaccp alaccr
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:reaccp reaccr
   QUEUE:movaccq
      TYPE:is
      CLASS LIST:movaccp movaccr
         SERVICE TIMES:standard(stmovacc,cvmovacc)
   QUEUE:drdservq
      TYPE:is
      CLASS LIST:drdserv
         SERVICE TIMES:constant(stdrd)
   QUEUE:drcq
      TYPE:passive
      TOKENS:numdrc
      DSPL:fcfs
      ALLOCATE NODE LIST:aldrc
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:redrc
   QUEUE:saq
      TYPE:passive
      TOKENS:numsa
      DSPL:fcfs
      ALLOCATE NODE LIST:alsa
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:resa
   QUEUE:seekdtq
      TYPE:is
      CLASS LIST:mnt dem stage cylf
         SERVICE TIMES:constant(stmnt) stdem ststage constant(stcylf)
   QUEUE:drdrewq
      TYPE:is
      CLASS LIST:drdrew
         SERVICE TIMES:stdrdrew
   SET NODES:setmnt setdem
      ASSIGNMENT LIST:jv(0)=mntni jv(0)=demni
   SET NODES:setstage setcylf
      ASSIGNMENT LIST:jv(0)=stageni jv(0)=cylfni
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:smnt sdem sstage scylf
      ARRIVAL TIMES:iatmnt iatdem iatstage iatcylf
      :smnt->setmnt->alhrtmds->aldrdmds
      :sdem->setdem->alhrtmds
      :sstage->setstage->alhrtmds
      :scylf->setcylf->alhrtc->aldrdc
      :aldrdmds aldrdc->alaccp->movaccp->reaccp->drdserv->aldrc->alsa
      :alsa->mnt dem stage cylf;if(jv(0)=mntni) if(jv(0)=demni) ++
                               if(jv(0)=stageni) if(jv(0)=cylfni)
      :mnt dem stage cylf->resa->redrc->rehrt
      :rehrt->drdrew->alaccr->movaccr->reaccr->redrd->sink
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:drdq
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:5
   LIMIT - CP SECONDS:60
   TRACE:no
END
```

The interarrival times for the four types of requests are 43.9, 35.2, 20, and 42.4 seconds between arrivals. There are two accessors, two DRCs, two staging adapters and four DRDs. It takes an average of six seconds to move an accessor and five seconds to do a cartridge load. The seek and data transfer times average 2.14, 10.7, 9.7, and 2.14 seconds for the four different requests. It takes an average of 4.2 seconds to rewind a DRD. The coefficient of variation of moving the accessor is 0.19.

After running for 60 seconds of CPU time, we look at the mean queuing time statistics and continue the run. The new CPU time limit is set at 150 seconds. After this new limit, most of the confidence limits are small except for some related to the host response time and the DRD passive queue. This MSS is fairly heavily loaded. It probably requires a larger number of DRDs and maybe more accessors. Increasing either of these may require more DRCs and staging adapters.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 18:42:45  DATE: 03/29/84
MODEL:EX9.4
IATMNT:43.9
IATDEM:35.2
IATSTAG:20
IATCYLF:42.4
NUMACC:2
NUMDRC:2
NUMSA:2
NUMDRD:4
STMOVACC:6
STDRD:5
STMNT:2.14
STDEM:10.7
STSTAGE:9.7
STCYLF:2.14
STDRDREW:4.2
CVMOVACC:0.19
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.     46 DISCARDED EVENTS

                  SIMULATED TIME:      1.0643E+05
                        CPU TIME:           60.15
                NUMBER OF EVENTS:           80010
                NUMBER OF CYCLES:              27

WHAT:QTBO
```

```
ELEMENT          MEAN QUEUEING TIME
HOSTRTQ          186.65231(161.47067,211.83395) 27.0%
DRDQ             197.89339(172.65480,223.13197) 25.5%
ACCQ             7.11519(7.08713,7.14324) 0.8%
DRCQ             7.89622(7.78656,8.00587) 2.8%
SAQ              7.22508(7.12230,7.32786) 2.8%
MOVACCQ          5.99362(5.98413,6.00311) 0.3%
DRDSERVQ         5.00000
SEEKDTQ          7.22508(7.12230,7.32786) 2.8%
DRDREWQ          4.21470(4.14267,4.28673) 3.4%

WHAT:
CONTINUE RUN:yes

LIMIT - CP SECONDS:150

SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
RUN END: CPU LIMIT
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
RUN END: CPU LIMIT CYCLE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                  SIMULATED TIME:     2.6808E+05
                        CPU TIME:         150.12
                NUMBER OF EVENTS:         199374
                NUMBER OF CYCLES:            110

WHAT:ALLBO

ELEMENT          UTILIZATION
HOSTRTQ          1.8919E-05(1.4241E-05,2.3597E-05) 0.0%
 ALHRTMDS         1.8433E-05(1.3718E-05,2.3147E-05) 0.0%
 ALHRTC           4.8615E-07(4.7411E-07,4.9820E-07) 0.0%
```

```
DRDQ             0.96518(0.95358,0.97678) 2.3%
 ALDRDMDS         0.81390(0.80247,0.82533) 2.3%
 ALDRDC           0.15128(0.14797,0.15459) 0.7%
ACCQ             0.74407(0.73675,0.75139) 1.5%
 ALACCP           0.37188(0.36804,0.37573) 0.8%
 ALACCR           0.37218(0.36868,0.37569) 0.7%
DRCQ             0.44072(0.43323,0.44821) 1.5%
SAQ              0.44072(0.43323,0.44821) 1.5%

ELEMENT          THROUGHPUT
HOSTRTQ          0.12395(0.12274,0.12517) 2.0%
 ALHRTMDS         0.10062(0.09939,0.10185) 2.5%
 ALHRTC           0.02333(0.02284,0.02383) 4.2%
DRDQ             0.12395(0.12274,0.12517) 2.0%
 ALDRDMDS         0.10062(0.09939,0.10185) 2.5%
 ALDRDC           0.02333(0.02284,0.02383) 4.2%
ACCQ             0.24791(0.24548,0.25034) 2.0%
 ALACCP           0.12395(0.12274,0.12517) 2.0%
 ALACCR           0.12395(0.12274,0.12517) 2.0%
DRCQ             0.12395(0.12274,0.12517) 2.0%
SAQ              0.12395(0.12274,0.12517) 2.0%
MOVACCQ          0.24791(0.24548,0.25034) 2.0%
 MOVACCP          0.12395(0.12274,0.12517) 2.0%
 MOVACCR          0.12395(0.12274,0.12517) 2.0%
DRDSERVQ         0.12395(0.12274,0.12517) 2.0%
SEEKDTQ          0.12395(0.12274,0.12517) 2.0%
 MNT              0.02274(0.02225,0.02323) 4.4%
 DEM              0.02790(0.02734,0.02845) 4.0%
 STAGE            0.04999(0.04899,0.05098) 4.0%
 CYLF             0.02333(0.02284,0.02383) 4.2%
DRDREWQ          0.12395(0.12274,0.12517) 2.0%
REHRT            0.12395
REDRD            0.12395
REACCP           0.12395
REACCR           0.12395
REDRC            0.12395
RESA             0.12395
SETMNT           0.02274
SETDEM           0.02790
SETSTAGE         0.04999
SETCYLF          0.02333
SMNT             0.02274
SDEM             0.02790
SSTAGE           0.04999
SCYLF            0.02333
SINK             0.12395

ELEMENT          MEAN QUEUE LENGTH
HOSTRTQ          18.91885(14.24051,23.59720) 49.5%
 ALHRTMDS         18.43269(13.71835,23.14706) 51.2%
 ALHRTC           0.48615(0.47410,0.49820) 5.0%
DRDQ             20.30960(15.62110,24.99811) 46.2%
 ALDRDMDS         19.55798(14.83429,24.28168) 48.3%
 ALDRDC          0.75162(0.73454,0.76869) 4.5%
ACCQ            1.76481(1.74394,1.78568) 2.4%
 ALACCP          0.89282(0.88163,0.90402) 2.5%
 ALACCR          0.87199(0.86210,0.88187) 2.3%
DRCQ            0.95738(0.93825,0.97650) 4.0%
SAQ             0.88143(0.86645,0.89642) 3.4%
MOVACCQ         1.48814(1.47350,1.50278) 2.0%
 MOVACCP         0.74377(0.73608,0.75146) 2.1%
 MOVACCR         0.74437(0.73735,0.75138) 1.9%
DRDSERVQ        0.61977(0.61369,0.62585) 2.0%
SEEKDTQ         0.88143(0.86645,0.89642) 3.4%
 MNT             0.04866(0.04760,0.04972) 4.4%
 DEM             0.30039(0.29291,0.30787) 5.0%
 STAGE           0.48245(0.47069,0.49421) 4.9%
 CYLF            0.04993(0.04888,0.05099) 4.2%
DRDREWQ         0.51876(0.51223,0.52529) 2.5%

ELEMENT         STANDARD DEVIATION OF QUEUE LENGTH
HOSTRTQ         15.51591
 ALHRTMDS        15.48125
 ALHRTC          0.71744
DRDQ            15.55947
 ALDRDMDS        15.51930
 ALDRDC          0.88582
ACCQ            1.01432
 ALACCP          0.82495
 ALACCR          0.78750
DRCQ            0.91519
SAQ             0.75812
MOVACCQ         0.68333
 MOVACCP         0.67579
 MOVACCR         0.67276
DRDSERVQ        0.65242
SEEKDTQ         0.75812
 MNT             0.21721
 DEM             0.51709
 STAGE           0.62557
 CYLF            0.22059
DRDREWQ         0.66858

ELEMENT         MEAN QUEUEING TIME
HOSTRTQ         152.62796(115.92844,189.32748) 48.1%
 ALHRTMDS        183.18936(138.09811,228.28059) 49.2%
 ALHRTC          20.83528(20.55783,21.11273) 2.7%
DRDQ            163.84782(127.14409,200.55157) 44.8%
 ALDRDMDS        194.37273(149.27301,239.47244) 46.4%
 ALDRDC          32.21257(31.90994,32.51520) 1.9%
ACCQ            7.11881(7.09781,7.13982) 0.6%
 ALACCP          7.20286(7.17328,7.23244) 0.8%
 ALACCR          7.03476(7.01626,7.05327) 0.5%
DRCQ            7.72365(7.60674,7.84055) 3.0%
SAQ             7.11097(7.02090,7.20104) 2.5%
MOVACCQ         6.00277(5.99661,6.00894) 0.2%
```

```
 MOVACCP        6.00035(5.99177,6.00894) 0.3%
 MOVACCR        6.00519(5.99616,6.01423) 0.3%
DRDSERVQ       5.00000
SEEKDTQ        7.11097(7.02090,7.20104) 2.5%
 MNT            2.14000
 DEM            10.76840(10.56662,10.97019) 3.7%
 STAGE          9.65173(9.48919,9.81427) 3.4%
 CYLF           2.14000(2.14000,2.14000) 0.0%
DRDREWQ        4.18511(4.14339,4.22683) 2.0%

ELEMENT        STANDARD DEVIATION OF QUEUEING TIME
HOSTRTQ        148.59668
 ALHRTMDS       149.09648
 ALHRTC         6.50429
DRDQ           148.64529
 ALDRDMDS       149.18004
 ALDRDC         7.91753
ACCQ           2.14633
 ALACCP         2.29846
 ALACCR         1.97904
DRCQ           9.14761
SAQ            8.78364
MOVACCQ        1.13785
 MOVACCP        1.13738
 MOVACCR        1.13833
SEEKDTQ        8.78364
 DEM            10.64073
 STAGE          9.56278
DRDREWQ        4.18850

ELEMENT        MEAN TOKENS IN USE
HOSTRTQ        18.91885(14.24051,23.59720) 49.5%
DRDQ           3.86072(3.81433,3.90711) 2.4%
ACCQ           1.48814(1.47350,1.50278) 2.0%
DRCQ           0.88143(0.86645,0.89642) 3.4%
SAQ            0.88143(0.86645,0.89642) 3.4%

ELEMENT        MEAN TOTAL TOKENS IN POOL
HOSTRTQ        1.0000E+06
DRDQ           4.00000(4.00000,4.00000) 0.0%
ACCQ           2.00000(2.00000,2.00000) 0.0%
DRCQ           2.00000(2.00000,2.00000) 0.0%
SAQ            2.00000(2.00000,2.00000) 0.0%

ELEMENT        MAXIMUM QUEUE LENGTH
HOSTRTQ        70
 ALHRTMDS       69
 ALHRTC         6
DRDQ           71
 ALDRDMDS       71
 ALDRDC         7
ACCQ           4
 ALACCP         4
```

```
  ALACCR              4
 DRCQ                4
 SAQ                 2
 MOVACCQ             2
  MOVACCP             2
  MOVACCR             2
 DRDSERVQ            4
 SEEKDTQ             2
  MNT                 2
  DEM                 2
  STAGE               2
  CYLF                2
 DRDREWQ             4

 ELEMENT             MAXIMUM QUEUEING TIME
 HOSTRTQ             692.52759
  ALHRTMDS            692.52759
  ALHRTC              60.36292
 DRDQ                703.02490
  ALDRDMDS            703.02490
  ALDRDC              70.21457
 ACCQ                15.91407
  ALACCP              15.91407
  ALACCR              15.20627
 DRCQ                96.60123
 SAQ                 96.60123
 MOVACCQ             7.97442
  MOVACCP             7.97442
  MOVACCR             7.97426
 DRDSERVQ            5.00000
 SEEKDTQ             96.60123
  MNT                 2.14000
  DEM                 96.60123
  STAGE               85.10968
  CYLF                2.14000
 DRDREWQ             43.82709

 ELEMENT             OPEN CHAIN POPULATION
 CH1                 20.30960(15.62109,24.99811) 46.2%

 ELEMENT             OPEN CHAIN RESPONSE TIME
 CH1                 163.84782(127.14407,200.55156) 44.8%
```

9.4. FURTHER READING

The capacity planning model presented in Section 9.1 resulted from conversations with Alex Birman and similar models described in the literature such as those in Lazowska, Zahorjan, Graham, and Sevcik [108]. The following papers are some references on capacity planning: Bronner [30, 31], Cooper [49], and Major [118]. The IBM manuals [80] contain some information about MVS. The system memory model in Section 9.2 is based on discussions with Al Blum, Lorenzo Donatiello, Ambuj Goyal, Phil Heidelberger, Steve Lavenberg, and Don Towsley. See Blum, Donatiello, Heidelberger, Lavenberg, and MacNair [22], for some further summary results from this model. The model of the mass storage subsystem is the result of conversations with Mike Coome [48].

There are several books which discuss computer system models: Allen [3], Ferrari [62], Kobayashi [98], Lavenberg, editor, [100], Lazowska, Zahorjan, Graham, and Sevcik [108], and Sauer and Chandy [152]. There are numerous papers dealing with computer system models. Some of them are Allen [4], Avi-Itzhak and Heyman [6], Boyse and Warn [26], Brandwajn [27, 28], Brown, Browne, and Chandy [32], Browne, Chandy, Brown, Keller, Towsley, and Dissley [33], Buzen [36, 39, 40], Chandy and Sauer [45], Chiu and Chow [46], Denning and Buzen [57], Kienzle and Sevcik [89], Lazowska [107], Lipsky and Church [111], Lo [113], Reiser [140], Rose [144], Sauer and Chandy [150, 151], Sauer and MacNair [153, chapter 7 of 156], Sauer, MacNair, and Kurose [158, 159, 160], Schwetman [168], and Wong and Graham [190].

9.5. EXERCISES

9.1 Construct and solve some models of computer systems you are familiar with.

9.2 Construct a model of a system that uses printer spooling. Assume there are two buffers which are filled and emptied by two independent tasks.

9.3 Construct a model which includes paging and swapping.

9.4 Construct a model with multiple paths to I/O devices and the simultaneous use of channels, control units, and head of strings.

9.5 Construct models of tightly coupled and loosely coupled multiprocessor systems.

9.6 Construct a model of a data-base system.

CHAPTER 10

COMMUNICATION NETWORKS

This chapter discusses three models of communication networks. The first model is a system with remote terminals controlled by a high speed processor and using high speed lines. The second model is a satellite system with three earth stations. The last model illustrates some common protocols found in communication networks, including polling, flow control, and packetizing.

10.1. REMOTE TERMINALS

The system we are modeling consists of 200 remote terminals controlled by high speed processors and lines connected to a host computer. There is one high speed processor for each group of 100 terminals. There are smaller, faster processors controlling the high speed lines. The host computer is running an interactive operating system mainly used for program development, experimentation, text processing, and graphics. The system also includes a facility for switching all 100 terminals attached to one of the high speed processors over to the other high speed processor in the event that a high speed line or line control processor goes down. Figure 10.1 illustrates a diagram of the system.

Figure 10.2 shows a model diagram of this system. This is an open model driven by a transaction arrival rate. We are modeling only one leg of the system shown in Figure 10.1. Therefore only the terminals communicating over one set of the high-speed lines will be represented in the model. The high speed processor is designated as a 4381. There is an active queue with two classes representing processing for inbound and outbound transactions through the 4381. There are two separate multiserver queues for inbound and outbound messages on the high speed lines and the smaller processors which are controlling the lines. There is an infinite server queue representing remote entry and host processing. The two classes at this infinite server queue represent different types of transactions. Q1 is for shorter transactions, and Q2 is for longer transactions.

This model is being solved analytically. There is a large parameter space which must be investigated, so the analytic solution is necessary for fast solution. There are numeric parameters for the arrival rate in transactions per hour per user, the number of users, the transmission block size, the number of lines, and the hour of the day. The hour is used to determine

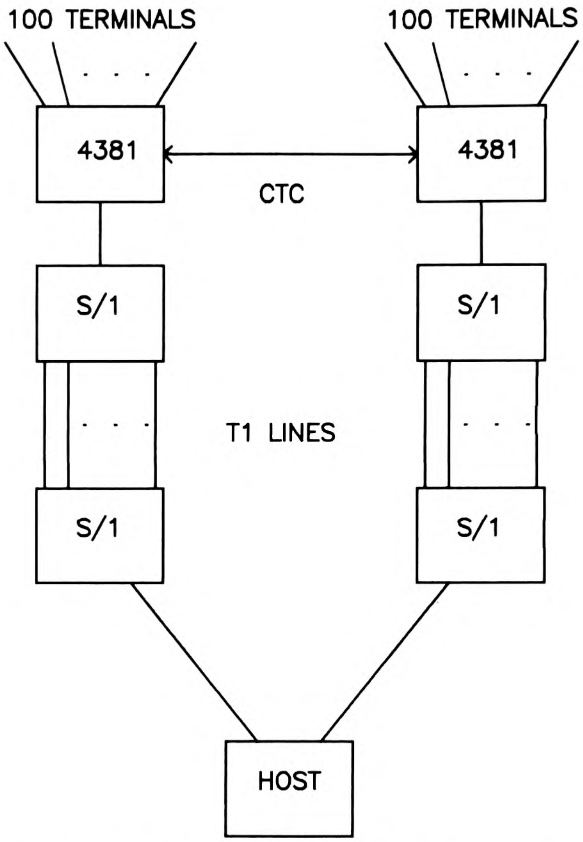


Figure 10.1. Diagram of Remote Terminal System

the amount of processing necessary on the host. It is different at different times of the day.

There are several numeric identifiers defined. These will be used as service times at the active queues. There is an active queue with two classes for the 4381. The two classes are used for inbound and outbound messages. There are two multiserver queues for the slower processors and the high-speed lines. Since the high-speed lines are full duplex, one multiserver queue is for inbound transactions and one for outbound transactions. There is an infinite server with two classes representing processing on the host. The two classes are for different types of transactions. There is a branching probability which determines the type of transaction. The model contains an open chain, and the arrival rate is used in an expression to calculate an interarrival time in seconds.

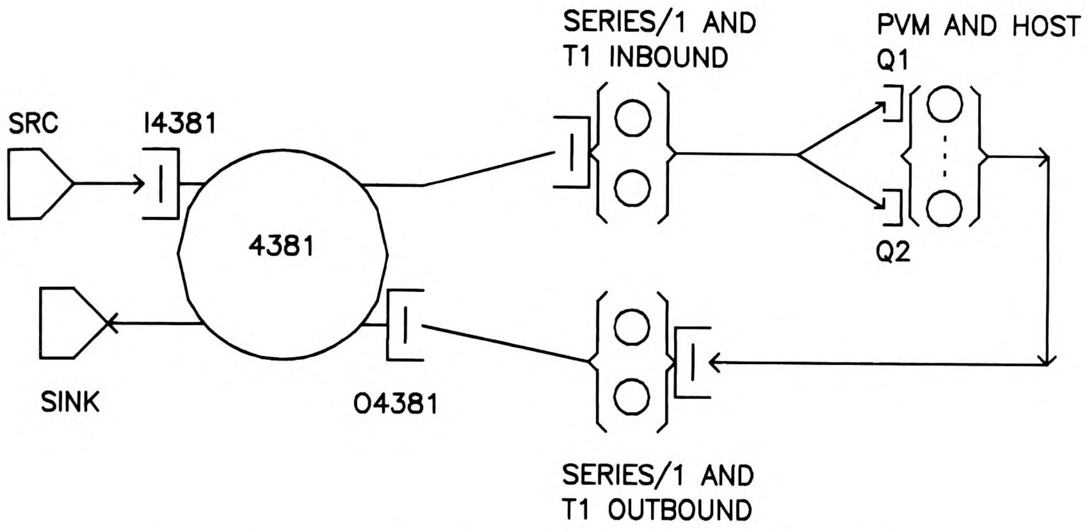


Figure 10.2. Model Diagram of Remote Terminal System

```
MODEL:EX10.1
   METHOD:numerical
   NUMERIC PARAMETERS:arrate users blksize nlines hour
      /* arrate is transactions per hour per user */
      /* hour - 9 to 16 (9am to 4pm) */
   NUMERIC IDENTIFIERS:stq1(8) stq2(8)
      STQ1:.372 .388 .338 .312 .349 .367 .408 .395
      STQ2:.721 .662 .652 .550 .586 .623 .715 .838
   NUMERIC IDENTIFIERS:st4381
      ST4381:.012       /* sec./trans. */
   NUMERIC IDENTIFIERS:stt1 sts1 sthost1 sthost2
      STT1:(8*blksize/1500)   /* 8bits*2kb/1.5mbps */
      STS1:.002          /* sec.        */
      STHOST1:(2*.012)+stq1(hour-8)   /* pvm in and out */
      STHOST2:(2*.012)+stq2(hour-8)   /* pvm in and out */
   QUEUE:q4381
      TYPE:fcfs
      CLASS LIST:i4381 o4381
         SERVICE TIMES:st4381
   QUEUE:t1iq   /* series/1->t1->series/1 */
      TYPE:active
      SERVERS:nlines
      DSPL:fcfs
      CLASS LIST:t1i
```

```
         WORK DEMANDS:stt1+(2*sts1)
      SERVER -
         RATES:1
   QUEUE:t1oq    /* series/1->t1->series/1 */
      TYPE:active
      SERVERS:nlines
      DSPL:fcfs
      CLASS LIST:t1o
         WORK DEMANDS:stt1+(2*sts1)
      SERVER -
         RATES:1
   QUEUE:hostq
      TYPE:is
      CLASS LIST:host1 host2
         SERVICE TIMES:sthost1 sthost2
   CHAIN:chn
      TYPE:open
      SOURCE LIST:src
         ARRIVAL TIMES:3600/(arrate*users)    /* sec. */
      :src->i4381->t1i->host1 host2;.592 .408->t1o->o4381->sink
END
```

The following interactive dialog will be used to solve the model multiple times and produce a file which will be used to plot the results. We can specify the number of plots and the number of curves on each plot. The eight curves will represent data for the eight different hours from nine in the morning until four in the afternoon. The numeric parameter ARRATE will be varied along the X-axis. The different arrival rates will be 503, 528, 553, 578, and 603. The Y-coordinate variable will be the HOUR. The expression evaluated to produce the data for the plots represents the relative degradation due to the remote delay as compared to local terminals. There are 100 terminals, a block size of two, and one high-speed line.

```
NUMBER OF PLOTS:1
NUMBER OF CURVES ON EACH PLOT:8
X-COORDINATE -
   X-VARIABLE NAME:ARRATE
      ARRATE VALUES:503 TO 603 BY 25
Y-COORDINATE -
   Y-VARIABLE NAME:HOUR
      HOUR VALUES:9 TO 16
   EXPRESSION FOR PLOT 1:(rtm(chn)-qt(hostq)+0.024)/rtm(chn)
OTHER MODEL PARAMETERS -
USERS:100
BLKSIZE:2
NLINES:1
```

Figure 10.3 depicts the graph of the relative remote delay for the different arrival rates and the eight hours during the day. At 4:00 pm the system exhibits the lowest relative degradation, and at 12 noon we find the largest relative degradation. These hours during the day correspond to the maximum and minimum expected host service times.

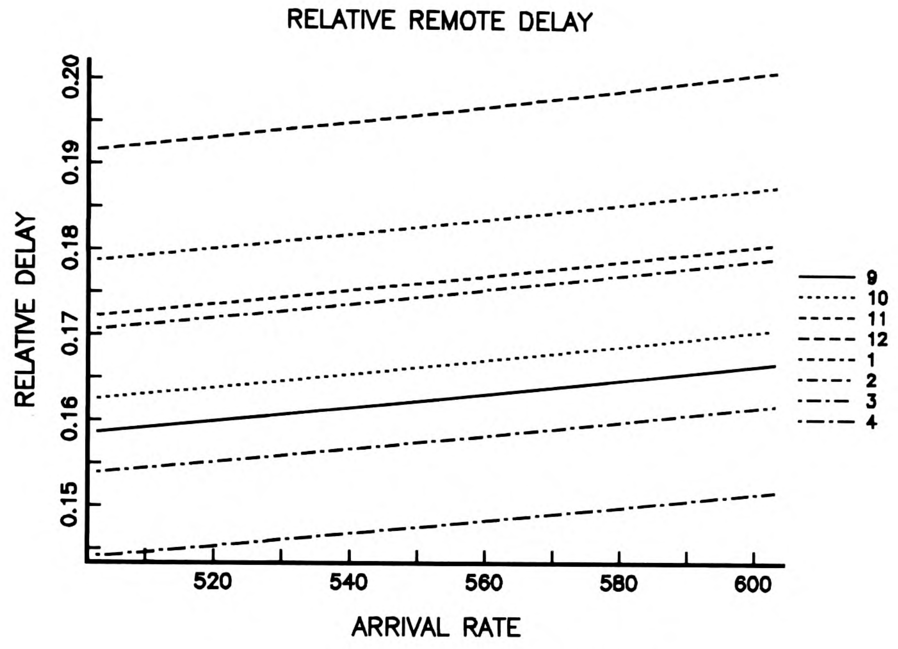


Figure 10.3. Plot of Model Results

In addition to these parameter values, this model was also solved with 200 users, a block size of four and six, and two and three lines. A three-dimensional plot with the block size varied in the third dimension is shown in Figure 10.4.

The table that follows shows the relative degradation from 90 solutions of the model. These results are from the peak period at 4:00 pm. The model was solved for 720 different model parameter combinations. To conserve space, the other results are not being shown. The results shown on the first three lines are the ones plotted in Figure 10.4. With 200 users, a block size of six, and one line, the system was saturated. Hence, the results are displayed for slightly smaller parameter values in this region.

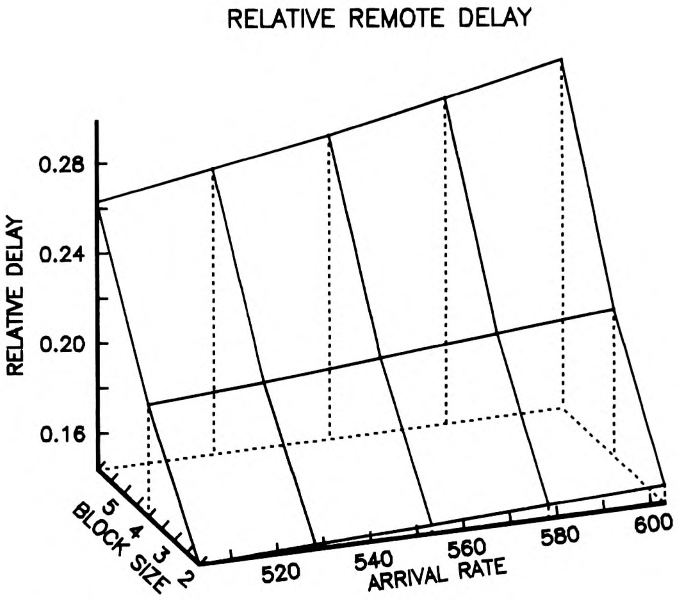


Figure 10.4. Three-dimensional Plot of Results

```
                            HOUR 16 (4:00 pm)
        ARRATE   503  528  553  578  603 USERS BLKSIZE NLINES
Relative        .144 .146 .148 .150 .152  100     2       1
Degradation     .194 .197 .201 .205 .209  100     4       1
                .263 .271 .279 .289 .299  100     6       1
                .203 .214 .227 .242 .262  200     2       1
                .320 .344 .374 .410 .455  200     4       1
                .422 .474 .545 .649 .823  190     5       1

                .135 .136 .137 .139 .140  100     2       2
                .163 .165 .166 .168 .169  100     4       2
                .192 .194 .196 .197 .199  100     6       2
                .181 .191 .203 .217 .236  200     2       2
                .212 .222 .233 .247 .266  200     4       2
                .251 .262 .276 .291 .311  200     6       2

                .134 .136 .137 .139 .140  100     2       3
                .161 .163 .164 .165 .166  100     4       3
                .187 .188 .189 .191 .192  100     6       3
                .180 .189 .201 .215 .234  200     2       3
                .205 .214 .225 .239 .256  200     4       3
                .230 .239 .250 .263 .280  200     6       3
```

10.2. SATELLITE MODEL

In this section, we discuss a model of a store-and-forward node scheduled satellite system. Because of the complexity of the model, we are using simulation to obtain the performance measures. The model includes many of the features that exist in the real system. There are three different earth stations which transmit messages to one another. Every earth station can transmit messages to itself or to any other earth station. Each message is identified by its source and destination pair. We used an exponential interarrival time distribution for messages entering the model from the earth stations. Each earth station can have a unique mean interarrival time for the messages it generates.

Messages entering the model may be successful transmissions, new transmissions which might collide, or retransmissions. A message must first be allocated a transmission buffer. Each source and destination pair can have a different buffer capacity depending on the transmission schedule selected. After obtaining a buffer, which corresponds to a slot in a time frame, the message waits until the beginning of the next frame. Each frame has a specified number of slots corresponding to the schedule selected. We have used a constant delay to represent the transmission of messages in each time slot. For each combination of source and destination, where the sources are different and the destinations are unique, the messages can be transmitted in parallel. The buffer is normally released as soon as the slot transmission is completed.

The three earth stations, labeled A, B, and C, generate messages at the three sources labeled STATIONA, STATIONB, and STATIONC. The probability of a collision is specified in the routing. Messages which collide leave the model. The arrival rate of messages from the earth stations includes messages which may have previously collided. Those messages which do not collide are probabilistically routed to determine the source and destination pair according to the schedule. The schedule used for this model permitted ten slots in each frame for transmission from station A to A, A to B, and A to C. Therefore, one third of the noncolliding A messages were routed to each destination. For stations B and C, the slot allocations were ten, five, and five for destinations A, B, and C, respectively. After tagging each message with its source/destination pair at the set nodes, messages were discarded if all the buffers in the frame were in use. When buffers are available, they are allocated to arriving messages. Messages which are allocated buffers wait until the beginning of the next frame. A passive queue and a timing mechanism are used to synchronize the beginning and ending of a frame transmission. A constant delay of 30 time units is used for the frame transmission. All waiting messages are scheduled into the next frame according to their source/destination pairs and the specified schedule.

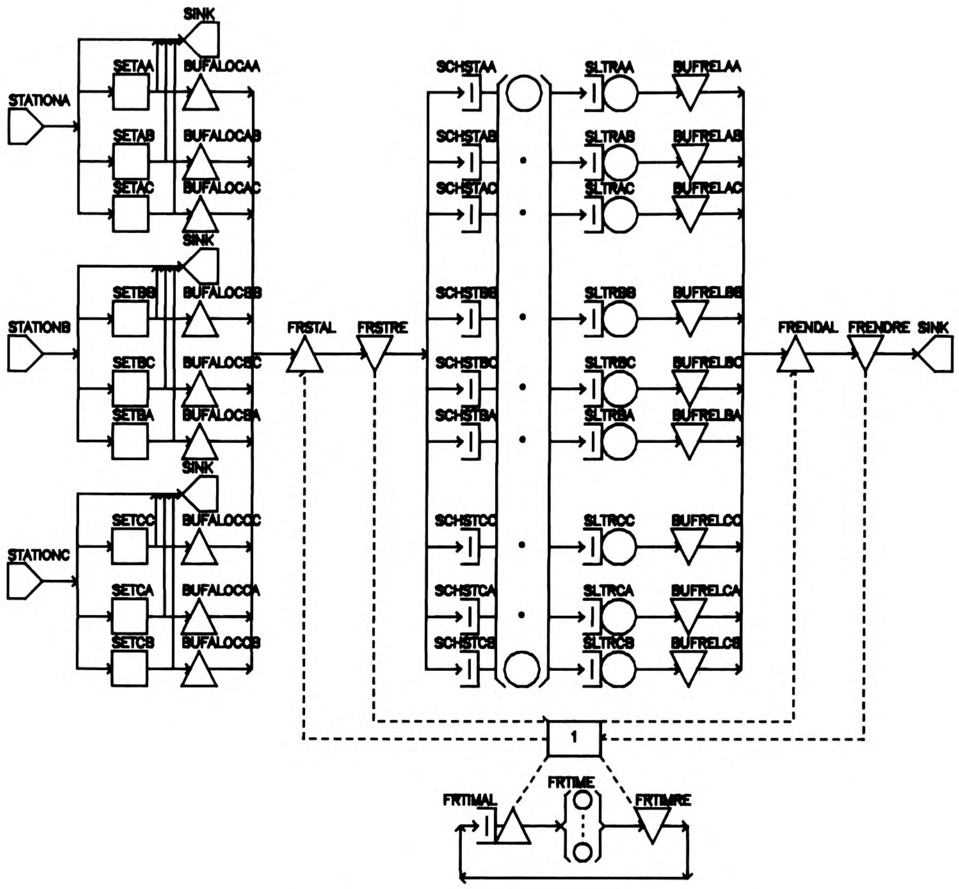


Figure 10.5. Model Diagram of Satellite System

Each message experiences a constant delay of one time unit for the slot transmission. After completing the slot transmission, the message releases the buffer it was allocated and leaves the model when the frame is ended.

```
MODEL:EX10.2
   METHOD:simulation
   NUMERIC PARAMETERS:iata iatb iatc
   NUMERIC IDENTIFIERS:aa ab ac
      AA:1
      AB:2
      AC:3
   NUMERIC IDENTIFIERS:bb bc ba
      BB:4
      BC:5
      BA:6
   NUMERIC IDENTIFIERS:cc ca cb
      CC:7
      CA:8
      CB:9
```

```
MAX JV:1
QUEUE:bufferqaa
   TYPE:passive
   TOKENS:10
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocaa
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelaa
QUEUE:bufferqab
   TYPE:passive
   TOKENS:10
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocab
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelab
QUEUE:bufferqac
   TYPE:passive
   TOKENS:10
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocac
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelac
QUEUE:bufferqbb
   TYPE:passive
   TOKENS:5
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocbb
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelbb
QUEUE:bufferqbc
   TYPE:passive
   TOKENS:5
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocbc
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelbc
QUEUE:bufferqba
   TYPE:passive
   TOKENS:10
   DSPL:fcfs
   ALLOCATE NODE LIST:bufalocba
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelba
QUEUE:bufferqcc
   TYPE:passive
   TOKENS:5
   DSPL:fcfs
   ALLOCATE NODE LIST:bufaloccc
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST:bufrelcc
QUEUE:bufferqca
   TYPE:passive
   TOKENS:10
```

```
      DSPL:fcfs
      ALLOCATE NODE LIST:bufalocca
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:bufrelca
   QUEUE:bufferqcb
      TYPE:passive
      TOKENS:5
      DSPL:fcfs
      ALLOCATE NODE LIST:bufaloccb
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:bufrelcb
   QUEUE:framepq
      TYPE:passive
      TOKENS:1
      DSPL:prty
      ALLOCATE NODE LIST:frstal frtimal frendal
         NUMBERS OF TOKENS TO ALLOCATE:1 1 1
         PRIORITIES:2 3 1
      RELEASE NODE LIST:frstre frtimre frendre
   QUEUE:schstq
      TYPE:is
      CLASS LIST:schstaa schstab schstac
         SERVICE TIMES:constant(10) constant(20) constant(0)
      CLASS LIST:schstbb schstbc schstba
         SERVICE TIMES:constant(0) constant(10) constant(20)
      CLASS LIST:schstcc schstca schstcb
         SERVICE TIMES:constant(20) constant(0) constant(10)
   QUEUE:frtimaq
      TYPE:is
      CLASS LIST:frtime
         SERVICE TIMES:constant(30)
   QUEUE:sltrqaa
      TYPE:fcfs
      CLASS LIST:sltraa
         SERVICE TIMES:constant(1)
   QUEUE:sltrqab
      TYPE:fcfs
      CLASS LIST:sltrab
         SERVICE TIMES:constant(1)
   QUEUE:sltrqac
      TYPE:fcfs
      CLASS LIST:sltrac
         SERVICE TIMES:constant(1)
   QUEUE:sltrqbb
      TYPE:fcfs
      CLASS LIST:sltrbb
         SERVICE TIMES:constant(1)
   QUEUE:sltrqbc
      TYPE:fcfs
      CLASS LIST:sltrbc
         SERVICE TIMES:constant(1)
   QUEUE:sltrqba
      TYPE:fcfs
      CLASS LIST:sltrba
         SERVICE TIMES:constant(1)
   QUEUE:sltrqcc
      TYPE:fcfs
      CLASS LIST:sltrcc
         SERVICE TIMES:constant(1)
   QUEUE:sltrqca
      TYPE:fcfs
      CLASS LIST:sltrca
         SERVICE TIMES:constant(1)
   QUEUE:sltrqcb
      TYPE:fcfs
      CLASS LIST:sltrcb
         SERVICE TIMES:constant(1)
   SET NODES:setaa setab setac
      ASSIGNMENT LIST:jv(0)=aa jv(0)=ab jv(0)=ac
   SET NODES:setbb setbc setba
      ASSIGNMENT LIST:jv(0)=bb jv(0)=bc jv(0)=ba
   SET NODES:setcc setca setcb
      ASSIGNMENT LIST:jv(0)=cc jv(0)=ca jv(0)=cb
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:stationa stationb stationc
      ARRIVAL TIMES:iata iatb iatc
      :stationa->sink setaa setab setac;.9 .0333333 .0333333 .0333334
      :stationb->sink setbb setbc setba;.9 .025 .025 .05
      :stationc->sink setcc setca setcb;.9 .025 .05 .025
      :setaa->bufalocaa sink;if(ta>0) if(t)
      :setab->bufalocab sink;if(ta>0) if(t)
      :setac->bufalocac sink;if(ta>0) if(t)
      :setbb->bufalocbb sink;if(ta>0) if(t)
      :setbc->bufalocbc sink;if(ta>0) if(t)
      :setba->bufalocba sink;if(ta>0) if(t)
      :setcc->bufaloccc sink;if(ta>0) if(t)
      :setca->bufalocca sink;if(ta>0) if(t)
      :setcb->bufaloccb sink;if(ta>0) if(t)
      :bufalocaa->frstal
      :bufalocab->frstal
      :bufalocac->frstal
      :bufalocbb->frstal
      :bufalocbc->frstal
      :bufalocba->frstal
      :bufaloccc->frstal
      :bufalocca->frstal
      :bufaloccb->frstal
      :frstal->frstre
      :frstre->schstaa;if(jv(0)=aa)
      :frstre->schstab;if(jv(0)=ab)
      :frstre->schstac;if(jv(0)=ac)
      :frstre->schstbb;if(jv(0)=bb)
      :frstre->schstbc;if(jv(0)=bc)
      :frstre->schstba;if(jv(0)=ba)
      :frstre->schstcc;if(jv(0)=cc)
      :frstre->schstca;if(jv(0)=ca)
      :frstre->schstcb;if(jv(0)=cb)
      :schstaa->sltraa->bufrelaa->frendal
      :schstab->sltrab->bufrelab->frendal
      :schstac->sltrac->bufrelac->frendal
      :schstbb->sltrbb->bufrelbb->frendal
      :schstbc->sltrbc->bufrelbc->frendal
      :schstba->sltrba->bufrelba->frendal
      :schstcc->sltrcc->bufrelcc->frendal
      :schstca->sltrca->bufrelca->frendal
      :schstcb->sltrcb->bufrelcb->frendal
      :frendal->frendre->sink
      :frtimal->frtime->frtimre->frtimal
   CONFIDENCE INTERVAL METHOD:none
   INITIAL STATE DEFINITION -
   CHAIN:ch1
   NODE LIST:frtimal
      INIT POP:1
   RUN LIMITS -
      SIMULATED TIME:3000
   LIMIT - CP SECONDS:30
   TRACE:no
END
```

We can model different schedules, representing symmetric and asymmetric types. We represented collisions both implicitly by specifying arrival rates of successful transmissions and by probabilistically turning away collided messages. Blocking was modeled by rejecting messages which did not collide if the buffer capacity for the source/destination pair was exceeded. The buffers can be held until the completion of the frame time, or they can be released when the slot transmission is complete. We ran the simulation for 3000 time units. The results shown include the number of departures at all nodes, the average and maximum delays, and the throughputs as a function of the load on the system for one set of interarrival times.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 10:11:07  DATE: 04/15/84
MODEL:EX10.2
IATA:.4
IATB:.4
IATC:.4
RUN END: SIMULATED TIME LIMIT
NO ERRORS DETECTED DURING SIMULATION.

                       SIMULATED TIME:     3000.00000
                             CPU TIME:           9.59
                     NUMBER OF EVENTS:          26732

WHAT:ND(*)
```

```
ELEMENT          NUMBER OF DEPARTURES
BUFFERQAA        236
BUFFERQAB        225
BUFFERQAC        249
BUFFERQBB        180
BUFFERQBC        169
BUFFERQBA        365
BUFFERQCC        172
BUFFERQCA        369
BUFFERQCB        187
FRAMEPQ          4375
 FRSTAL           2152
 FRTIMAL           100
 FRENDAL          2123
SCHSTQ           2152
 SCHSTAA           236
 SCHSTAB           225
 SCHSTAC           249
 SCHSTBB           180
 SCHSTBC           169
 SCHSTBA           365
 SCHSTCC           172
 SCHSTCA           369
 SCHSTCB           187
FRTIMAQ          100
SLTRQAA          236
SLTRQAB          225
SLTRQAC          249
SLTRQBB          180
SLTRQBC          169
SLTRQBA          365
SLTRQCC          172
SLTRQCA          369
SLTRQCB          187
BUFRELAA         236
BUFRELAB         225
BUFRELAC         249
BUFRELBB         180
BUFRELBC         169
BUFRELBA         365
BUFRELCC         172
BUFRELCA         369
BUFRELCB         187
FRSTRE           2152
FRTIMRE          100
FRENDRE          2123
SETAA            240
SETAB            226
SETAC            250
SETBB            183
SETBC            174
SETBA            370
SETCC            191
```

```
SETCA          374
SETCB          197
STATIONA       7463
STATIONB       7439
STATIONC       7426
SINK           22278

WHAT:QT(*)

ELEMENT        MEAN QUEUEING TIME
BUFFERQAA      27.11955
BUFFERQAB      37.71048
BUFFERQAC      17.40508
BUFFERQBB      17.65688
BUFFERQBC      25.89842
BUFFERQBA      37.93280
BUFFERQCC      37.27525
BUFFERQCA      18.43593
BUFFERQCB      25.97539
FRAMEPQ        16.79979
 FRSTAL         15.14503
 FRTIMAL        30.00000
 FRENDAL        17.85539
SCHSTQ         9.83271
 SCHSTAA        10.00000
 SCHSTAB        20.00000
 SCHSTAC        0.00000
 SCHSTBB        0.00000
 SCHSTBC        10.00000
 SCHSTBA        20.00000
 SCHSTCC        20.00000
 SCHSTCA        0.00000
 SCHSTCB        10.00000
FRTIMAQ        30.00000
SLTRQAA        2.33898
SLTRQAB        2.16444
SLTRQAC        2.26506
SLTRQBB        1.73889
SLTRQBC        1.76923
SLTRQBA        2.76712
SLTRQCC        1.83721
SLTRQCA        2.82114
SLTRQCB        1.90909

WHAT:MXQT(*)

ELEMENT        MAXIMUM QUEUEING TIME
BUFFERQAA      40.92558
BUFFERQAB      50.93857
BUFFERQAC      30.95584
BUFFERQBB      30.99225
BUFFERQBC      40.93324
BUFFERQBA      50.97237
```

```
BUFFERQCC        50.70691
BUFFERQCA        30.87671
BUFFERQCB        40.92767
FRAMEPQ          30.00000
 FRSTAL           29.99225
 FRTIMAL          30.00000
 FRENDAL          30.00000
SCHSTQ           20.00000
 SCHSTAA          10.00000
 SCHSTAB          20.00000
 SCHSTAC          0.00000
 SCHSTBB          0.00000
 SCHSTBC          10.00000
 SCHSTBA          20.00000
 SCHSTCC          20.00000
 SCHSTCA          0.00000
 SCHSTCB          10.00000
FRTIMAQ          30.00000
SLTRQAA          9.00000
SLTRQAB          7.00000
SLTRQAC          9.00000
SLTRQBB          5.00000
SLTRQBC          5.00000
SLTRQBA          10.00000
SLTRQCC          5.00000
SLTRQCA          10.00000
SLTRQCB          5.00000

WHAT:TP(*)

ELEMENT          THROUGHPUT
BUFFERQAA        0.07867
BUFFERQAB        0.07500
BUFFERQAC        0.08300
BUFFERQBB        0.06000
BUFFERQBC        0.05633
BUFFERQBA        0.12167
BUFFERQCC        0.05733
BUFFERQCA        0.12300
BUFFERQCB        0.06233
FRAMEPQ          1.45833
 FRSTAL           0.71733
 FRTIMAL          0.03333
 FRENDAL          0.70767
SCHSTQ           0.71733
 SCHSTAA          0.07867
 SCHSTAB          0.07500
 SCHSTAC          0.08300
 SCHSTBB          0.06000
 SCHSTBC          0.05633
 SCHSTBA          0.12167
 SCHSTCC          0.05733
 SCHSTCA          0.12300
 SCHSTCB          0.06233
FRTIMAQ          0.03333
SLTRQAA          0.07867
SLTRQAB          0.07500
SLTRQAC          0.08300
SLTRQBB          0.06000
SLTRQBC          0.05633
SLTRQBA          0.12167
SLTRQCC          0.05733
SLTRQCA          0.12300
SLTRQCB          0.06233
BUFRELAA         0.07867
BUFRELAB         0.07500
BUFRELAC         0.08300
BUFRELBB         0.06000
BUFRELBC         0.05633
BUFRELBA         0.12167
BUFRELCC         0.05733
BUFRELCA         0.12300
BUFRELCB         0.06233
FRSTRE           0.71733
FRTIMRE          0.03333
FRENDRE          0.70767
SETAA            0.08000
SETAB            0.07533
SETAC            0.08333
SETBB            0.06100
SETBC            0.05800
SETBA            0.12333
SETCC            0.06367
SETCA            0.12467
SETCB            0.06567
STATIONA         2.48767
STATIONB         2.47967
STATIONC         2.47533
SINK             7.42600
```

10.3. COMMUNICATION PROTOCOL MODEL

This section considers remote terminals connected to an interactive computing system. We assume the terminals are organized in three separate groups. The terminals share a full duplex 2400 baud line to the computer system. In order to avoid conflicts between traffic destined from a terminal group to the computer system, a polling protocol gives each group a turn to transmit any traffic it has for the computing system. The messages sent from the terminals to the computing system are fairly short, with a maximum length of 640 bits. However, the messages sent from the computing system to the terminals are longer and more variable in length, with a mean length of 800 bits. To prevent a long message from monopolizing the line from the computing system to the terminals, the messages are divided into packets of maximum length of 256 bits. Only 240 of the 256 bits are used for data, with the remaining bits used for control information. To prevent a terminal controller from receiving more data than it can handle, a simple window flow control protocol is used. The protocol allows only a single message (typically, several packets) to be sent to a terminal group before that group explicitly requests another message to be sent.

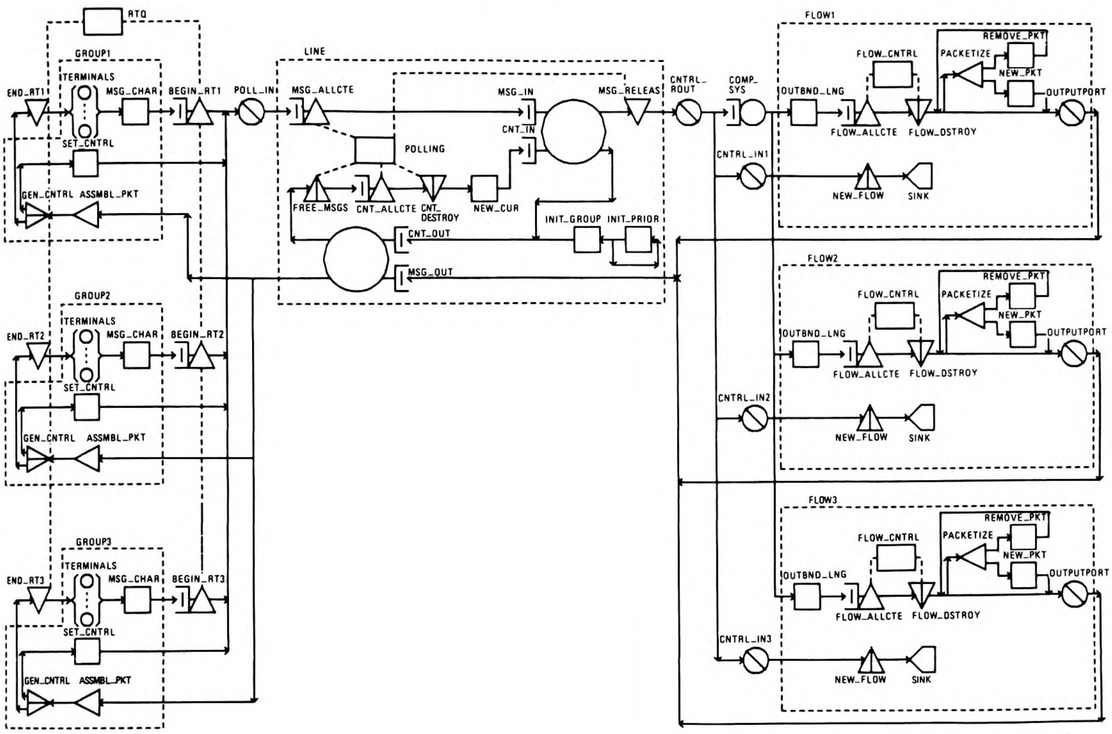


Figure 10.6. Model Diagram of Communication Protocol Model

The model consists of three submodels, a queue representing the computer system, and a passive queue used for measuring response times. The first submodel, TERM__GROUP, represents a terminal group. There will be one invocation of TERM__GROUP for each group. The second submodel, POLL__LINE, represents the communication line. There is just one invocation of POLL__LINE. The third submodel, FLOW__N__PKT, represents the window flow control protocol and the division of messages into packets. There will be one invocation of FLOW__N__PKT for each terminal group.

```
MODEL:EX10.3
   /* Computer system with several remote terminal groups. */
   /* Groups connected to system by polled communication   */
   /* line.  Flow control and packetizing of messages.     */
```

```
METHOD:simulation
NUMERIC IDENTIFIERS:no_terms /*per group*/ thinktime
   NO_TERMS:10
   THINKTIME:20
NUMERIC IDENTIFIERS:control data
   CONTROL:0    /*Code to be used for control messages*/
   DATA:1       /*Code to be used for data messages*/
NUMERIC IDENTIFIERS:group msg_type msg_leng
   GROUP:0      /*JV to be used to indicate group*/
   MSG_TYPE:1   /*JV to be used to indicate type*/
   MSG_LENG:2   /*JV to be used to indicate length*/
MAX JV:2
QUEUE:rtq /*response time*/
   TYPE:passive
   TOKENS:2147483647 /*"infinity"*/
   DSPL:fcfs
   ALLOCATE NODE LIST:begin_rt1 begin_rt2 begin_rt3
      NUMBERS OF TOKENS TO ALLOCATE:1
   RELEASE NODE LIST: end_rt1   end_rt2   end_rt3
QUEUE:comp_sysq
   TYPE:active
   DSPL:ps
   CLASS LIST:comp_sys
      WORK DEMANDS:1
   SERVER-
      RATE:1.4 2.0 2.25 2.4
DUMMY NODES:poll_in cntrl_rout cntrl_in1 cntrl_in2 cntrl_in3
```

Jobs are initially placed at the terminals to represent users. At the end of a think time, a job goes to set node MSG__CHAR, which sets job variables in terms of message characteristics, that is, the group producing the message, the fact that this is a data message, and the message length. The job then goes to node parameter BEGIN__RT, which is an allocate node for response time measurement. Jobs representing packets returning from the computing system go to fusion node ASSMBL__PKT. When all packets of a message have arrived at the fusion node, a single job representing the assembled message leaves the fusion node. That job goes to split node GEN__CNTRL to generate a control message, which will eventually allow another message to be sent, as we discuss shortly. The control message job goes to set node SET__CNTRL, which sets the job variables giving its characteristics. From the set node the control message job will go to communication line. The job representing the message goes to node parameter END__RT, a release node for response time measurement, and then goes to the terminals.

```
SUBMODEL:term_group
   NUMERIC PARAMETERS:group_no
   NODE PARAMETERS:begin_rt end_rt
   CHAIN PARAMETERS:c
   QUEUE:terminalsq
```

```
         TYPE:is
         CLASS LIST:terminals
            SERVICE TIMES:thinktime
      SET NODES:msg_char /*message characteristics*/
      ASSIGNMENT LIST:jv(group)=group_no                    ++
                      jv(msg_type)=data                     ++
                      jv(msg_leng)=uniform(24,640,1)
      SET NODES:set_cntrl
      ASSIGNMENT LIST:jv(group)=group_no                    ++
                      jv(msg_type)=control                  ++
                      jv(msg_leng)=32
      SPLIT NODES:gen_cntrl
      FUSION NODES:assmbl_pkt
      CHAIN:c
         TYPE:external
         INPUT:assmbl_pkt
         OUTPUT:set_cntrl
         :assmbl_pkt->gen_cntrl->end_rt set_cntrl;split
         :end_rt->terminals->msg_char->begin_rt
   END OF SUBMODEL TERM_GROUP
```

The key element of the POLL__LINE submodel is the use of the vector of priorities, CUR__PRIOR, which is used with the passive queue POLLING. There are three priority levels for a group: high priority for the flow control messages, medium priority for the data messages, and low priority for the polling job. Group *i* has highest priority given by CUR__PRIOR(*i*) for flow control messages, priority CUR__PRIOR(*i*)+1 for data messages, and priority CUR__PRIOR(*i*)+2 for the polling job. Polling is accomplished by the polling job creating a token at node FREE__MSGS and then waiting at allocate node CNT__ALLCTE until all higher priority jobs (flow control and data messages for the group being polled) have received the token, spent a service time at class MSG__IN, and then released the token at MSG__RELEAS. When the polling job receives the token, it increases the CUR__PRIOR value for the group just polled by three times NO__GROUPS, thus giving the group just polled the lowest priority.

```
   SUBMODEL:poll_line
      NUMERIC PARAMETERS:no_groups
      NODE PARAMETERS:inboundin inboundout
      CHAIN PARAMETERS:c
      GLOBAL VARIABLES:cur_group cur_prior(no_groups)
         CUR_GROUP:1
         CUR_PRIOR:0
      QUEUE:polling
         TYPE:passive
         TOKENS:0
         DSPL:prty
         ALLOCATE NODE LIST:msg_allcte
            NUMBERS OF TOKENS TO ALLOCATE:1
            PRIORITIES:cur_prior(jv(group))+jv(msg_type)
         ALLOCATE NODE LIST:cnt_allcte
            NUMBERS OF TOKENS TO ALLOCATE:1
            PRIORITIES:cur_prior(cur_group)+2
         RELEASE NODE LIST:msg_releas
         DESTROY NODE LIST:cnt_dstroy
         CREATE NODE LIST:free_msgs
            NUMBERS OF TOKENS TO CREATE:1
      QUEUE:inbound
         TYPE:fcfs
         CLASS LIST:msg_in
            SERVICE TIMES:standard(jv(msg_leng),0)/2400
         CLASS LIST:cnt_in
            SERVICE TIMES:32/2400
      QUEUE:outbound
         TYPE:prty
         CLASS LIST:msg_out
            SERVICE TIMES:standard(jv(msg_leng),0)/2400
            PRIORITIES:2
         CLASS LIST:cnt_out
            SERVICE TIMES:32/2400
            PRIORITIES:1
      SET NODES:new_cur
      ASSIGNMENT LIST:cur_prior(cur_group)=                          ++
                                 cur_prior(cur_group)+3*no_groups    ++
                      cur_group=(cur_group mod no_groups)+1
      SET NODES:init_prior
      ASSIGNMENT LIST:cur_prior(cur_group)=cur_group*3-2             ++
                      cur_group=cur_group+1
      SET NODES:init_group
      ASSIGNMENT LIST:cur_group=1
      CHAIN:c
         TYPE:external
         INPUT:msg_out
         OUTPUT:msg_out
         :inboundin->msg_allcte->msg_in->msg_releas->inboundout
      CHAIN:pollingjob
         TYPE:closed
         POPULATION:1
         :init_prior->init_prior;if(cur_group<=no_groups)
         :init_prior->init_group;if(t)
         :init_group->cnt_out->free_msgs->cnt_allcte->cnt_dstroy
         :cnt_dstroy->new_cur->cnt_in->cnt_out
   END OF SUBMODEL POLL_LINE
```

In the FLOW__N__PKT submodel a job representing a message from the computer system goes to set node OUTBND__LNG to establish the length of the message. The job then goes to allocate node FLOWALLCTE to wait for a token. A token will be made available by a job representing a flow control message arriving from node parameter CNTRL__IN and going to create node NEW__FLOW. When a job waiting at FLOWALLCTE gets a token, it will then generate new jobs representing packets at fission node

PACKETIZE. Set node REMOVE__PKT decrements the message length by 240 (the number of data bits in a packet), and set node NEW__PKT sets the new packet's JV(MSG__LNG) to 256 (data bits plus control bits).

```
SUBMODEL:flow_n_pkt
/* flow control and message packetization submodel.  One */
/* invocation of this for every invocation of term_group */
   NODE PARAMETERS:cntrl_in
   CHAIN PARAMETERS:c
   QUEUE:flow_cntrl
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:flowallcte
         NUMBERS OF TOKENS TO ALLOCATE:1
      DESTROY NODE LIST:flowdstroy
      CREATE NODE LIST:new_flow
         NUMBERS OF TOKENS TO CREATE:1
   SET NODES:outbnd_lng
   ASSIGNMENT LIST:jv(msg_leng)=standard(800,1)
   SET NODES:remove_pkt
   ASSIGNMENT LIST:jv(msg_leng)=jv(msg_leng)-240
   SET NODES:new_pkt
   ASSIGNMENT LIST:jv(msg_leng)=256
   FISSION NODES:packetize
   DUMMY NODES:outputport
   CHAIN:c
      TYPE:external
      INPUT:outbnd_lng
      OUTPUT:outputport
      :outbnd_lng->flowallcte->flowdstroy
      :flowdstroy->packetize outputport;if(jv(msg_leng)>256) if(t)
      :packetize->remove_pkt new_pkt;fission
      :remove_pkt->packetize outputport;if(jv(msg_leng)>256) if(t)
      :new_pkt->outputport
      :cntrl_in->new_flow->sink
 END OF SUBMODEL FLOW_N_PKT
 INVOCATION:group1
    TYPE:term_group
    GROUP_NO:1
    BEGIN_RT:begin_rt1
    END_RT:end_rt1
    C:c
 INVOCATION:group2
    TYPE:term_group
    GROUP_NO:2
    BEGIN_RT:begin_rt2
    END_RT:end_rt2
    C:c
 INVOCATION:group3
    TYPE:term_group
    GROUP_NO:3
```

```
      BEGIN_RT:begin_rt3
      END_RT:end_rt3
      C:c
   INVOCATION:line
      TYPE:poll_line
      NO_GROUPS:3
      INBOUNDIN:poll_in
      INBOUNDOUT:cntrl_rout
      C:c
   INVOCATION:flow1
      TYPE:flow_n_pkt
      CNTRL_IN:cntrl_in1
      C:c
   INVOCATION:flow2
      TYPE:flow_n_pkt
      CNTRL_IN:cntrl_in2
      C:c
   INVOCATION:flow3
      TYPE:flow_n_pkt
      CNTRL_IN:cntrl_in3
      C:c
   CHAIN:c
      TYPE:open
      :begin_rt1 begin_rt2 begin_rt3->poll_in
      :cntrl_rout->comp_sys;if(jv(msg_type)=data)
      :cntrl_rout->cntrl_in1;if(jv(group)=1)
      :cntrl_rout->cntrl_in2;if(jv(group)=2)
      :cntrl_rout->cntrl_in3;if(jv(group)=3)
      :comp_sys->flow1.input;if(jv(group)=1)
      :comp_sys->flow2.input;if(jv(group)=2)
      :comp_sys->flow3.input;if(jv(group)=3)
      :flow1.output flow2.output flow3.output->line.input
      :line.output->group1.input;if(jv(group)=1)
      :line.output->group2.input;if(jv(group)=2)
      :line.output->group3.input;if(jv(group)=3)
      :group1.output group2.output group3.output->poll_in
   QUEUES FOR QUEUEING TIME DIST:rtq
      VALUES:.5 1 2 4 8
   NODES FOR QUEUEING TIME DIST:begin_rt1 begin_rt2 begin_rt3
      VALUES:.5 1 2 4 8
   CONFIDENCE INTERVAL METHOD:spectral
   INITIAL STATE DEFINITION-
   CHAIN:line.pollingjob
      NODE LIST:line.init_prior
      INIT POP:1
   CHAIN:c
      NODE LIST:group1.terminals group2.terminals group3.terminals
      INIT POP: no_terms          no_terms          no_terms
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      CONFIDENCE INTERVAL QUEUES:rtq rtq comp_sysq
         MEASURES:                qt  qtd qt
         ALLOWED WIDTHS:          10  10  10
      CONFIDENCE INTERVAL NODES:begin_rt1 begin_rt2 begin_rt3
         MEASURES:                qt        qt        qt
         ALLOWED WIDTHS:          100       100       100
   INITIAL PORTION DISCARDED:10
   INITIAL PERIOD LIMITS-
      QUEUES FOR DEPARTURE COUNTS:rtq
         DEPARTURES:1000
   LIMIT - CP SECONDS:500
   TRACE:no
END
```

The following results from the simulation display some performance measures for utilizations, throughput, mean queueing time, and mean queue length. The simulation was continued until the accuracy criteria were satisfied for another sampling period. Some additional performance measures are also illustrated.

```
RESQ2 VERSION DATE: APRIL 3, 1982 - TIME: 17:56:53  DATE: 04/03/82
MODEL:EX10.3
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.   2930 DISCARDED EVENTS

                  SIMULATED TIME:     5028.19141
                        CPU TIME:         381.07
                NUMBER OF EVENTS:         227673

WHAT:ut(line.msg_in,line.cnt_in,line.msg_out,line.cnt_out)

INVOCATION     ELEMENT        UTILIZATION
LINE            MSG_IN         0.20631
LINE            CNT_IN         0.23099
LINE            MSG_OUT        0.48462
LINE            CNT_OUT        0.23007

WHAT:tp(rtq,begin_rt1,begin_rt2,begin_rt3)

INVOCATION     ELEMENT        THROUGHPUT
               RTQ            1.35874
                BEGIN_RT1      0.46657
                BEGIN_RT2      0.44887
                BEGIN_RT3      0.44330

WHAT:qtbo(rtq,begin_rt1,begin_rt2,begin_rt3,comp_sysq)

INVOCATION     ELEMENT        MEAN QUEUEING TIME
               RTQ            2.30391(2.21360,2.39422) 7.8%
```

```
                  BEGIN_RT1        2.29731(2.21029,2.38434) 7.6%
                  BEGIN_RT2        2.35772(2.27390,2.44153) 7.1%
                  BEGIN_RT3        2.25636(2.15468,2.35804) 9.0%
                 COMP_SYSQ        1.20234(1.14318,1.26150) 9.8%

WHAT:ql(rtq,begin_rt1,begin_rt2,begin_rt3,comp_sysq)

INVOCATION       ELEMENT          MEAN QUEUE LENGTH
                 RTQ              3.13178
                  BEGIN_RT1        1.07279
                  BEGIN_RT2        1.05869
                  BEGIN_RT3        1.00030
                 COMP_SYSQ        1.63460

WHAT:
CONTINUE RUN:yes
EXTRA SAMPLING PERIODS:1
LIMIT - CP SECONDS:1000

SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
SAMPLING PERIOD END: RTQ DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.   2930 DISCARDED EVENTS

                   SIMULATED TIME:      7542.98047
                         CPU TIME:          570.16
                 NUMBER OF EVENTS:          343411

WHAT:ut(line.msg_in,line.cnt_in,line.msg_out,line.cnt_out)

INVOCATION       ELEMENT          UTILIZATION
LINE              MSG_IN           0.20599
LINE              CNT_IN           0.23301
LINE              MSG_OUT          0.47914
LINE              CNT_OUT          0.23180

WHAT:tp(rtq,begin_rt1,begin_rt2,begin_rt3)

INVOCATION       ELEMENT          THROUGHPUT
                 RTQ              1.35861
                  BEGIN_RT1        0.45274
                  BEGIN_RT2        0.45141
                  BEGIN_RT3        0.45446

WHAT:qtbo(rtq,begin_rt1,begin_rt2,begin_rt3,comp_sysq)
```

```
INVOCATION      ELEMENT          MEAN QUEUEING TIME
                RTQ              2.27488(2.21307,2.33669) 5.4%
                 BEGIN_RT1        2.27176(2.20855,2.33498) 5.6%
                 BEGIN_RT2        2.31046(2.22283,2.39809) 7.6%
                 BEGIN_RT3        2.24264(2.14938,2.33590) 8.3%
                COMP_SYSQ        1.19929(1.16632,1.23225) 5.5%

WHAT:ql(rtq,begin_rt1,begin_rt2,begin_rt3,comp_sysq)

INVOCATION      ELEMENT          MEAN QUEUE LENGTH
                RTQ              3.09146
                 BEGIN_RT1        1.02852
                 BEGIN_RT2        1.04340
                 BEGIN_RT3        1.01954
                COMP_SYSQ        1.62968

WHAT:qtdbo(*)

INVOCATION      ELEMENT          QUEUEING TIME DISTRIBUTION
                RTQ              5.00E-01:0.03708(0.03404,0.04012) 0.6%
                                 1.00E+00:0.19633(0.18753,0.20513) 1.8%
                                 2.00E+00:0.54167(0.52525,0.55808) 3.3%
                                 4.00E+00:0.87529(0.86545,0.88513) 2.0%
                                 8.00E+00:0.98985(0.98771,0.99199) 0.4%
                 BEGIN_RT1        5.00E-01:0.03748(0.02998,0.04498) 1.5%
                                  1.00E+00:0.19590(0.18600,0.20580) 2.0%
                                  2.00E+00:0.53206(0.51399,0.55013) 3.6%
                                  4.00E+00:0.87877(0.86588,0.89166) 2.6%
                                  8.00E+00:0.99180(0.98863,0.99497) 0.6%
                 BEGIN_RT2        5.00E-01:0.03465(0.03048,0.03883) 0.8%
                                  1.00E+00:0.18767(0.16722,0.20811) 4.1%
                                  2.00E+00:0.52658(0.49773,0.55543) 5.8%
                                  4.00E+00:0.86990(0.85482,0.88498) 3.0%
                                  8.00E+00:0.98913(0.98526,0.99301) 0.8%
                 BEGIN_RT3        5.00E-01:0.03909(0.03283,0.04535) 1.3%
                                  1.00E+00:0.20537(0.18920,0.22153) 3.2%
                                  2.00E+00:0.56622(0.54158,0.59086) 4.9%
                                  4.00E+00:0.87719(0.86155,0.89283) 3.1%
                                  8.00E+00:0.98862(0.98558,0.99166) 0.6%

WHAT:qt(line.msg_allcte,line.msg_out)

INVOCATION      ELEMENT          MEAN QUEUEING TIME
LINE             MSG_ALLCTE       0.19342
LINE             MSG_OUT          0.58095

WHAT:ql(flow1.flowallcte,flow2.flowallcte,flow3.flowallcte)

INVOCATION      ELEMENT          MEAN QUEUE LENGTH
FLOW1            FLOWALLCTE       0.10062
FLOW2            FLOWALLCTE       0.10598
FLOW3            FLOWALLCTE       0.10036
```

```
WHAT:gv

INVOCATION         ELEMENT             FINAL VALUES OF GLOBAL VARIABLES
LINE               CUR_GROUP           2.00000
LINE               CUR_PRIOR(1)        3.9830E+05
LINE               CUR_PRIOR(2)        3.9829E+05
LINE               CUR_PRIOR(3)        3.9829E+05
```

10.4. FURTHER READING

The model described in Section 10.1 is a result of some conversations with J. Voldman [185]. Section 10.2 is based on a model presented in Kadar [86] and Kadar, MacNair, and Tang [87]. The model in Section 10.3 is from Sauer, MacNair, and Kurose [159, 160]. There is also a similar model in Sauer, MacNair, and Kurose [161]. Other communication network model references include Bharath-Kumar and Kermani [21], Kleinrock [94], Reiser [143], Sauer and MacNair [156], Schwartz [165, 166], Stewart [176], and Wong [189].

10.5. EXERCISES

10.1 Construct and solve models of communication networks you are familiar with.

10.2 Construct models which incorporate the following protocols: acknowledgements, time outs, packetizing of messages, adaptive routing, and flow control.

10.3 Construct models of different local area network schemes.

10.4 Construct a model of a communication system which switches telephone conversations between multiple input and output ports.

CHAPTER 11

MANUFACTURING SYSTEMS

Just as the modeling of computer systems and communication networks has increased in popularity in recent years, modeling of manufacturing systems is also increasing. With the advent of flexible manufacturing systems, modeling of manufacturing systems is becoming particularly important. This chapter discusses several simple models of manufacturing systems. The first section presents a model of tool failures using a preemptive priority queue. In Section 11.2, there is a model of load balancing in a system with parallel resources. The third section illustrates a simple way of representing a robotic type of system. Section 11.4 depicts a system with merging lines where parts from the different lines have to be synchronized. Although simulation is used in most of the models of this chapter, analytic models of flexible manufacturing systems have been used with good success.

11.1. TOOL FAILURES

Manufacturing systems frequently consist of many tools to perform various types of tasks so that tool failure has a major impact on the performance of the system. If these failures are neglected, the model predictions will be overly optimistic. The model discussed in this section presents a simple way of depicting tool failures. Figure 11.1 illustrates a model with one tool. Tasks arriving at the source of this open model go to a transfer unit. A portion of the tasks are sent to the tool, and the rest bypass the tool. All tasks eventually go to another transfer unit and then leave at the sink. Tool failures are represented by a single task that travels along a separate path shown at the bottom of the diagram. There is an infinite server representing the time the tool is in operation. Then the failure task goes to a set node to take a sample from a distribution to determine the failure time. A second class at the tool service center represents the failure time. Since the tool service center is defined as a priority queue, the failure task has priority over the normal tasks. The time the failure task spends in service at the tool models the down time. After the failure is repaired, the failure task goes to another set node to log some statistics related to the failure and returns to the infinite server for the next operational period.

Because of using a global variable and priority queueing, we are using simulation to solve the model. The model contains one numeric parameter representing the mean time between arrivals. The global variable is a vector with three elements. The first element is the number of failures, the second

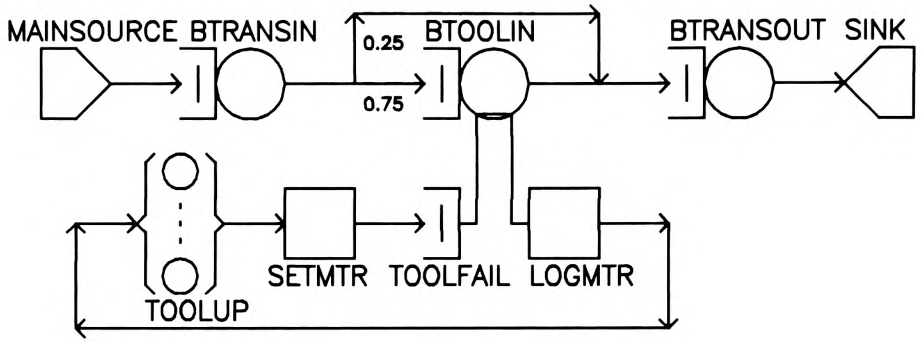


Figure 11.1. Model Diagram of Tool Failures

the total failure time, and the third the average failure time. There is an active queue for the input transfer unit. The tool is a preemptive priority queue with two classes. The failure tasks have priority over the normal tasks processed at the tool. The down time is taken from a value previously stored in a job variable at set node SETMTR. The output transfer unit is an active queue. The tool up time is modeled as an infinite server queue. This could have been an FCFS server, since only a single failure task is in use. There are two set nodes. SETMTR takes a sample from an exponential distribution which is used as the tool down time. The other set node, LOGMTR, adds one to the number of failures, adds the current failure time to the cumulative failure time, and divides the total failure time by the number of failures to calculate the average failure time. The open chain definition is straightforward. Seventy-five percent of the tasks go to the tool for processing and 25 percent bypass the tool. The regenerative method is used to produce confidence intervals. The regeneration state is one task at the TOOLUP queue. A 90 percent level of confidence is being used to determine the confidence interval widths. Since the sequential stopping rule is not employed, the run will stop after approximately 10,000 departures from the TOOL queue.

```
MODEL:EX11.1
```

```
   METHOD:simulation
   NUMERIC PARAMETERS:mtba
   GLOBAL VARIABLES:faillog(3)
      FAILLOG:0 0 0
   QUEUE:transin
      TYPE:fcfs
      CLASS LIST:btransin
         SERVICE TIMES:4
   QUEUE:tool
      TYPE:prtypr
      PREEMPT DIST:1
      CLASS LIST:      btoolin toolfail
         SERVICE TIMES:20      constant(jv(0))
         PRIORITIES:   2       1
   QUEUE:transout
      TYPE:fcfs
      CLASS LIST:btransout
         SERVICE TIMES:4
   QUEUE:qtoolup
      TYPE:is
      CLASS LIST:toolup
         SERVICE TIMES:3600   /* mean time between failures */
   SET NODES:setmtr
      ASSIGNMENT LIST:jv(0)=exponential(60)
   SET NODES:logmtr
      ASSIGNMENT LIST:faillog(1)=faillog(1)+1 ++
                      faillog(2)=faillog(2)+jv(0) ++
                      faillog(3)=faillog(2)/faillog(1)
   CHAIN:main
      TYPE:open
      SOURCE LIST:mainsource
      ARRIVAL TIMES:mtba
      :mainsource->btransin->btoolin btransout;.75 .25
      :btoolin->btransout->sink
      :toolup->setmtr->toolfail->logmtr->toolup
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CHAIN:main
      NODE LIST:toolup
         REGEN POP:1
         INIT POP:1
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:no
   RUN GUIDELINES -
      QUEUES FOR DEPARTURE COUNTS:tool
         DEPARTURES:10000
   LIMIT - CP SECONDS:250
   TRACE:no
END
```

The run stops after the 10,000 departures from the TOOL queue. The elapsed simulation time, CPU time, number of events, and number of regeneration cycles are shown. We have gotten a large number of regeneration cycles. The failures are probably having an effect on the performance measures. There were 134 failures, with each failure lasting about 60 seconds.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 15:05:59  DATE: 05/13/84
MODEL:EX11.1
MTBA:40
WARNING -- MODEL MAY NOT BE TRULY REGENERATIVE
           BECAUSE OF USE OF GLOBAL VARIABLES
RUN END: TOOL DEPARTURE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                  SIMULATED TIME:     5.2124E+05
                        CPU TIME:          18.53
                NUMBER OF EVENTS:          49495
                NUMBER OF CYCLES:           6445

WHAT:ALLBO

ELEMENT        UTILIZATION
TRANSIN        0.10128(0.09921,0.10334) 0.4%
TOOL           0.39835(0.38753,0.40916) 2.2%
 BTOOLIN        0.38270(0.37259,0.39282) 2.0%
 TOOLFAIL       0.01564(-0.00604,0.03733) 4.3%
TRANSOUT       0.09964(0.09761,0.10167) 0.4%
QTOOLUP        0.00000(0.00000,0.00000)

ELEMENT        THROUGHPUT
TRANSIN        0.02517(0.02481,0.02553) 2.8%
TOOL           0.01919(0.01884,0.01953) 3.6%
 BTOOLIN        0.01893(0.01859,0.01927) 3.6%
 TOOLFAIL       2.5708E-04(-3.3968E-06,5.1756E-04) 202.6%
TRANSOUT       0.02517(0.02481,0.02553) 2.8%
QTOOLUP        2.5708E-04(-3.3968E-06,5.1756E-04) 202.6%
SETMTR         2.5708E-04
LOGMTR         2.5708E-04
MAINSOURCE     0.02517
SINK           0.02517

ELEMENT        MEAN QUEUE LENGTH
TRANSIN        0.11300(0.11032,0.11567) 4.7%
TOOL           0.69370(0.64762,0.73977) 13.3%
 BTOOLIN        0.67805(0.63363,0.72248) 13.1%
 TOOLFAIL       0.01564(-0.00604,0.03733) 277.3%
TRANSOUT       0.11010(0.10756,0.11265) 4.6%
QTOOLUP        0.98436(0.96267,1.00604) 4.4%

ELEMENT        STANDARD DEVIATION OF QUEUE LENGTH
TRANSIN        0.35553
TOOL           1.17756
 BTOOLIN        1.15602
 TOOLFAIL        0.12409
TRANSOUT        0.34743
QTOOLUP         0.12409

ELEMENT         MEAN QUEUEING TIME
TRANSIN         4.48917(4.40901,4.56933) 3.6%
TOOL            36.15456(34.23720,38.07191) 10.6%
 BTOOLIN         35.81923(33.94498,37.69348) 10.5%
 TOOLFAIL        60.84576(52.68535,69.00618) 26.8%
TRANSOUT        4.37425(4.29914,4.44936) 3.4%
QTOOLUP         3829.00464(3269.20581,4388.80078) 29.2%

ELEMENT         STANDARD DEVIATION OF QUEUEING TIME
TRANSIN         4.57808
TOOL            40.50555
 BTOOLIN         40.10490
 TOOLFAIL        58.53436
TRANSOUT        4.33619
QTOOLUP         3651.13574

ELEMENT         MAXIMUM QUEUE LENGTH
TRANSIN         4
TOOL            16
 BTOOLIN         16
 TOOLFAIL         1
TRANSOUT        5
QTOOLUP         1

ELEMENT         MAXIMUM QUEUEING TIME
TRANSIN         41.93031
TOOL            408.18262
 BTOOLIN         408.18262
 TOOLFAIL        399.50244
TRANSOUT        46.74709
QTOOLUP         1.9568E+04

ELEMENT         OPEN CHAIN POPULATION
MAIN            1.90115(1.86011,1.94219) 4.3%

ELEMENT         OPEN CHAIN RESPONSE TIME
MAIN            75.53021(74.09219,76.96822) 3.8%

WHAT:GV

ELEMENT         FINAL VALUES OF GLOBAL VARIABLES
FAILLOG(1)      134.00000
FAILLOG(2)      8153.33203
FAILLOG(3)      60.84576
```

Now we solve the model for six different values of the mean time between arrivals. We start at 40 seconds and reduce it to 15 seconds. We plot the mean queue lengths and the mean queueing times for the input transfer unit and the tool. These plots are shown in Figure 11.2.

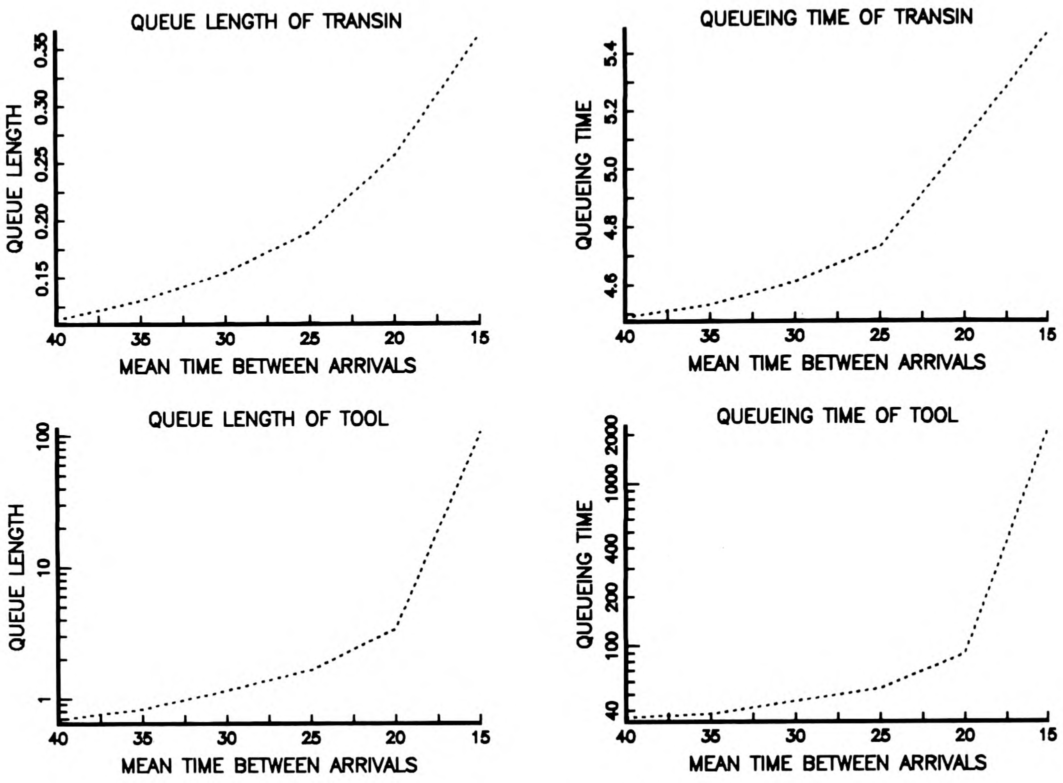


Figure 11.2. Graphical Results of Tool Failure Model

11.2. LOAD BALANCING

Some systems contain multiple devices operating in parallel in which we can increase the throughput in the system by balancing the load delivered to each device. This section describes such a system where it is not obvious what proportion of jobs should be sent to each device. Figure 11.3 shows four mold presses, eight transfer units, and six conveyors. Tasks arriving at the source go to a transfer unit to be placed on a conveyor or the first mold press. A transfer unit is always needed after every conveyor and every mold press. Tasks which complete at a mold press go through a series of transfer units and conveyors before leaving the system. The problem is to determine what proportion of jobs to send to each mold press.

The model contains a numeric parameter representing the proportion of jobs sent to each mold press. It is a vector with four elements. These

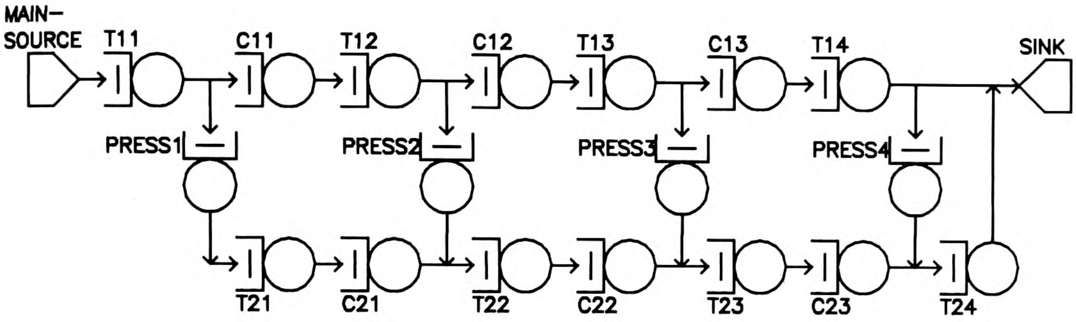


Figure 11.3. Model Diagram of Mold Presses

proportions cannot be used directly as the routing probabilities. A numeric identifier for the routing probabilities is calculated based on the proportions. The service centers for all the resources in the system are active queues. The transfer units and the mold presses are FCFS servers, and the conveyors are infinite servers. This model can be solved analytically. The chain is an open chain, and the routing is a straightforward representation of the information shown in the model diagram. The decision points use the routing probabilities calculated for the numeric identifier vector Q.

```
MODEL:EX11.2
   METHOD:numerical
   NUMERIC PARAMETERS:p(4)
   NUMERIC IDENTIFIERS:q(4)
      Q:p(1)                                         ++
        p(2)/(1-q(1))                                ++
        p(3)/(1-q(1))*(1-q(2))                       ++
        p(4)/(1-q(1))*(1-q(2))*(1-q(3))
   QUEUE:qt11
      TYPE:fcfs
      CLASS LIST:t11
         SERVICE TIMES:4
   QUEUE:qt12
      TYPE:fcfs
      CLASS LIST:t12
         SERVICE TIMES:4
   QUEUE:qt13
      TYPE:fcfs
      CLASS LIST:t13
         SERVICE TIMES:4
   QUEUE:qt14
      TYPE:fcfs
      CLASS LIST:t14
         SERVICE TIMES:4
   QUEUE:qt21
      TYPE:fcfs
      CLASS LIST:t21
         SERVICE TIMES:4
   QUEUE:qt22
      TYPE:fcfs
      CLASS LIST:t22
         SERVICE TIMES:4
   QUEUE:qt23
      TYPE:fcfs
      CLASS LIST:t23
         SERVICE TIMES:4
   QUEUE:qt24
      TYPE:fcfs
      CLASS LIST:t24
         SERVICE TIMES:4
   QUEUE:qpress1
      TYPE:fcfs
      CLASS LIST:press1
         SERVICE TIMES:40
   QUEUE:qpress2
      TYPE:fcfs
      CLASS LIST:press2
         SERVICE TIMES:40
   QUEUE:qpress3
      TYPE:fcfs
      CLASS LIST:press3
         SERVICE TIMES:40
   QUEUE:qpress4
      TYPE:fcfs
      CLASS LIST:press4
         SERVICE TIMES:40
   QUEUE:qc11
      TYPE:is
      CLASS LIST:c11
         SERVICE TIMES:5
   QUEUE:qc12
      TYPE:is
      CLASS LIST:c12
         SERVICE TIMES:5
   QUEUE:qc13
      TYPE:is
      CLASS LIST:c13
```

```
         SERVICE TIMES:5
   QUEUE:qc21
      TYPE:is
      CLASS LIST:c21
         SERVICE TIMES:5
   QUEUE:qc22
      TYPE:is
      CLASS LIST:c22
         SERVICE TIMES:5
   QUEUE:qc23
      TYPE:is
      CLASS LIST:c23
         SERVICE TIMES:5
   CHAIN:main
      TYPE:open
      SOURCE LIST:mainsource
      ARRIVAL TIMES:15
      :mainsource->t11->press1 c11;q(1) 1-q(1)
      :c11->t12->press2 c12;q(2) 1-q(2)
      :c12->t13->press3 c13;q(3) 1-q(3)
      :c13->t14->press4 sink;q(4) 1-q(4)
      :press1->t21
      :press2->t22
      :press3->t23
      :press4->t24
      :t21->c21->t22->c22->t23->c23->t24->sink
END
```

The most obvious proportion of jobs to send to each mold press is 25 percent. The results that follow use this equal loading scheme, but we will see that it is not the best solution.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 15:27:52  DATE: 05/25/84
MODEL:EX11.2
P:.25 .25 .25 .25
NO ERRORS DETECTED DURING NUMERICAL SOLUTION

WHAT:ALL

ELEMENT         UTILIZATION
QT11            0.26667
QT12            0.20000
QT13            0.13333
QT14            0.10370
QT21            0.06667
QT22            0.13333
QT23            0.16296
QT24            0.18089
QPRESS1         0.66667
QPRESS2         0.66667
QPRESS3         0.29630
QPRESS4         0.17924
```

```
QC11             0.00000
QC12             0.00000
QC13             0.00000
QC21             0.00000
QC22             0.00000
QC23             0.00000

ELEMENT          THROUGHPUT
QT11             0.06667
QT12             0.05000
QT13             0.03333
QT14             0.02593
QT21             0.01667
QT22             0.03333
QT23             0.04074
QT24             0.04522
QPRESS1          0.01667
QPRESS2          0.01667
QPRESS3          7.4074E-03
QPRESS4          4.4810E-03
QC11             0.05000
QC12             0.03333
QC13             0.02593
QC21             0.01667
QC22             0.03333
QC23             0.04074

ELEMENT          MEAN QUEUE LENGTH
QT11             0.36364
QT12             0.25000
QT13             0.15385
QT14             0.11570
QT21             0.07143
QT22             0.15385
QT23             0.19469
QT24             0.22083
QPRESS1          2.00000
QPRESS2          2.00000
QPRESS3          0.42105
QPRESS4          0.21838
QC11             0.25000
QC12             0.16667
QC13             0.12963
QC21             0.08333
QC22             0.16667
QC23             0.20370

ELEMENT          MEAN QUEUEING TIME
QT11             5.45455
QT12             5.00000
QT13             4.61538
QT14             4.46281
QT21             4.28571
```

```
QT22                4.61538
QT23                4.77876
QT24                4.88333
QPRESS1             119.99998
QPRESS2             119.99998
QPRESS3             56.84210
QPRESS4             48.73537
QC11                5.00000
QC12                5.00000
QC13                5.00000
QC21                5.00000
QC22                5.00000
QC23                5.00000

ELEMENT             OPEN CHAIN POPULATION
MAIN                7.16342

ELEMENT             OPEN CHAIN RESPONSE TIME
MAIN                107.45126
```

Figure 11.4 shows a simple graph of different loadings versus the process time. The process time is the open chain response time. The first loading was the equal proportion case (.25, .25, .25, .25). The following table shows all four loadings:

	P(1)	P(2)	P(3)	P(4)
1	0.25	0.25	0.25	0.25
2	0.20	0.20	0.30	0.30
3	0.15	0.20	0.30	0.35
4	0.15	0.15	0.25	0.45

As the graph shows, when we send a larger proportion of jobs to the furthest mold press, the process time decreases. This is because of the time it takes to use the transfer units and the conveyors.

11.3. A ROBOT

The system modeled in this section employs a robot to perform some simple processing automatically. Pieces to be riveted move along a conveyor to an orientation station. There is one spot at the orientation station for one piece to be oriented properly. After being oriented, a single robot picks up one piece and moves it to the riveting machine. As soon as a piece is removed from the orientation station, a new piece can begin its orientation. After moving to the riveting machine, the piece is riveted and moved by the robot to an output station. There is one spot at the output station, and it takes time to put a piece down at the output station and to move onto

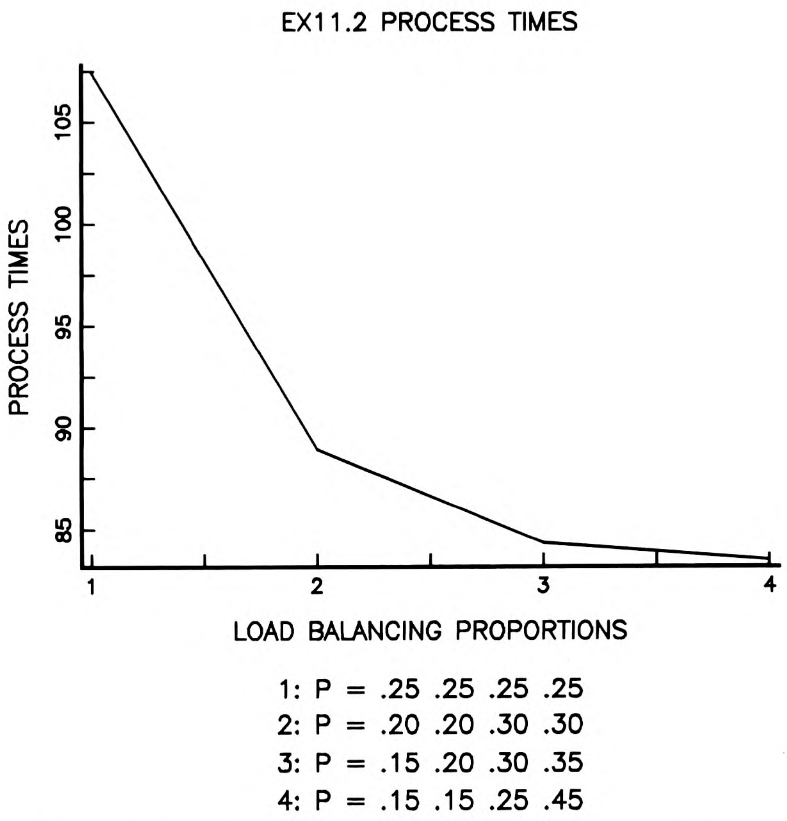


Figure 11.4. Graphical Results of Mold Press Model

another conveyor and down the line. A diagram of the model is shown in Figure 11.5.

The model contains numeric parameters for the mean service times at the orientation station, the pickup operation, the time to move to the riveting machine, the riveting time, the time to move to the output station, the time to put a piece on the output station, and the time to remove a piece from the output station. The mean interarrival time between the pieces is also a numeric parameter. There are passive queues for the orientation staging area, the robot, and the output staging area. Each passive queue is defined with a single token. The service times are represented by active FCFS service centers. The routing is a straightforward implementation of the system description. The regenerative method is used to construct the confidence intervals at the 90 percent level of confidence. The sequential stopping rule is employed to detect when the specified level of accuracy is achieved.

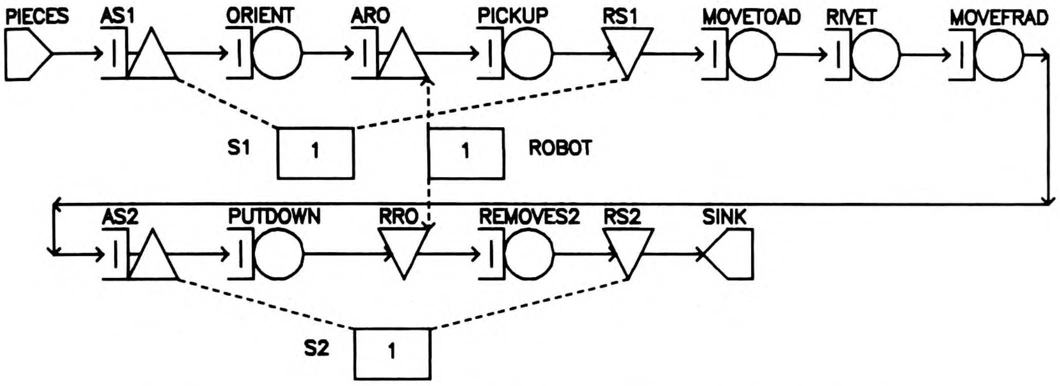


Figure 11.5. Model Diagram of a Simple Robotic System

```
MODEL:EX11.3
   METHOD:simulation
   NUMERIC PARAMETERS:stor stpi stmt stri stmf stpu stre iat
   QUEUE:stagearea1
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:as1
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:rs1
   QUEUE:orientq
      TYPE:fcfs
      CLASS LIST:orient
         SERVICE TIMES:stor
   QUEUE:robot
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:aro
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:rro
   QUEUE:pickupq
      TYPE:fcfs
      CLASS LIST:pickup
```

```
         SERVICE TIMES:stpi
   QUEUE:movetoadq
      TYPE:fcfs
      CLASS LIST:movetoad
         SERVICE TIMES:stmt
   QUEUE:rivetq
      TYPE:fcfs
      CLASS LIST:rivet
         SERVICE TIMES:stri
   QUEUE:movefradq
      TYPE:fcfs
      CLASS LIST:movefrad
         SERVICE TIMES:stmf
   QUEUE:stagearea2
      TYPE:passive
      TOKENS:1
      DSPL:fcfs
      ALLOCATE NODE LIST:as2
         NUMBERS OF TOKENS TO ALLOCATE:1
      RELEASE NODE LIST:rs2
   QUEUE:putdownq
      TYPE:fcfs
      CLASS LIST:putdown
         SERVICE TIMES:stpu
   QUEUE:removes2q
      TYPE:fcfs
      CLASS LIST:removes2
         SERVICE TIMES:stre
   CHAIN:chain1
      TYPE:open
      SOURCE LIST:pieces
      ARRIVAL TIMES:iat
      :pieces->as1->orient->aro->pickup->rs1
      :rs1->movetoad->rivet->movefrad
      :movefrad->as2->putdown->rro->removes2->rs2->sink
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:robot
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:200
   LIMIT - CP SECONDS:50
   TRACE:no
END
```

The model was run with an arbitrary set of parameter values. These parameter values produced the results that follow. This is a very short simulation run, but some of the results are fairly accurate.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 13:08:06  DATE: 05/26/84
MODEL:EX11.3
STOR:1.5
STPI:0.5
STMT:1.5
STRI:5.5
STMF:1.5
STPU:0.5
STRE:1.5
IAT:15
SAMPLING PERIOD END: CYCLE GUIDELINE
SAMPLING PERIOD END: CYCLE GUIDELINE
NO ERRORS DETECTED DURING SIMULATION.

                      SIMULATED TIME:      2.0172E+04
                            CPU TIME:            4.89
                    NUMBER OF EVENTS:           10736
                    NUMBER OF CYCLES:             400

WHAT:ALLBO

ELEMENT         UTILIZATION
STAGEAREA1      0.43068(0.39040,0.47096) 8.1%
ROBOT           0.62161(0.58882,0.65439) 6.6%
STAGEAREA2      0.12934(0.12261,0.13608) 1.3%
ORIENTQ         0.10118(0.09547,0.10689) 1.1%
PICKUPQ         0.03273(0.03088,0.03458) 0.4%
MOVETOADQ       0.10032(0.09368,0.10697) 1.3%
RIVETQ          0.35379(0.33085,0.37674) 4.6%
MOVEFRADQ       0.10029(0.09432,0.10627) 1.2%
PUTDOWNQ        0.03221(0.03035,0.03407) 0.4%
REMOVES2Q       0.09713(0.09141,0.10285) 1.1%

ELEMENT         THROUGHPUT
STAGEAREA1      0.06653(0.06387,0.06919) 8.0%
ROBOT           0.06653(0.06387,0.06919) 8.0%
STAGEAREA2      0.06653(0.06387,0.06919) 8.0%
ORIENTQ         0.06653(0.06387,0.06919) 8.0%
PICKUPQ         0.06653(0.06387,0.06919) 8.0%
MOVETOADQ       0.06653(0.06387,0.06919) 8.0%
RIVETQ          0.06653(0.06387,0.06919) 8.0%
MOVEFRADQ       0.06653(0.06387,0.06919) 8.0%
PUTDOWNQ        0.06653(0.06387,0.06919) 8.0%
REMOVES2Q       0.06653(0.06387,0.06919) 8.0%
RS1             0.06653
RRO             0.06653
RS2             0.06653
PIECES          0.06653
SINK            0.06653

ELEMENT         MEAN QUEUE LENGTH
STAGEAREA1      0.90559(0.71895,1.09224) 41.2%
```

```
ROBOT            0.91838(0.85112,0.98565)  14.6%
STAGEAREA2       0.13160(0.12443,0.13876)  10.9%
ORIENTQ          0.10118(0.09547,0.10689)  11.3%
PICKUPQ          0.03273(0.03088,0.03458)  11.3%
MOVETOADQ        0.10032(0.09368,0.10697)  13.2%
RIVETQ           0.35379(0.33085,0.37674)  13.0%
MOVEFRADQ        0.10029(0.09432,0.10627)  11.9%
PUTDOWNQ         0.03221(0.03035,0.03407)  11.5%
REMOVES2Q        0.09713(0.09141,0.10285)  11.8%

ELEMENT          STANDARD DEVIATION OF QUEUE LENGTH
STAGEAREA1       1.45542
ROBOT            0.81762
STAGEAREA2       0.34465
ORIENTQ          0.30157
PICKUPQ          0.17792
MOVETOADQ        0.30043
RIVETQ           0.47815
MOVEFRADQ        0.30039
PUTDOWNQ         0.17656
REMOVES2Q        0.29614

ELEMENT          MEAN QUEUEING TIME
STAGEAREA1       13.61248(11.08257,16.14238) 37.2%
ROBOT            13.80473(13.14850,14.46096) 9.5%
STAGEAREA2       1.97810(1.90309,2.05310)  7.6%
ORIENTQ          1.52090(1.46083,1.58096)  7.9%
PICKUPQ          0.49195(0.47108,0.51281)  8.5%
MOVETOADQ        1.50803(1.43086,1.58521)  10.2%
RIVETQ           5.31809(5.07697,5.55922)  9.1%
MOVEFRADQ        1.50758(1.44455,1.57062)  8.4%
PUTDOWNQ         0.48420(0.46369,0.50471)  8.5%
REMOVES2Q        1.46005(1.39437,1.52572)  9.0%

ELEMENT          STANDARD DEVIATION OF QUEUEING TIME
STAGEAREA1       16.73315
ROBOT            8.72792
STAGEAREA2       1.55407
ORIENTQ          1.47744
PICKUPQ          0.50944
MOVETOADQ        1.49297
RIVETQ           5.77120
MOVEFRADQ        1.45702
PUTDOWNQ         0.47655
REMOVES2Q        1.43762

ELEMENT          MEAN TOKENS IN USE
STAGEAREA1       0.43068(0.39040,0.47096)  18.7%
ROBOT            0.62161(0.58882,0.65439)  10.5%
STAGEAREA2       0.12934(0.12261,0.13608)  10.4%
```

```
ELEMENT          MEAN TOTAL TOKENS IN POOL
STAGEAREA1       1.00000
ROBOT            1.00000
STAGEAREA2       1.00000

ELEMENT          MAXIMUM QUEUE LENGTH
STAGEAREA1       11
ROBOT            2
STAGEAREA2       2
ORIENTQ          1
PICKUPQ          1
MOVETOADQ        1
RIVETQ           1
MOVEFRADQ        1
PUTDOWNQ         1
REMOVES2Q        1

ELEMENT          MAXIMUM QUEUEING TIME
STAGEAREA1       106.27850
ROBOT            65.42307
STAGEAREA2       12.49809
ORIENTQ          9.92921
PICKUPQ          4.44203
MOVETOADQ        12.54598
RIVETQ           55.68083
MOVEFRADQ        10.61911
PUTDOWNQ         3.37284
REMOVES2Q        11.30502

ELEMENT          OPEN CHAIN POPULATION
CHAIN1           1.59160(1.37743,1.80577) 26.9%

ELEMENT          OPEN CHAIN RESPONSE TIME
CHAIN1           23.92429(21.22171,26.62685) 22.6%
```

11.4. MERGING LINES

In manufacturing systems there are often many different types of parts that must be merged together, and a single unit progresses on down the line. The diagram in Figure 11.6 shows a simple example of two parts from different lines merging together and one part continuing after the merger. This process is basically a synchronization procedure. The parts flowing down one line must be synchronized with the parts flowing down the other line. This is a perfect application of passive resources. There is a source and an active queue for each type of part. There is also a passive queue with a create, an allocate, and a destroy node for synchronization. A type one part creates a token for a type two part to use. A type two part creates a token for a type one part to use. When both types of parts are available, one part continues on down the line and the other leaves the model through the sink.

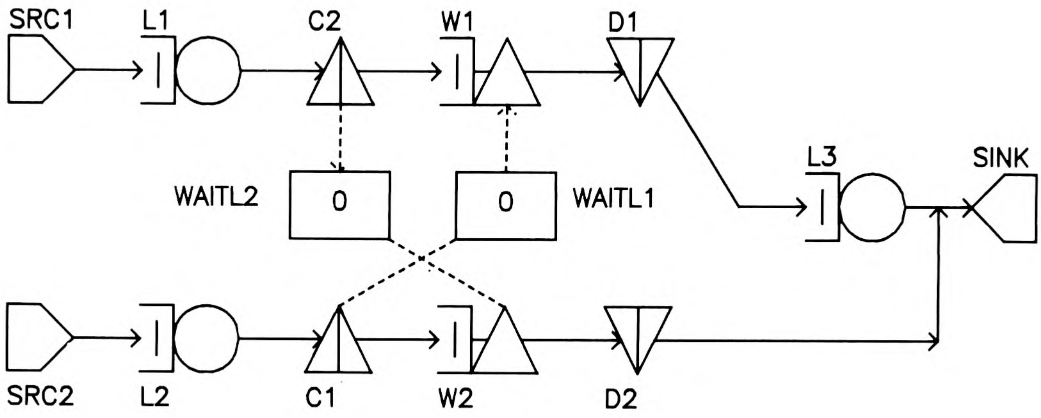


Figure 11.6. Model Diagram of Merging Lines

This model is solved by simulation because of the passive queues. There are three active queues, one for each of the two types of parts and one representing the merged part. The two passive queues are for synchronizing the two types of parts. The routing contains two sources, one for each type of part. The regenerative method is used for constructing confidence intervals. The sequential stopping rule is used until the mean queueing time of the merged part reaches a specified level of accuracy.

```
MODEL:EX11.4
   METHOD:simulation
   QUEUE:q1
      TYPE:fcfs
      CLASS LIST:l1
         SERVICE TIMES:.5
   QUEUE:q2
      TYPE:fcfs
      CLASS LIST:l2
         SERVICE TIMES:.5
   QUEUE:q3
      TYPE:fcfs
      CLASS LIST:l3
```

```
        SERVICE TIMES:.5
   QUEUE:waitl1
      TYPE:passive
      TOKENS:0
      DSPL:fcfs
      ALLOCATE NODE LIST:w1
         NUMBERS OF TOKENS TO ALLOCATE:1
      DESTROY NODE LIST:d1
      CREATE NODE LIST:c1
         NUMBERS OF TOKENS TO CREATE:1
   QUEUE:waitl2
      TYPE:passive
      TOKENS:0
      DSPL:fcfs
      ALLOCATE NODE LIST:w2
         NUMBERS OF TOKENS TO ALLOCATE:1
      DESTROY NODE LIST:d2
      CREATE NODE LIST:c2
         NUMBERS OF TOKENS TO CREATE:1
   CHAIN:ch1
      TYPE:open
      SOURCE LIST:src1 src2
      ARRIVAL TIMES:1 1
      :src1->l1->c2->w1->d1->l3->sink
      :src2->l2->c1->w2->d2->sink
   CONFIDENCE INTERVAL METHOD:regenerative
   REGENERATION STATE DEFINITION -
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
      QUEUES TO BE CHECKED:q3
         MEASURES:qt
         ALLOWED WIDTHS:10
   SAMPLING PERIOD GUIDELINES -
      CYCLES:200
   LIMIT - CP SECONDS:50
   TRACE:no
END
```

Twenty-eight regeneration cycles occurred during about 50 seconds of solution time. Some of the performance measures related to the passive queues illustrate a lot of variability. Most of the performance measures for the active queues are fairly accurate.

```
RESQ2 VERSION DATE: JANUARY 18, 1984 - TIME: 13:17:41  DATE: 05/26/84
MODEL:EX11.4
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.  29439 DISCARDED EVENTS

                 SIMULATED TIME:     1.8251E+04
                       CPU TIME:          50.30
               NUMBER OF EVENTS:          90735
               NUMBER OF CYCLES:             28
```

```
WHAT:ALLBO

ELEMENT          UTILIZATION
WAITL2           0.00000
Q1               0.50346(0.49644,0.51048)  1.4%
Q2               0.49686(0.49058,0.50314)  1.3%
Q3               0.49702(0.48994,0.50410)  1.4%

ELEMENT          THROUGHPUT
WAITL1           0.99432(0.98691,1.00173)  1.5%
WAITL2           0.99432(0.98691,1.00173)  1.5%
Q1               0.99432(0.98691,1.00173)  1.5%
Q2               0.99432(0.98691,1.00173)  1.5%
Q3               0.99432(0.98691,1.00173)  1.5%
D1               0.99432
C1               0.99432
D2               0.99432
C2               0.99432
SRC1             0.99432
SRC2             0.99432
SINK             1.98865

ELEMENT          MEAN QUEUE LENGTH
WAITL1           19.65564(2.11294,37.19836)  178.5%
WAITL2           10.88694(0.76088,21.01299)  186.0%
Q1               0.99718(0.95723,1.03714) 8.0%
Q2               0.98545(0.94567,1.02523) 8.1%
Q3               0.96349(0.91751,1.00946) 9.5%

ELEMENT          STANDARD DEVIATION OF QUEUE LENGTH
WAITL1           27.18333
WAITL2           15.07572
Q1               1.39796
Q2               1.39158
Q3               1.34793

ELEMENT          MEAN QUEUEING TIME
WAITL1           19.76784(2.16205,37.37363)  178.1%
WAITL2           10.94908(0.74751,21.15063)  186.3%
Q1               1.00288(0.96461,1.04114) 7.6%
Q2               0.99108(0.95440,1.02775) 7.4%
Q3               0.96898(0.92320,1.01476) 9.4%

ELEMENT          STANDARD DEVIATION OF QUEUEING TIME
WAITL1           26.91983
WAITL2           15.17125
Q1               1.00655
Q2               0.98196
Q3               0.94441

ELEMENT          MEAN TOKENS IN USE
WAITL1           0.00000
```

```
WAITL2          0.00000

ELEMENT         MEAN TOTAL TOKENS IN POOL
WAITL1          10.88694(0.76088,21.01299) 186.0%
WAITL2          19.65564(2.11294,37.19836) 178.5%

ELEMENT         MAXIMUM QUEUE LENGTH
WAITL1          114
WAITL2          65
Q1              13
Q2              12
Q3              12

ELEMENT         MAXIMUM QUEUEING TIME
WAITL1          102.37941
WAITL2          78.29776
Q1              9.09138
Q2              7.41888
Q3              7.97958

ELEMENT         OPEN CHAIN POPULATION
CH1             33.48871(24.57817,42.39923) 53.2%

ELEMENT         OPEN CHAIN RESPONSE TIME
CH1             16.83992(12.39086,21.28897) 52.8%
```

11.5. FURTHER READING

The models discussed in Sections 11.1 and 11.2 about tool failures and load balancing are from Oates [129]. The robot model in Section 11.3 is based on one presented in Medeiros and Sadowski [121]. Taylor and Clayton [180] have a more complicated version of the merging lines model described in Section 11.4. Some additional references for models related to manufacturing systems are Cavaille and Dubois [41], Chow, MacNair, and Sauer [47], Engelke, Grotrian, Scheuing, Schmackpfeffer, and Solf [59], Law and Kelton [106], Radloff [138], and Suri [177].

11.6. EXERCISES

11.1 Construct and solve models of manufacturing systems you are familiar with.

11.2 Generalize the model in Section 11.4 to allow for more than one part of each type to be merged into one subassembly.

11.3 Construct models representing various work-in-process schemes.

CHAPTER 12

EPILOGUE

Some theoretical aspects of performance modeling require sophisticated mathematical analysis and techniques. In spite of this, it is our opinion that people with a limited mathematical background can be taught the skills necessary to conduct successful modeling projects. This book addresses some of the topics necessary to develop these skills. We have seen many examples of people with limited mathematical background, both students in classes we have taught and professionals beginning modeling projects, who are capable of learning the necessary skills.

Performance modeling is an art. As such it is a very difficult skill to teach. As Fromm [67] has stated in a different context, performing something which is an art "requires knowledge and effort." There is a great deal to be learned about performance modeling, and it takes a considerable amount of effort to learn it well. In this book we have tried to present the practical aspects of performance modeling. If we want to learn about performance modeling we have to master the theory and the practice. Knowing just the theory is not enough. A lot of practice is required to meld theory and practice to yield intuition that can be used in the art of performance modeling.

We discussed the process of modeling and the formulation of models. We need to understand the system we are trying to model. Without that understanding, it will be very difficult or impossible to model the system accurately. We also have to know the purpose of the model. The purpose will determine the type of model and the level of detail incorporated into the model. Of the many different types of models, we described several. Probability distributions are particularly important in the types of models we focused on, because they allow us to characterize various aspects of the models. Formulating models with parameters allows us to solve the models numerous times by simply substituting different parameter values.

A small set of modeling elements is generally sufficient to represent many different complex systems. We have presented those elements used with the Research Queueing Package. Other modeling packages have similar building blocks. Many packages also have symbols to represent the model elements, which can then be combined to produce a diagram of the model which explicitly shows its behavior. These model diagrams are particularly useful in describing to others how the model works. There are many modeling packages and simulation languages available to aid in the construction and solution of models of extended queueing networks of contention systems. A modeling package simplifies an analyst's job of obtaining performance measures for the system to be modeled.

We discussed two main types of solution techniques. An analytic solution is performed by solving a set of equations relating the input parameters to the performance measures. This method is usually the quickest and most accurate for the model being solved. However, it is applicable only in a limited number of situations. Sometimes when a direct analytic solution does not apply, an approximation technique can be used. Approximation techniques are generally difficult to apply, sometimes because of the complexity of the solution and sometimes because of a lack of bounds on the results.

Simulation is the other solution technique we concentrated on. Simulation is a very general approach to performance modeling, but it has some disadvantages. The randomness found in the results and the long solution time are its two main drawbacks. Confidence intervals allow us to deal intelligently with the randomness, and as computers are becoming faster and equipped with more memory, solution time is also becoming less of a problem. A hybrid approach that combines simulation with an analytic solution can also be attractive in certain cases to reduce the overall solution time.

Structuring a model with submodels aids in clarifying the model, in repeating similar portions of a model, in sharing common submodels between performance analysts, in varying the model structure, and in decomposing a model. As with the hybrid approach, decomposition is a technique which can be used to reduce the total solution time in certain situations. It can also lead to a pure analytic solution of a nonproduct form model.

Of course, accurately interpreting the results of a performance model is crucial to the success of a modeling study. We discussed many different types of performance measures, sources of errors, the accuracy of simulation results, model validation, the level of detail, modification analysis, sensitivity analysis, and plotting of results. Modification analysis is a particularly valuable skill for a performance analyst to develop in order to find the answers to many different "what if" questions.

We looked at many different case studies dealing with four major application areas: everyday life systems, computer systems, communication networks, and manufacturing systems. Very few performance analysts will ever model systems similar to the ones described in Chapter 8 dealing with everyday life systems. However, everyone can understand how the systems work, and the approach to modeling them can be very instructive. The three other application areas are very pertinent in the world of modeling.

There are many places to continue your study of performance modeling. The special issues of the *ACM Computing Surveys 10,* 3 (September 1978) and the *IEEE Computer 13,* 4 (April 1980) are good starting places for queueing network models of computer systems. The following books and papers are also highly recommended: Allen [3], Bruell and Balbo [34], Buzen [36], Crane and Lemoine [56], Ferrari [62], Ferrari, Serazzi and Zeigner [63], Fishman [64, 65], Gelenbe and Mitrani [69], Gordon [70, 71], Heidelberger and Lavenberg [77], Iglehart and Shedler [82], Kleijnen [92, 93], Kleinrock [94, 95, 96], Kobayashi [98], Lavenberg [100], Law and Kelton [106], Lazowska, Zahorjan, Graham, and Sevcik [108], Maisel and Gnugnoli [117], Pritsker [133, 134], Pritsker and Pegden [135], Russell [146], Sauer and Chandy [152], Sauer and MacNair [156], Schriber [164], Schwartz [165], Shannon [171], and Trivedi [183].

Our purpose in writing this book was to present some easy-to-use approaches to modeling complex contention systems. We hope we have added to your knowledge and understanding of performance modeling. It is difficult for people with little or no background in modeling to get started in this field without the help of a course and a good instructor. We have tried to make this task easier.

What does the future hold in store for modeling? There are many more aids which will be made available to assist performance analysts. Model construction can be made simpler by designing modeling languages for specific applications. These types of systems are already available for computer systems, communication networks, and manufacturing systems. Faster, more accurate solution techniques will become available. Better graphics, both for model input and for model results, will make modeling easier. Animation of a simulation model as it is running is a useful debugging tool and can give insight into the behavior of the system operation. Facilities for making model debugging easier would be beneficial. Powerful modeling packages available on personal computers would put modeling aids at the disposal of many more people. The future of performance modeling looks very promising. We hope you share our enthusiasm for the coming improvements.

BIBLIOGRAPHY

1. A.K. Agrawala, S.K. Tripathi, M. Abrams, K.K. Ramakrishnan, M. Singhal, and S.H. Son, "STEP-1: A User Friendly Performance Analysis Tool," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.
2. A.O. Allen, "Elements of Queueing Theory for System Design," *IBM Systems Journal 14,* 2 (1975), 161–87.
3. A.O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications.* New York: Academic Press, 1978.
4. A.O. Allen, "Queueing Models of Computer Systems," *IEEE Computer 13,* 4 (April 1980), 13–24.
5. H.P. Artis, "Capacity Planning for MVS Computer Systems," in *Performance of Computer Installations,* ed. D. Ferrari. Amsterdam: North-Holland, 1978, 25–35.
6. B. Avi-Itzhak and D.P. Heyman, "Approximate Queueing Models for Multiprogramming Computer Systems," *Operations Research 21* (1973), 1212–30.
7. G. Balbo and S. Bruell, "Aggregation in Multiclass Queueing Networks," *Proceedings of the 1981 Computer Measurement Group International Conference,* New Orleans, LA, 1981, 92–96.
8. G. Balbo, A. Marsan, G. Ciardo, and G. Conte, "A Software Tool for the Automatic Analysis of Generalized Stochastic Petri Nets Models," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.
9. J. Banks and J.S. Carson, II, *Discrete-Event System Simulation.* Englewood Cliffs, NJ: Prentice-Hall, 1984.
10. Y. Bard, "An Analytic Model of the VM/370 System," *IBM Journal of Research and Development 22* (1978), 498–508.
11. Y. Bard, "The VM/370 Performance Predictor," *ACM Computing Surveys 10,* 3 (September 1978), 333–342.
12. Y. Bard, "Some Extensions to Multiclass Queueing Network Analysis," in *4th International Symposium on Modeling and Performance Evaluation of Computer Systems,* eds. M. Arato, A. Butrimenko and E. Gelenbe. Amsterdam: North-Holland, 1979.
13. Y. Bard, "A Simple Approach to System Modelling," *Performance Evaluation 1,* 3 (November 1981), 225–48.

14. F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM 22,* 2 (April 1975), 248–60.

15. H. Beilner and J. Mater, "Simulation and Analytic Modelling of Computer System Performance Using the Software Tool COPE," *ECOMA 10 Conference Proceedings,* 1982, 179–83.

16. H. Beilner and J. Mater, "COPE: Past, Presence and Future," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

17. B. Beizer, *Micro-Analysis of Computer System Performance.* New York: Van Nostrand Reinhold, 1978.

18. BGS Systems, Inc., "BEST/1-MVS User's Guide," Waltham, MA, 1983.

19. BGS Systems, Inc., "BEST/1-VM User's Guide," Waltham, MA, 1983.

20. BGS Systems, Inc., "BEST/1-SNA User's Guide," Waltham, MA, 1983.

21. K. Bharath-Kumar and P. Kermani, "Performance Evaluation Tool (PET): An Analysis Tool for Computer Communications Networks," *IEEE Journal on Selected Areas in Communications SAC-2,* 1 (January 1984), 220–25.

22. A. Blum, L. Donatiello, P. Heidelberger, S.S. Lavenberg, and E.A. MacNair, "Experiments with Decomposition of Extended Queueing Network Models," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

23. Boole and Babbage, "Capacity Management Facilities Planning and Use Guide," Sunnyvale, CA, 1983.

24. M. Booyens, P.S. Kritzinger, A. Krzesinski, P. Teunissen, and S. Van Wyk, "SNAP: An Analytic Multiclass Queueing Network Analyzer," Technical Report ITR83-08-00 (September 1983), Institute for Applied Computer Science, University of Stellenbosch.

25. M. Booyens, P.S. Kritzinger, A. Krzesinski, P. Teunissen, and S. van Wyk, "SNAP: An Analytic Multiclass Queueing Network Analyser," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

26. J.W. Boyse and D.R. Warn, "A Straightforward Model for Computer Performance Prediction," *Computing Surveys 7,* 2 (1975), 73–93.

27. A. Brandwajn, "Equivalence and Decomposition Methods with Applications to a Model of a Time-sharing Virtual Memory System," *Proceedings International Symposium Rocquencourt,* 1974, 58–88.

28. A. Brandwajn, "A Model of a Time-Sharing System Solved Using Equivalence and Decomposition Methods," *Acta Informatica 4,* 1 (1974), 11–47.

29. A. Brandwajn, "Issues in Mainframe System Modeling—Lessons from Model Development at Amdahl," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

30. L. Bronner, "Overview of the Capacity Planning Process for Production Data Processing," *IBM Systems Journal 19,* 1 (January 1980), 4–27.

31. L. Bronner, "Capacity Planning Basic Hand Analysis," IBM Technical Bulletin GG22-9344, Gaithersburg, MD, December 1983.

32. R.M. Brown, J.C. Browne, and K.M. Chandy, "Memory Management and Response Time," *Communications of the ACM 20,* 3 (March 1977), 153–65.

33. J.C. Browne, K.M. Chandy, R.M. Brown, T.W. Keller, D. Towsley, and C.W. Dissley, "Hierarchical Techniques for Development of Realistic Models of Complex Computer Systems," *IEEE Proceedings 63,* 6 (June 1975), 966–75.

34. S.C. Bruell and G. Balbo, *Computational Algorithms for Closed Queueing Networks.* New York: Elsevier North-Holland, 1980.

35. S.C. Bruell, G. Balbo, S. Ghanta, and P.V. Afshari, "A Mean Value Analysis Based Package for the Solution of Product Form Queueing Network Models," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

36. J.P. Buzen, *Queueing Network Models of Multiprogramming,* Ph.D. Thesis, Harvard University, Cambridge, MA, 1971. New York: Garland Publishing, 1980.

37. J.P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM 16,* 9 (September 1973), 527–31.

38. J.P. Buzen, "Fundamental Laws of Computer System Performance," *Acta Informatica 7,* 2 (1976), 167–82.

39. J.P. Buzen, "A Queueing Network Model of MVS," *ACM Computing Surveys 10,* 3 (September 1978), 319–32.

40. J.P. Buzen, "Skills for Successful Computer Performance Modeling," *Proceedings of Symposium on Computer Resource Allocation,* Pretoria, South Africa, April 1979.

41. J.-B. Cavaille and D. Dubois, "Heuristic Methods Based on Mean Value Analysis for Flexible Manufacturing Systems Performance Evaluation," *Proceedings 21st IEEE Conference on Decision and Control,* 1982, 1061–65.

42. K.M. Chandy, U. Herzog, and L.S. Woo, "Parametric Analysis of Queueing Networks," *IBM Journal of Research and Development 19,* 1 (January 1975), 43–49.

43. K.M. Chandy, J.H. Howard, Jr., and D.F. Towsley, "Product Form and Local Balance in Queueing Networks," *Journal of the ACM 24,* 2 (April 1977), 250–63.

44. K.M. Chandy and D. Neuse, "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computer Systems," *Communication of the ACM 25,* 2 (February 1982), 126–33.

45. K.M. Chandy and C.H. Sauer, "Approximate Methods for Analyzing Queueing Network Models of Computer Systems," *ACM Computing Surveys 10,* 3 (September 1978), 281–318.

46. W.W. Chiu and W.-M. Chow, "A Performance Model of MVS," *IBM Systems Journal 17,* 4 (1978), 444–62.

47. W.-M. Chow, E.A. MacNair, and C.H. Sauer, "Analysis of Manufacturing Systems by the Research Queueing Package," IBM Research Report RC-10769, Yorktown Heights, NY, October 1984. Also to appear in the *IBM Journal of Research and Development 29,* 4 (July 1985).

48. M. Coome, private communication (1979).

49. J.C. Cooper, "A Capacity Planning Methodology," *IBM Systems Journal 19,* 1 (January 1980), 28–45.

50. P.J. Courtois, "Decomposability, Instabilities and Saturation in Multiprogramming Systems," *Communications of the ACM 18,* 7 (July 1975), 371–77.

51. P.J. Courtois, *Decomposability: Queueing and Computer System Applications.* New York: Academic Press, 1977.

52. P.J. Courtois, "Exact Aggregation in Queueing Networks," *Proceedings First Meeting AFCET-SMF,* Paris, September 1978, 35–51.

53. H. Cramer, *Mathematical Methods of Statistics.* Princeton, NJ: Princeton University Press, 1946.

54. M.A. Crane and D.L. Iglehart, "Simulating Stable Stochastic Systems Part II: Markov Chains," *Journal ACM 21,* 1 (1974), 114–23.

55. M.A. Crane and D.L. Iglehart, "Simulating Stable Stochastic Systems. Part III: Regenerative Processes and Discrete-Event Simulation," *Operations Research 23,* 1 (1975), 33–45.

56. M.A. Crane and A.J. Lemoine, *An Introduction to the Regenerative Method for Simulation Analysis,* Lecture Notes in Control and Information Sciences, Vol. 4. New York: Springer-Verlag, 1977.

57. P.J. Denning and J.P.Buzen, "The Operational Analysis of Queueing Network Models," *ACM Computing Surveys 10,* 3 (September 1978), 225–62.

58. M.E. Drummond, Jr., *Evaluation and Measurement Techniques for Digital Computer Systems,* Englewood Cliffs, NJ: Prentice-Hall, 1973.

59. H. Engelke, J. Grotrian, C. Scheuing, A. Schmackpfeffer, and B. Solf, "Structured Modeling of Manufacturing Processes," *Annual Simulation Symposium,* Tampa, FL, 1983, 55–68.

60. W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. I, 3rd ed.,* New York: Wiley, 1968.

61. W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. II, 2nd ed.,* New York: Wiley, 1971.

62. D. Ferrari, *Computer Systems Performance Evaluation.* Englewood Cliffs, NJ: Prentice-Hall, 1978.

63. D. Ferrari, G. Serazzi, and A. Zeigner, *Measurement and Tuning of Computer Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

64. G.S. Fishman, *Concepts and Methods in Discrete Event Digital Simulation.* New York: Wiley, 1973.

65. G.S. Fishman, *Principles of Discrete Event Simulation.* New York: Wiley, 1978.

66. S. Freireich, private communication, 1983.

67. E. Fromm, *The Art of Loving.* New York: Harper and Row, 1956.

68. K. Garg, "An Approach to Multiprocessor Performance Analysis Using Timed Petrinets Models," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

69. E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems.* New York: Academic Press, 1980.

70. G. Gordon, *The Application of GPSS V to Discrete System Simulation.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

71. G. Gordon, *System Simulation, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 1978.

72. W.J. Gordon and G.F. Newell, "Closed Queueing Networks with Exponential Servers," *Operations Research 15* (1967), 244–65.

73. G.S. Graham, "Guest Editor's Overview: Queueing Network Models of Computer System Performance," *ACM Computing Surveys 10,* 3 (September 1978), 219–24.

74. G.S. Graham, E.D. Lazowska, and K.C. Sevcik, "Components of Software Packages for the Solution of Queueing Network Models," *Proceedings CPEUG '82,* Washington, DC, 1982, 183–87.

75. P. Heidelberger and P.D. Welch, "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations," *Communications of the ACM 24,* 4 (April 1981), 233–45.

76. P. Heidelberger and P.D. Welch, "Simulation Run Length Control in the Presence of an Initial Transient," *Operations Research 31* (1983), 1109–44.

77. P. Heidelberger and S.S. Lavenberg, "Computer Performance Evaluation Methodology," *IEEE Transactions on Computers, Vol.C-33,* 12, (December 1984), 1195–220.

78. H. Hellerman and T.F. Conroy, *Computer System Performance.* New York: McGraw-Hill, 1975.

79. R.V. Hogg and A.T. Craig, *Introduction to Mathematical Statistics,* 3rd ed. New York: Macmillan, 1970.

80. IBM Corporation, "OS/VS2 System Logic Library," Vols. 1–7, SY28-0713 to SY28-0720.

81. D.L. Iglehart, "Regenerative Method for Simulation Analysis," in ed. K.M. Chandy and R.T. Yeh. *Current Trends in Programming Methodology, Volume III: Software Modeling and Its Impact on Performance.* Englewood Cliffs, NJ: Prentice-Hall, 1978, 52–71.

82. D.L. Iglehart and G.S. Shedler, *Regenerative Simulation of Response Times in Networks of Queues.* New York: Springer-Verlag, 1980.

83. Information Research Associates, "CADS—Computer Analysis and Design System: A Brief Description," Austin, TX, 1981.

84. Information Research Associates, "PAWS/A User Guide," Austin, TX, 1983.

85. J. R. Jackson, "Jobshop-Like Queueing Systems," *Management Science 10,* 1 (October 1963), 131–42.

86. I. Kadar, "An On-Board Digital Processing Multibeam Store-and-Forward Node Satellite System," *Proceedings 1980 National Telecommunications Conference,* Houston, TX, December 1980, 70.2.1–70.2.3.

87. I. Kadar, E.A. MacNair, and D.T. Tang, "On-Board Satellite Processing: An Efficient Multibeam Processing Satellite System Via Store-and-Forward Scheduling," IBM SST-GTA Report 79/2.14, Yorktown Heights, NY, November 1981.

88. M.G. Kienzle and K.C. Sevcik, "A Systematic Approach to the Performance Modelling of Computer Systems," *Proceedings 4th International Symposium on Modelling and Performance Evaluation of Computer Systems,* Vienna, 1979, 3–27.

89. M.G. Kienzle and K.C. Sevcik, "Survey of Analytic Queueing Network Models of Computer Systems," *Proceedings ACM SIGMETRICS Conference on Simulation, Measurement and Modeling of Computer Systems,* Boulder, CO, 1979, 113–29.

90. I. Kino and S. Morita, "PERFORMS—A Support System for Computer System Performance Evaluation," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

91. P.J. Kiviat, R. Villanueva, and H. Markowitz, *The SIMSCRIPT II Programming Language.* Englewood Cliffs, NJ: Prentice-Hall, 1969.

92. J.P.C. Kleijnen, *Statistical Techniques in Simulation, Part I.* New York: Dekker, 1974.

93. J.P.C. Kleijnen, *Statistical Techniques in Simulation, Part II.* New York: Dekker, 1975.

94. L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay.* New York: McGraw-Hill, 1964. Reprinted, Dover Publications, 1972.

95. L. Kleinrock, *Queueing Systems. Vol. I: Theory.* New York: Wiley, 1975.

96. L. Kleinrock, *Queueing Systems. Vol. II: Computer Applications.* New York: Wiley, 1976.

97. D.E. Knuth, *The Art of Computer Programming, Vol. 2, Semi-Numerical Algorithms.* Reading, MA: Addison-Wesley, 1969.

98. H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Reading, MA: Addison-Wesley, 1978.

99. S.S. Lam and Y.L. Lien, "A Tree Convolution Algorithm for the Solution of Queueing Networks," *Communications of the ACM 26,* 3 (March 1983), 203–15.

100. S.S. Lavenberg, ed., *Computer Performance Modeling Handbook.* New York: Academic Press, 1983.

101. S.S. Lavenberg and M. Reiser, "Stationary State Probabilities of Arrival Instants for Closed Queueing Networks with Multiple Types of Customers," *Journal of Applied Probability 17* (December 1980), 1048–61.

102. S.S. Lavenberg and C.H. Sauer, "Sequential Stopping Rules for the Regenerative Method of Simulation," *IBM Journal of Research and Development 21,* 6 (1977), 545–56.

103. S.S. Lavenberg and D.R. Slutz, "Introduction to Regenerative Simulation," *IBM Journal of Research and Development 19,* 5 (1975), 458–62.

104. A.M. Law, "Statistical Analysis of Simulation Output Data," *Operations Research 31,* 6 (November-December 1983), 983–1029.

105. A.M. Law and J.S. Carson, "A Sequential Procedure for Determining the Length of a Steady-State Simulation," *Operations Research 27,* (1979), 1011–25.

106. A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis.* New York: McGraw-Hill, 1982.

107. E.D. Lazowska, "The Use of Analytic Modelling in System Selection," *Proceedings CMG XI International Conference,* Boston, MA, 1980, 63–69.

108. E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

109. P.A.W. Lewis, A.S. Goodman, and J.M. Miller, "A Pseudo-Random Number Generator for the System/360," *IBM Systems Journal 8,* 2 (1969), 199–220.

110. W.R. Lilegdon and J.J. Talavage, "A MicroNET Application," *Proceedings of the 1983 Winter Simulation Conference,* Arlington, VA, December 1983, 497–506.

111. L. Lipsky and J.D. Church, "Application of a Queueing Network Model for a Computer System," *ACM Computing Surveys 9,* 3 (September 1977), 205–21.

112. J.D.C. Little, "A Proof of the Queueing Formula $L=\lambda W$," *Operations Research 9,* 3 (1961), 383–87.

113. T.L. Lo, "Computer Capacity Planning Using Queueing Network Models," *Proceedings IFIP W.G.7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation,* Toronto, 1980, 145–52.

114. E.A. MacNair and C.H. Sauer, "The Research Queueing Package: A Primer," *Proceedings of SHARE60,* San Francisco, CA, February 1983, 29–37.

115. E.A. MacNair and C.H. Sauer, "The Research Queueing Package: Parametric Solutions and Graphics," IBM Research Report RC–10585, Yorktown Heights, NY, June 1984.

116. J. Maierhofer and H. Schmidt, "Principles of Modeling with BORIS—A Block Oriented Interactive Simulation System," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

117. H. Maisel and G. Gnugnoli, *Simulation of Discrete Stochastic Systems.* Chicago: Science Research Associates, Inc., 1972.

118. J.B. Major, "Processor, I/O Path, and DASD Configuration Capacity," *IBM Systems Journal 20,* 1 (January 1981), 63–85.

119. J. McKenna and D. Mitra, "Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage," *The Bell System Technical Journal 61,* 5 (May–June 1982), 661–83.

120. J. McKenna, D. Mitra, and K.G. Ramakrishnan, "A Class of Closed Markovian Queuing Networks: Integral Representations, Asymptotic Expansions, and Generalizations," *The Bell System Technical Journal 60,* 5 (May–June 1981), 599–641.

121. D.J. Medeiros and R.P. Sadowski, "Simulation of Robotic Manufacturing Cells: A Modular Approach," *Simulation 40,* 1 (January 1983), 3–12.

122. D. Merle, D. Potier and M. Veran, "A Tool for Computer System Performance Analysis," in *Performance of Computer Installations,* ed. D. Ferrari. Amsterdam: North-Holland, 1978, 195–213.

123. I. Mitrani, *Simulation Techniques for Discrete Event Systems.* Cambridge: Cambridge University Press, 1982.

124. A.M. Mood and F.A. Graybill, *Introduction to the Theory of Statistics.* New York: McGraw-Hill, 1963.

125. B. Mueller, "NUMAS: A Tool for the Numerical Modelling of Computer Systems," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.

126. R.R. Muntz, "Queueing Networks: A Critique of the State of the Art and Directions for the Future," *ACM Computing Surveys 10,* 3 (September 1978), 353–60.

127. D. Neuse, K.M. Chandy, J. Misra, and R. Berry, "Simulation Tools in Performance Evaluation," *CPEUG 81* (Computer Performance Evaluation Users Group), San Antonio, TX, November 1981, 331–34.

128. H.C. Nguyen, A. Ockene, R. Revell, and W.J. Skwish, "The Role of Detailed Simulation in Capacity Planning," *IBM Systems Journal 19,* 1 (1980), 81–101.

129. W.J. Oates, "Manufacturing Modeling Using RESQ," *Proceedings of the 1984 Winter Simulation Conference,* Dallas, TX, November 1984, 357–59.

130. E. Parzen, *Modern Probability Theory and Its Applications.* New York: Wiley, 1960.

131. C.D. Pegden, "Introduction to SIMAN," *Proceedings of the 1983 Winter Simulation Conference,* Arlington, VA, December 1983, 231–41.

132. J.L. Peterson, *Petrinet Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

133. A.A.B. Pritsker, *The GASP IV Simulation Language.* New York: Wiley, 1974.

134. A.A.B. Pritsker, *Modeling and Analysis Using Q-GERT Networks.* New York: Wiley, 1977.

135. A.A.B. Pritsker and C.D. Pegden, *Introduction to Simulation and SLAM.* West Lafayette, IN: Systems Publishing, 1979.

136. Performance Systems, Inc., "An Overview of the SCERT II Computer Performance Prediction System," Rockville, MD, 1981.

137. Quantitative System Performance, Inc., "MAP User Guide," Seattle, WA, 1982.

138. R.L. Radloff, "RESQ Manufacturing Modeling System RMMS," IBM internal report, East Fishkill, NY, March 1984.

139. K.G. Ramakrishnan and D. Mitra, "An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks," *Bell System Technical Journal 61* (1982), 2849–72.

140. M. Reiser, "Interactive Modeling of Computer Systems," *IBM Systems Journal 15,* 4 (1976), 309–27.

141. M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," *Journal of the ACM 27,* 2 (April 1980), 313–22.

142. M. Reiser and C.H. Sauer, "Queueing Network Models: Methods of Solution and their Program Implementation," in *Current Trends in Programming Methodology, Vol. III: Software Modeling and Its Impact on Performance,* ed. K.M. Chandy and R.T. Yeh. Englewood Cliffs, NJ: Prentice-Hall, 1978, 115–67.

143. M. Reiser, "Performance Evaluation of Data Communication Systems," IBM Research Report RZ-1092, August 1981.

144. C.A. Rose, "Validation of a Queueing Model with Classes of Customers," *Proceedings International Symposium Computer Performance*

Modeling, Measurement and Evaluation, Cambridge, MA, 1976, 318–25.

145. C.A. Rose, "A Measurement Procedure for Queueing Network Models of Computer Systems," *ACM Computing Surveys 10,* 3 (September 1978), 263–80.

146. E.C. Russell, *Simulation and SIMSCRIPT II.5.* Los Angeles, CA: CACI, Inc., 1976.

147. C.H. Sauer, "Passive Queue Models of Computer Networks," *Computer Networking Symposium,* Gaithersburg, MD, December 1978. IEEE Catalog No. 78CH1400-1.

148. C.H. Sauer, "Computational Algorithms for State-Dependent Queueing Networks," *ACM Transactions on Computer Systems 1,* 1 (February 1983), 67–92.

149. C.H. Sauer, "Approximate Solution of Queueing Networks with Simultaneous Resource Possession," *IBM Journal of Research and Development 25* (1981), 894–903.

150. C.H. Sauer and K.M. Chandy, "Approximate Analysis of Central Server Models," *IBM Journal of Research and Development 19,* 3 (May 1975), 301–13.

151. C.H. Sauer and K.M. Chandy, "Approximate Solution of Queueing Models," *IEEE Computer 13,* 4 (April 1980), 25–32.

152. C.H. Sauer and K.M. Chandy, *Computer Systems Performance Modeling.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

153. C.H. Sauer and E.A. MacNair, "Simultaneous Resource Possession in Queueing Models of Computers," *Performance Evaluation Review 7,* 1 and 2 (1978), 41–52.

154. C.H. Sauer and E.A. MacNair, "Queueing Network Software for Systems Modeling," *Software-Practice and Experience 9,* 5 (May 1979), 369–80.

155. C. H. Sauer and E.A. MacNair, "The Research Queueing Package Version 2: Availability Notice," IBM Research Report RA-144, Yorktown Heights, NY, August 1982.

156. C.H. Sauer and E.A. MacNair, *Simulation of Computer Communication Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

157. C.H. Sauer, E.A. MacNair, and J.F. Kurose, "The Research Queueing Package: Past, Present and Future," *Proceedings 1982 National Computer Conference,* Houston, TX, 1982, 273–80.

158. C.H. Sauer, E.A. MacNair, and J.F. Kurose, "The Research Queueing Package Version 2: Introduction and Examples," IBM Research Report RA-138, Yorktown Heights, NY, April 1982.

159. C.H. Sauer, E.A. MacNair, and J.F. Kurose, "The Research Queueing Package Version 2: CMS Users Guide," IBM Research Report RA-139, Yorktown Heights, NY, April 1982.

160. C.H. Sauer, E.A. MacNair, and J.F. Kurose, "The Research Queueing Package Version 2: TSO Users Guide," IBM Research Report RA-140, Yorktown Heights, NY, April 1982.

161. C.H. Sauer, E.A. MacNair, and J.F. Kurose, "Queueing Network Simulations of Computer Communication," *IEEE Journal on Selected Areas in Communications SAC-2,* 1 (January 1984), 203–19.

162. C.H. Sauer, E.A. MacNair, and S. Salza, "A Language for Extended Queueing Networks," *IBM Journal of Research and Development 24,* 6 (November 1980), 747–55.

163. C.H. Sauer, M. Reiser, and E.A. MacNair, "RESQ - A Package for Solution of Generalized Queueing Networks," *Proceedings 1977 National Computer Conference,* Dallas, TX, 1977, 977–86.

164. T.J. Schriber, *Simulation Using GPSS.* New York: Wiley, 1974.

165. M. Schwartz, *Computer Communication Network Design and Analysis.* Englewood Cliffs, NJ: Prentice-Hall, 1977.

166. M. Schwartz, "Performance Analysis of the SNA Virtual Route Pacing Control," *IEEE Transactions on Communications COM-30,* 1 (January 1982), 172–84.

167. P. Schweitzer, "Approximate Analysis of Multiclass Closed Networks of Queues," *Proceedings International Conference on Stochastic Control and Optimization,* 1979.

168. H.D. Schwetman, "Hybrid Simulation Models of Computer Systems," *Communications of the ACM 21,* 9 (September 1978), 718–23.

169. K.C. Sevcik, G.S. Graham, and J. Zahorjan, "Configuration and Capacity Planning in a Distributed Processing System," *Proceedings 16th CPEUG Meeting,* Orlando, FL, 1980, 165–71.

170. K.C. Sevcik and I. Mitrani, "The Distribution of Queueing Network States at Input and Output Instants," *Journal of the ACM 28,* 2 (April 1981), 358–71.

171. R.E. Shannon, *Systems Simulation: The Art and Science.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

172. J.G. Shanthikumar and R.G. Sargent, "A Unifying View of Hybrid Simulation/Analytic Models and Modeling," *Operations Research 31* (1983), 1030–52.

173. S.W. Sherman, F. Baskett and J.C. Browne, "Trace-driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," *Communications of the ACM 15* (1972), 1063–69.

174. C.E. Skinner, "A Priority Queueing System with Server Walking Time," *Operations Research 15* (1967), 278–85.

175. J. Spragins, "Analytical Queueing Models: Guest Editor's Introduction," *IEEE Computer 13,* 4 (April 1980), 9–11.

176. H.M. Stewart, "Performance Analysis of Complex Communications Systems," *IBM Systems Journal 18,* 3 (1979), 356–73.

177. R. Suri, "New Techniques for Modelling and Control of Flexible Automated Manufacturing Systems," *Proceedings of the IFAC 8th Triennial World Congress,* Kyoto, Japan, 1981, 175–81.
178. R. Suri, "Robustness of Queueing Network Formulas," *Journal of the ACM 30,* (1983), 564–94.
179. L. Svobodova, *Computer Performance Measurement and Evaluation Methods: Analysis and Applications.* New York: Elsevier, 1976.
180. B.W. Taylor and E.R. Clayton, "Simulation of a Production Line System with Machine Breakdowns Using Network Modeling," *Computers and Operations Research 9,* 4 (1982), 255–64.
181. A. Thomasian and B. Nadj, "Aggregation of Stations in Queueing Network Models of Multiprogrammed Computers," *Performance Evaluation Review 10* (1981), 86–96.
182. S.J. Tolopka and H.D. Schwetman, "Mix-Dependent Job Scheduling—An Application of Hybrid Simulation." *1979 National Computer Conference Proceedings, AFIPS Volume 48,* AFIPS Press, 1979, 45–49.
183. K.S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1982.
184. M. Veran and D. Potier, "QNAP2: A Portable Environment for Queueing Systems Modelling," *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis,* Paris, May 1984.
185. J. Voldman, private communication, 1984.
186. P. Wessels, private communication, 1982.
187. W. Whitt, "The Queueing Network Analyzer," *The Bell Technical Journal 62,* 9 (November 1983), 2779–815.
188. W. Whitt, "Performance of the Queueing Network Analyzer," *The Bell Technical Journal 62,* 9 (November 1983), 2817–43.
189. J.W. Wong, "Queueing Network Models of Computer Communication Networks," *ACM Computing Surveys 10,* 3 (September 1978), 343–52.
190. J.W. Wong and G.S. Graham, "A Self-Assessment Procedure Dealing with Queueing Network Models of Computer Systems," *Communications of the ACM 22,* 8 (August 1979), 449–54.
191. J. Zahorjan, K.C. Sevcik, D.L. Eager, and B.I. Galler, "Balanced Job Bound Analysis of Queueing Networks," *Communications of the ACM 25,* 2 (February 1982), 134–41.

INDEX

[illegible]