# CONFIGURATION OF COMPUTING SYSTEMS: AN APPROACH USING QUEUEING NETWORK MODELS

by

Charles Herbert Sauer, B.A.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 1975

CONFIGURATION OF COMPUTING SYSTEMS: AN APPROACH

USING QUEUEING NETWORK MODELS

APPROVED BY SUPERVISORY COMMITTEE:

K. M. Chandy

J. C. Browne

Jerry A. Welch

A. G. Pearson

John H. Howard, Jr.

## ACKNOWLEDGEMENTS

I would like to thank Professor K. M. Chandy for his continued support, advice and encouragement throughout the development and completion of this research and dissertation. I would also like to thank Professors J. C. Browne, J. H. Howard, A. G. Pearson and T. A. Welch for their review and suggestions concerning this dissertation. U. Herzog and L. Woo suggested some of the problems considered here; I am grateful for their advice and encouragement. I must acknowledge my indebtedness to Professor A. G. Pearson and E. H. Pearson for their encouragement throughout my graduate studies and especially for their generous support during the initial stages of my studies. This research was supported in part by National Science Foundation Grant GJ-35109.

C.H.S.

February 1975

CONFIGURATION OF COMPUTING SYSTEMS: AN APPROACH

USING QUEUEING NETWORK MODELS

Publication No. _____

Charles Herbert Sauer, Ph.D.
The University of Texas at Austin, 1975

Supervising Professor: K. Mani Chandy

An approach to configuration design of computing systems is presented. This approach is based on analysis and optimization of queueing network models of computing systems. Efficient optimization of open queueing networks with different classes of customers is considered. Efficient numerical analysis techniques and inexpensive approximate analysis techniques for a large class of central server models are presented. These techniques are suitable for thorough study of a large parameter space of configurations. The central server models considered include non-exponential distributions, different classes of customers and scheduling disciplines with priorities. Simulation analysis of a very general class of queueing networks is discussed. These techniques allow determination of confidence intervals for open, closed and mixed queueing networks with different classes of customers and both passive and active servers. A language for description and analysis of this class of queueing networks is presented.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER PAGE

CHAPTER PAGE

# CHAPTER I

# INTRODUCTION

## 1.1 Design of Computing Systems

Computing facilities are sufficiently complex that we cannot hope to choose an optimal or near-optimal system without thorough analysis of the many configurations available. This work presents tools for analysis of computing systems and a coherent approach to important aspects of choice of a particular configuration of a computing system. This approach utilizes the tools we present along with existing techniques. Though we will primarily consider configuration of new systems the tools and approach we present are also appropriate to reconfiguration of existing systems.

In general we assume that we are given a characterization of the workload for the proposed system and characterizations of the components available for the proposed system. Given these characterizations, we will want to solve one of two problems. Either we will be given constraints on the cost of the system and required to maximize performance of the system without violating the cost constraints, or we will be given performance constraints and required to minimize cost. In the case of existing systems, we may wish to maximize performance without changing the hardware configuration.

## 1.2 Performance Measures

Many performance measures are possible, and the measures used may be dependent on the proposed system. We will assume that the primary measure of interest is the time required by the system to service a user request and respond to the user. This response time measure may be refined in a variety

of ways. We may be interested in only the mean response time, or we may require some other estimates of the distribution of response times. We may wish to differentiate between users on a political basis, an economic basis, a basis of mode of access to the system, such as batch or interactive, or some other basis. Consider as an example a university computation center, with a wide variety of users and applications. The large majority of users make very small requests on the system while a few users have applications which place heavy demands on the system. In order to give good response to the "average user", priority may be given to the users with small requests, otherwise the applications with heavy demands clog the system and cause poor response time for the average user. However, if the small requests are given too high priority, the users with heavy demands may get very poor response. This may not be readily apparent from response time measures which do not distinguish between users. Such a situation will be politically unwise since the users with heavy demands provide a much larger share of the center's support than do the users that are given priority. Some users may be willing to pay more for computational service in order to get priority service, or other users may be willing to suffer poor response in order to pay at a discounted rate. (Service for such users may be scheduled when the system is not in demand such as late at night or on weekends.) For interactive users, distinction may be made according to the kind of interaction. We may wish to give virtually instantaneous response to those using text editors or computer aided instruction, but be willing to accomodate slower response for interactions requiring substantial computation.

A measure of secondary importance is the throughput of customers through the system. Increasing throughput may degrade response time for some users. We wish to maximize throughput without unduly sacrificing response time characteristics. Measures concerned with individual components of the system are also of importance. Measures concerned with components exclusively held by a customer may give indications of the sensitivity of more important measures to fluctuations in the workload. For example, if such a component is almost always in use, then the response time may be very sensitive to temporary increases in the workload. If such a component is little used, it is likely that we can improve response time by increasing the utilization of that component. We will be concerned with utilizations, queue lengths and waiting times for exclusively held components.

We will not consider reliability of the system, though this is clearly an important performance measure.

## 1.3 Workload Characterization

We will assume that the workload has been characterized at a fairly gross level of detail. It is unlikely that the workload can be characterized precisely until the system is operational, and precise characterization may not be possible even then. Notice that since we cannot provide precise characterization of the workload we can provide only estimates of the performance of the system; some error is inevitable and small additional error due to our analysis must be considered acceptable.

The workload characterization will be based on what we know about the use of the proposed system. We must apply this knowledge and measurements from existing systems to characterize the workload (B4,B5,J2).

More specifically, we will assume that the workload is characterized in terms of the arrival rate(s) of user requests (we may distinguish between users as we did for response times) and in terms of the specific nature of the requests. Generally a user request will consist of several cycles, each cycle consisting of a computation part and a data transfer part. During the computation part of the cycle, a processor uses data found in input buffers, if the program has input data, and places results in output buffers. When input buffers become empty or when output buffers are filled the program requests that data be transferred from secondary storage or to secondary storage, respectively. It may be possible that, because of multiple buffering, the program can continue computation while data transfer is taking place. In this case the two parts of the cycle partially or completely overlap.

For the computation part of the cycle we may wish to distinguish between different kinds of programs being executed, for example distinguishing between execution of user programs and execution of different programs provided by the system such as compilers, loaders and text editors. Studies of existing systems (J2) show that these different kinds of programs have markedly different computational characteristics. The two computational characteristics of primary interest are the distribution of the amount of memory needed by a program and the distribution of the length of the computation part of the cycle. Of course the length of the computation will be strongly dependent on the speed of the memory and processor used. We assume that we can make a memory and processor independent characterization of the distribution of the computational period and adjust the distribution with multiplicative factors dependent on a given processor and a given memory. For the data transfer

part of the cycle, we may also wish to distinguish between the programs being executed. In particular, the relative frequency of access to different data files and memory requirements will be dependent on the program which is executing; other characteristics are less likely to exhibit strong program dependence. The data transfer part can be characterized by the number of files and for each file the size, relative access frequency and the distribution of characters transferred per access. In addition to characterizations of the parts of the cycle, we assume that we have a characterization for the distribution of the number of cycles per request. Again, this may be dependent on the kind of request.

## 1.4 System Component Characterization

We assume that the system components have been characterized in terms of capabilities and costs. The components of primary interest are hardware elements such as memory, central (computational) processors and file storage (input/output) devices and the operating system, in particular the schedulers. Software other than the operating system will impact performance and cost, but we will not take this software into consideration.

Memory characteristics of importance are the access times, the transfer rates, the quantity, the unit cost, the organization (we may have several levels of executable memory or a virtual memory system) and the memory scheduler. The important characteristics of the central processing units are the number of units, their speeds, their costs and the scheduler. Usually the scheduler will be a round-robin scheduler which attempts to share the processors among the programs needing a processor. The scheduler may give some programs priority over others. In multiple processor systems

some of the programs may be divided into concurrently executable tasks. The scheduler may assign these tasks to available processors.

For the input/output devices, a system can be characterized by the number of devices, the types of devices (drums, disks, tapes, etc.), the costs, the capacities, the transfer rates, the positioning times, the rotational delays, the schedulers and organizational considerations including channels, controllers and/or peripheral processors. These organizational considerations may be quite complex. Several slow speed devices, such as card readers or line printers, may be connected to a multiplexor channel which supports simultaneous transfer to or from several devices. Disk systems are widely used and have intricate organizations. For example, we may have a situation where positioning of the disk requires a possession of a channel, controller and disk to initiate the positioning, but only the disk for the rest of the positioning operation. All three units are then required for data transfer. Thus overlap of positioning with data transfer is possible where multiple disks are connected to a single controller. The disk scheduler may attempt to minimize positioning and rotational delays in choosing which programs to service.

## 1.5 Difficulty of the Problem

The problem of configuration of computing systems is very complex, and the complexity of the problem will continue to increase. We cannot hope for a complete solution to the problem now or in the near future. We can hope for better understanding of the difficulties involved, and for tools and theory which will provide guidance where our intuition is too weak. Much progress has been made and continues to be made in this area, and this

progress leads to better configurations and improved performance. Our research makes a substantial contribution to this progress.

Out models, in conjunction with models developed by others, can consider most of the workload and system characteristics described above. The primary exception is that we cannot consider general memory hierarchies at the level of detail considered above.

## 1.6 Organization of Chapters

In Chapter II we discuss previous work on modeling of computing systems as queueing networks and analysis of queueing network models, then summarize the tools we have developed and present an approach to configuration design using queueing network models. In Chapters III through VI we discuss our tools in detail.

# CHAPTER II

## SUMMARY OF PREVIOUS WORK, NEW RESULTS AND THE GENERAL DESIGN APPROACH

### 2.1 Previous Work

#### 2.1.1 Queueing Network Models of Computing Systems

Computing systems have become sufficiently complex and varied that we cannot hope to choose among configurations by actually assembling a variety of configurations and comparing their performance. We must have models of computing systems which reflect the possible configurations and indicate the performance to be expected from a given configuration. Further these models should be such that configurations can be easily defined and performance measures can be easily obtained.

Several different authors have proposed models of computing systems as closed networks of queues at central processing units and input/output devices. The earliest work in this area was that of Smith (S5) and Gaver (G1). Baskett (B1) studied the effects of different scheduling disciplines and service distributions in some of these models. Buzen (B6) called these models "central server models" and used central server models in the analysis of system bottlenecks. (See Figure 5.1) Foster (F2) has used central server models along with simulation studies to consider file placement in memory hierarchies.

More recently Brown (B4) has embedded a central server model within a queueing network which includes a queue for memory. He demonstrated that this model could obtain performance measures comparable to those obtained from empirical studies of a general purpose interactive system. Browne et al

(B5) have used extensions of these models to evaluate and improve the performance of a large computing system. The results of (B4,B5) and the approximate analysis techniques discussed below point to the use of approximation in model solution to avoid some of the approximation previously required by model assumptions. If we make strong model assumptions to allow exact solution, then we cannot use the models to study parameters and characteristics ignored in the model. We may not be able to analyze exactly models which consider these parameters and characteristics, but if we can get good approximate analyses, then we will have a basis for evaluating these parameters and characteristics. Though the performance measures obtained are not exact for the given model, the trends and effects predicted by the approximate analysis of these complex models give a picture of the actual system which is impossible to obtain from simpler models. For example, we must ignore scheduling priorities if we wish to apply local balance techniques. (Sec. 2.1.2). If we use a locally balanced model we cannot predict the effects of the priorities, but if we approximately analyze a model which includes priorities, we can predict the effects of different priority schemes. Clearly, more accurate models will give better predictions than models with assumptions that are too strong. As the need for more accurate models becomes apparent, two questions arise: 1) What are the deficiencies in the model and how can they be alleviated?, and 2) How much error is introduced by approximation in the solution? These questions cannot be completely answered at this time. Others have attempted to answer the second question by comparison with exact or simulation results, and we use this approach also. This is not entirely satisfactory since this validation may not expose the areas where the

approximations fail. Further research into error bounds for approximations is needed.

### 2.1.2 Analysis of Queueing Network Models

Though properties of isolated queues have been studied for most of this century, analysis of networks of queues is relatively recent. Jackson (J1) considered analysis of open networks of queues with exponential service time distributions (D1). Informally, a network is considered to be open if customers may arrive and depart, and closed if the same customers remain in the network at all times. Jackson showed that the solutions for these networks have a "product-form". By this we mean that the Markovian state probabilities (D1) can be expressed as a product of terms for each queue in the network. Such a product form does not exist for arbitrary networks. Gordon and Newell (G3) showed that similar closed form solutions exist for closed networks with exponential servers. (See equation 5.1). Though these product form solutions are easily expressed, direct evaluation of the solutions for state probabilities and application of the solutions to evaluation of performance measures is computationally expensive. Buzen (B6) developed efficient computational techniques for obtaining state probabilities and performance measures for closed networks with product form solutions. Chandy (C1) developed the concept of "local balance" and showed that it could be used to obtain product form solutions for a large class of queueing networks including those studied by Jackson and Gordon and Newell. This class also includes some networks with non-exponential service time distributions and some queueing disciplines other than First Come First Served. Baskett, Chandy, Muntz and Palacios (B2) used local balance to obtain product

form solutions for networks with different classes of customers and customer dependent behavior. Chandy, Herzog and Woo (C3) developed techniques for efficient parametric analysis of networks in local balance akin to Norton's Theorem for electrical circuits. They showed that for each queue in a general closed network one can represent the effects of the remainder of the network by a "composite queue", and that the parameters of the queue of interest may be varied without affecting the parameters of the composite queue. See Figures 5.1 and 5.2. A similar result holds for open networks. Reiser and Kobayshi (R1) extended the results of Baskett et al to more general networks and developed computational algorithms for these models. Chandy, Howard and Towsley (C4) developed the concept of "station balance", a sufficient condition for local balance. They showed that for queues in station balance, performance criteria such as utilization, queue length distributions and mean waiting times are independent of the form of the service time distribution, as long as the service time distributions are differentiable.

Closed form solutions have been very difficult to obtain for networks which do not have solutions obtainable by local balance techniques. Wallace and Rosenberg (W1) have applied iterative numerical techniques to solution of queueing networks. Herzog, Woo and Chandy (H1) have developed recursive numerical techniques for solution of queueing networks. These techniques are effective for small networks but require excessive computation for large or complex networks. Several authors including Gaver (G1), Baskett (B1), Shedler (S4) and Curtois and Georges (C8) have applied semi-Markov techniques to small closed networks. A variety of approximation techniques have been applied to solution of more general queueing networks.

Gaver and Shedler (G2) and Kobayashi (K3,K4) have applied diffusion approximations to queueing networks. Decomposition approximation techniques based on local balance techniques and numerical techniques have been used by Chandy, Herzog and Woo (C2), Brown (B4), Keller and Chandy (K1), Sauer and Chandy (S2) and Williams and Bhandiwad (W2). Simulation techniques are still the most general; simulation techniques especially applicable to queueing networks have been developed by Foster, McGehearty, Sauer and Waggoner (F1), Crane and Iglehart (C6,C7) and Lavenberg (L1).

Optimization of queueing networks has been studied by Kleinrock (K2) and Hogarth (H3).

## 2.2 Contributions of this Research

In Chapter III we extend the results of Kleinrock and Hogarth to optimization of a general class of open queueing networks with <u>different classes of customers</u>. Their results were restricted to networks with all customers identical. This class includes networks which may be solved using local balance techniques and some networks with priority queues. We show that the optimization procedure used by Hogarth may be applied to this class of networks.

In Chapter IV we apply the numerical solution techniques of Herzog, Woo and Chandy to several closed queueing networks important in computer system analysis. Their previous work suggested a partitioning scheme for analysis of structured Markovian state spaces, but applied the scheme only to a small group of models. We show that their scheme may be applied to many general models and give specific efficient computational algorithms for solution of these models. These models allow two queue closed networks with general service

time distributions, multiple identical servers, multiple classes of customers and priority queueing disciplines. These models are very important in approximate solution of more general networks. We demonstrate that these models are of interest in themselves with a study of effects of parallelism in central processing units. Multiprocessing is becoming increasingly common in computer architectures and these models are useful in analysis of these architectures. As measurement techniques become more refined, the need for distinguishing between different users and different programs becomes apparent. Different classes of customers are very useful in developing these distinctions. The need for priorities is increasing with the complexity of computing systems. This is especially true in networks of computers, where each computer is responsible for communication of messages between other computers, as well as processing of requests assigned to that computer.

These same considerations are important in models with a more general network structure than that considered in Chapter IV. The solutions of the models of Chapter IV are used in Chapter V to obtain approximate solutions for a general class of central server models. Exact solutions for this class of models are not available. This class includes the characteristics of the models of Chapter IV. Our techniques give exact results for a subset of this class which can be analyzed by local balance; we validate the techniques for general models by comparison with the results of an extensive group of simulations. The techniques of Chandy, Herzog and Woo may also be applied to this class of models, but their techniques require iterative decomposition analysis for each queue in the network. Our techniques provide solutions for the entire model in a single step and thus are much more economical for parametric analysis.

Crane and Iglehart have provided theorteical results showing how the concept of regeneration points can be applied to determine confidence results for simulations. In Chapter VI we show that these techniques may be applied to simulations of general queueing networks and discuss the practical considerations involved. We present a simulation language QUASCI, based on the language of (F1), which may be used to describe a large class of generalized queueing networks. This language is designed to facilitate application of the Crane and Iglehart techniques and prevent incorrect application of their techniques. This language is also well suited for solution packages using non-simulation techniques; we suggest that this language is appropriate for description of general models of computing systems. As the need for simulation studies will be with us for an indefinite period of time, we must develop a sound theoretical basis for simulation study. This theory should help guarantee that we are simulating the model we think we are simulating, and give us indications of the accuracy of our results. The simulator we have constructed and our language, QUASCI, are important early contributions in this area.

## 2.3 Parameterization of Central Server Models

Though the queueing network models we consider are very useful in analysis of computing systems, the problem of determining the parameters of the queueing network is definitely non-trivial. We briefly consider the problem of representing a computing system as a central server model.

We assume for ease of exposition that we are modeling an existing system. We assume that a measurement probe exists within this system and that we have reduced data obtained from this probe.

As we will see in Chapter V, a central server model is characterized by a queue for the central processing unit(s) (CPU) and several queues for input/output (I/O) devices. By device we may mean a storage device proper, a channel, a controller or whatever equipment is most representative of the data transfer process. Each of these queues will be described in terms of a queueing discipline, which we assume for now to be First Come First Served, and a service time distribution. To completely specify the model, we need also determine the number of programs in the system, and the probabilities associated with each I/O queue.

The CPU service time distribution can be characterized by the mean and standard deviation of the time from when a program gains use of the CPU until the program releases the CPU and makes a request for data transfer. Notice that we are assuming no overlap of computation with data transfer; we have already noted that this is not necessarily realistic. If significant overlap does occur, then we must make a separate analysis of the overlap to represent the cycle without overlap. See Towsley (T1) for further consideration of this problem.

Similarly, the I/O service time distributions may be characterized by the mean and standard deviation of the time from when a program obtains possession of the device until the program releases the device. Again, our representation is much simplified and an analysis of I/O subsystems may be necessary. We may have controllers and disk organizations which allow overlap of positioning with data transfer. See Browne et al (B5) for further consideration of this problem.

The probabilities associated with each I/O queue may be determined by measuring the relative frequency of access to the files located on the

device or devices represented by that queue. Finally, we can set the number of programs in the model to the number of programs in memory. Usually this value will be studied parametrically, with a range from a low value to the maximum number of programs which the memory will allow. If such parametric study is not desired, we may use the mean number of programs in memory during the measurement period.

We have assumed fairly complete data are available, but we can use limited data under certain circumstances. For example, if we only know the fraction of time the CPU is utilized, the length of the measurement period, and the number of CPU services during the measurement period, we can estimate the mean service time by the time the CPU is utilized divided by the number of requests. We can estimate the form of the distribution from measurements on other systems with similar applications.

Another area for caution, with any models we might choose, is that our measurements may be strongly dependent on the time and day they are made.

## 2.4 An Approach to Configuration of Computing Systems

We propose that the techniques presented here, augmented with the previously developed techniques mentioned above, provide a basis for a general approach to selection of computing system configurations. The specific approach used for a specific system will be strongly dependent on the intended application of the system, the components available and the budgetary or performance constraints.

As a first estimate of the configuration, we can use modified central server models as in Figure 3.2 to study the appropriate choice of central processing units and input/output devices. We assume memory

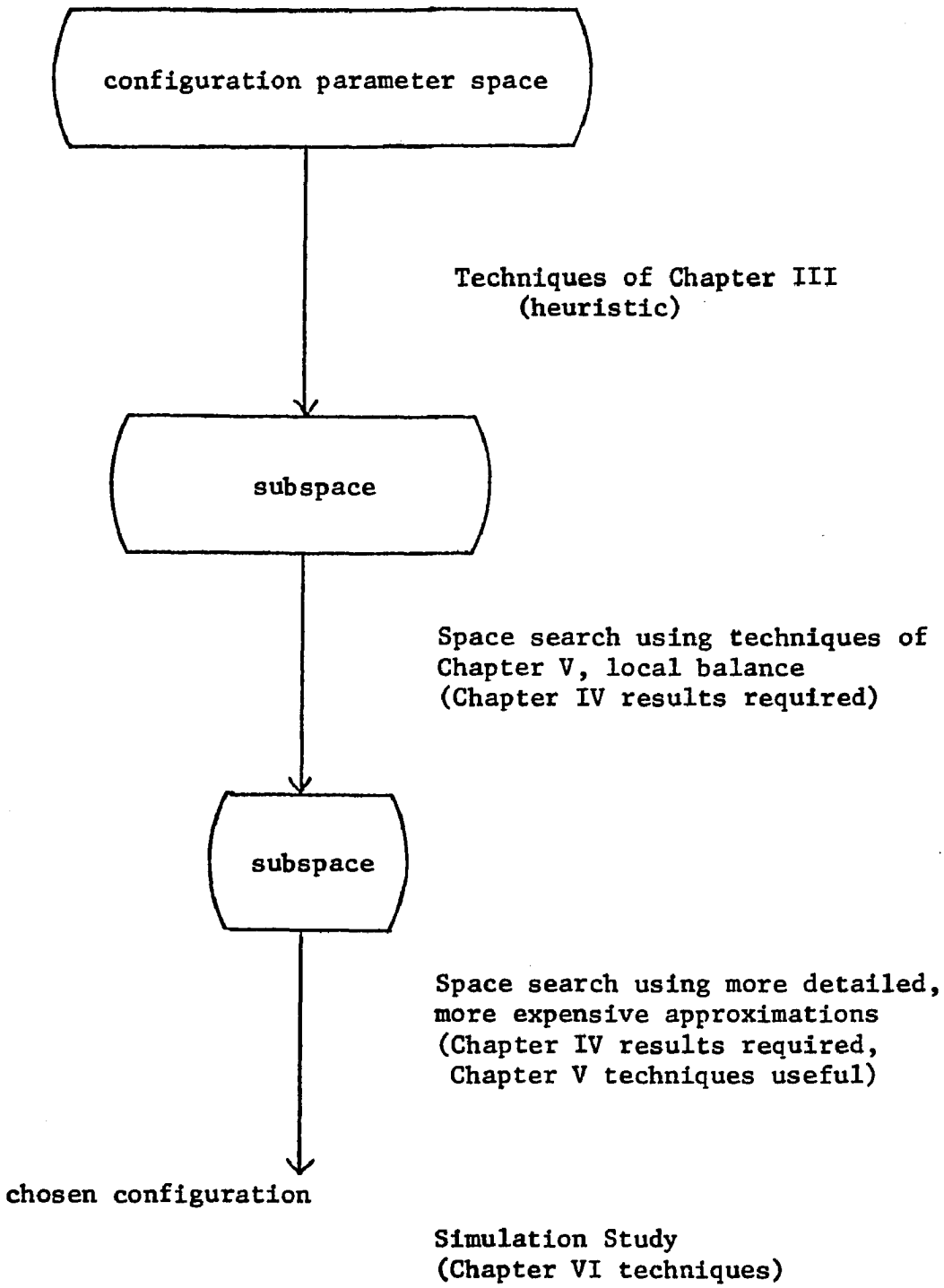
limitations will not significantly affect system performance when we use these models. The techniques of Chapter III can be used to determine the optimum configuration under these assumptions.

Having obtained a first estimate of the appropriate configuration, we can use more refined models similar to the first estimate and study a parameter space of these models using the techniques of Chapters IV and V, of Brown (B4) and of Keller and Chandy (K1). Since the techniques of Chapters IV and V are very inexpensive, we can afford to explore the parameter space thoroughly. Further, our techniques are compatible with those of Brown and of Keller and Chandy; it should be possible to combine techniques and study a parameter space of realistic models without prohibitive computational costs.

Finally, we can assure ourselves that the approximate results are valid by checking our results with the more expensive iterative approximations of Chandy, Herzog and Woo (C2) and then obtaining simulation results using the techniques of Chapter VI. See Figure 2.1.

The language QUASCI provides a convenient vehicle for this entire process, since it is compatible with non-simulation solution techniques. It should be possible to implement this language so that the implementation can have all of these techniques available and choose the solution technique appropriate to the model or models specified. We should be able to augment and implement QUASCI so that much of the parameter space searching process is automated.

Provided a general approach and framework, the computer system designer can turn attention to the problems of characterizing the workload



of the system, of determining what components are available to handle this workload and of developing appropriate models to analyze the possible configurations.

# CHAPTER III

## OPTIMIZATION OF QUEUEING NETWORKS WITH DIFFERENT CLASSES OF CUSTOMERS

### 3.1 Introduction

Queueing network models are useful in the design of communication networks (K2), computer networks (P1) and computing systems (B4,B6). It is desirable to find optimal configurations of these models so that we may attempt to find a near optimal configuration of the system being modeled. Kleinrock (K2) has considered optimal design of a class of open queueing networks with all customers having identical behavior. An open queueing network is one in which customers arrive and depart from the network, as opposed to a closed network in which the number of customers is constant. Hogarth (H3) has extended the results of Kleinrock and has also considered optimal design of closed networks with all customers having identical behavior.

We will consider optimization of open queueing networks with different classes of customers. Specifically, we consider minimization of the time customers spend in the network. We allow a variety of queueing disciplines at each server, and a general class of service time distributions.

In Section 3.2 we show convexity of waiting time at individual queues with Poisson arrivals as a function of processing rate at that queue. Convexity is important in efficiently solving optimization problems. In Section 3.3 we show convexity of total time spent in an open network with Poisson arrivals as a function of processing rates of the queues in the network. Section 3.4 considers the optimization problem statements and algorithms for

solution, and Section 3.5 discusses application of these models.

## 3.2 Convexity of Waiting Times at Individual Queues

The set of points S in n-dimensional Euclidean space is said to be convex if for any points x and y in S and any p in the interval [0,1], $px + (1-p)y$ is in S. A function f defined on the convex set S is said to be convex if and only if for any x and y in S and for any p in the interval [0,1],

$$f(px + (1-p)y) \leq pf(x) + (1-p)f(y) \tag{3.1}$$

This is one of several equivalent definitions (W3). We will find another characterization more useful. If the Hessian of f is defined and positive semi-definite for all x in S, then f is convex (W3). (The Hessian of f is the matrix of order n with i,jth element $\partial^2 f(x)/\partial\zeta_i\partial\zeta_j$, where $\zeta_i$ is the ith element of x.)

We will consider six different queueing disciplines; First-Come-First-Served (FCFS), Infinite Servers (IS), Processor Sharing (PS), Last-Come-First-Served-Preemptive-Resume (LCFSPR), Preemptive Priority, and Non-preemptive Priority. (PS is defined as the limiting case of a no overhead round robin discipline as the quantum goes to zero.) We allow arbitrary class dependent service time distributions with rational Laplace transforms and assume Poisson arrival processes with class dependent arrival rates. These results apply to other queueing disciplines.

Let us consider a single server queue with R classes of customers. Customers of class r, r = 1,...,R, arrive in a Poisson manner with rate $\lambda_r$ and request service with mean $\bar{s}_r$. We define $\lambda = \sum_{r=1}^{R} \lambda_r$ as the total arrival

rate, and define $\bar{s} = \sum_{r=1}^{R} \frac{\lambda_r}{\lambda} \bar{s}_r$ as the mean service time at the queue. We let $\rho = \lambda s$ and $\rho_r = \lambda_r \bar{s}_r$ be the overall utilization, and the utilization for class r customers, respectively. We assume the queue is not saturated, i.e., $\rho < 1$. We define $\mu = \frac{1}{\bar{s}}$ as the processing rate, and $\mu_r = \frac{\bar{s}}{\bar{s}_r}$ as the relative processing constant for class r. Customers of class r are processed with mean rate $\mu\mu_r$. Notice that $\mu_r$ is dimensionless; we assume that if we change the rate of the server and thus change $\mu$, $\mu_r$ is unaffected.

For FCFS queueing discipline, we know from the Khintchine-Pollaczek formula (M1) that the expected time until a customer begins service is

$$\frac{\rho\bar{s}(1+C^2)}{2(1-\rho)} \tag{3.2}$$

where C is the coefficient of variation of the service times (the standard deviation divided by the mean). In our case,

$$C = \frac{\sqrt{\sum_{r=1}^{R} \frac{\lambda_r}{\lambda} \bar{s}_r^2 - \bar{s}^2}}{\bar{s}} \tag{3.3}$$

where $\bar{s}_r^2$ is the second moment of the service time distribution for class r. We assume that C is a constant independent of the rate of the server.

The mean wait time for class r customers, $\bar{w}_r$, is the sum of (3.2) and the mean service time for class r. So we have

$$\begin{aligned} \bar{w}_r &= \frac{\rho\bar{s}(1+C^2)}{2(1-\rho)} + \bar{s}_r \\ &= \frac{\lambda\bar{s}^2(1+C^2)}{2(1-\lambda\bar{s})} + \bar{s}_r \\ &= \frac{\lambda(1+C^2)}{2\mu(\mu-\lambda)} + \frac{1}{\mu_r\mu} \end{aligned} \tag{3.4}$$

We want to show that $\bar{w}_r$ is a convex function of $\mu$. In this case we have a single dimensional vector space and the only element of the Hessian of $\bar{w}_r$ is $\frac{d^2\bar{w}_r}{d\mu^2}$. Differentiating (3.4) we obtain

$$\frac{d\bar{w}_r}{d\mu} = \frac{-\lambda(1+C^2)(2\mu-\lambda)}{2\mu^2(\mu-\lambda)^2} + \frac{-1}{\mu_r\mu^2} \tag{3.5}$$

and differentiating again we find

$$\frac{d^2\bar{w}_r}{d\mu^2} = \frac{\lambda(1+C^2)((\mu-\lambda)^2 + \mu(2\mu-\lambda))}{\mu^3(\mu-\lambda)^3} + \frac{2}{\mu_r\mu^3} \tag{3.6}$$

Since $\rho < 1$, $\lambda < \frac{1}{s} = \mu$ and (3.6) is positive. Thus the Hessian is positive semi-definite and $\bar{w}_r$ is convex.

If we have an infinite number of servers, each with rate $\mu$, then

$$\bar{w}_r = \bar{s}_r = \frac{1}{\mu_r\mu} \tag{3.7}$$

which is clearly convex.

If the queueing discipline is PS or LCFSPR, then we know (R1) that the mean queue length for class r, $\bar{q}_r$, is

$$\bar{q}_r = \frac{\rho_r}{(1-\rho)} \tag{3.8}$$

which we may rewrite as

$$\bar{q}_r = \frac{\rho_r\rho}{(1-\rho)} + \rho_r \tag{3.9}$$

Applying Little's Rule (L3), we have

$$\bar{w}_r = \frac{\bar{q}_r}{\lambda_r}$$

$$= \frac{\bar{s}\rho}{1-\rho} + \bar{s}_r \tag{3.10}$$

This is equivalent to $\bar{w}_r$ for the FCFS case with C = 1; thus $\bar{w}_r$ is again convex.

Now we consider queues with priority queueing disciplines, with priority based on customer class and highest priority given to the classes with lowest index. It is well known (M1) that for preemptive priority the waiting time is

$$\bar{w}_r = \frac{\sum_{i=1}^{r} \lambda_i \frac{(1 + C_i^2)}{(\mu\mu_i)^2}}{2(1 - \sum_{i=1}^{r-1} \rho_i)(1 - \sum_{i=1}^{r} \rho_i)} + \frac{1}{\mu\mu_r(1 - \sum_{i=1}^{r-1} \rho_i)} \tag{3.11}$$

Here $C_i$ is the coefficient of variation of the class i service time. Letting $\rho_i' = \mu\rho_i = \frac{\lambda_i}{\mu_i}$, we rewrite (3.11) as

$$\bar{w}_r = \frac{\sum_{i=1}^{r} \frac{\lambda_i(1 + C_i^2)}{\mu_i^2}}{2(\mu - \sum_{i=1}^{r-1} \rho_i')(\mu - \sum_{i=1}^{r} \rho_i')} + \frac{1}{\mu_r(\mu - \sum_{i=1}^{r-1} \rho_i')} \tag{3.12}$$

Taking derivatives, we get

$$\frac{d\bar{w}_r}{d\mu} = \frac{-(\sum_{i=1}^{r} \frac{\lambda_i(1+C_i^2)}{\mu_i^2})(2\mu - 2\sum_{i=1}^{r-1} \rho_i' - \rho_r')}{2(\mu - \sum_{i=1}^{r-1} \rho_i')^2 (\mu - \sum_{i=1}^{r} \rho_i')^2} + \frac{-1}{\mu_r(\mu - \sum_{i=1}^{r-1} \rho_i')^2} \tag{3.13}$$

and

$$\frac{d^2\bar{w}_r}{d\mu^2} = \frac{(\sum_{i=1}^{r} \frac{\lambda_i(1+C_i^2)}{\mu_i^2})((\mu - \sum_{i=1}^{r-1} \rho_i')^2 + (\mu - \sum_{i=1}^{r-1} \rho_i')(\mu - \sum_{i=1}^{r} \rho_i') + (\mu - \sum_{i=1}^{r} \rho_i')^2)}{(\mu - \sum_{i=1}^{r-1} \rho_i')^3 (\mu - \sum_{i=1}^{r} \rho_i')^3}$$

$$+ \frac{2}{\mu_r(\mu - \sum_{i=1}^{r} \rho_i')^3} \tag{3.14}$$

Since $\mu > \lambda' \geq \sum_{i=1}^{r} \rho_i'$, the value of (3.14) is positive and $\bar{w}_r$ is a convex function of $\mu$.

Again from (M1) we know that the waiting time for non-preemptive priority is

$$\bar{w}_r = \frac{\sum_{i=1}^{R} \frac{\lambda_i(1 + C_i^2)}{(\mu \cdot \mu_i)^2}}{2(1 - \sum_{i=1}^{r-1} \rho_i')(1 - \sum_{i=1}^{r} \rho_i')} + \frac{1}{\mu \cdot \mu_r} \tag{3.15}$$

which we rewrite as

$$\bar{w}_r = \frac{\sum_{i=1}^{R} \frac{\lambda_i(1 + C_i^2)}{\mu_i^2}}{2(\mu - \sum_{i=1}^{r-1} \rho_i')(\mu - \sum_{i=1}^{r} \rho_i')} + \frac{1}{\mu \cdot \mu_r} \tag{3.16}$$

$$\frac{d\bar{w}_r}{d\mu} = \frac{-(\sum_{i=1}^{R} \frac{\lambda_i(1+C_i^2)}{\mu_i^2})(2\mu - 2\sum_{i=1}^{r-1} \rho_i' - \rho_r')}{2(\mu - \sum_{i=1}^{r-1} \rho_i')^2(\mu - \sum_{i=1}^{r} \rho_i')^2} + \frac{-1}{\mu^2 \mu_r} \tag{3.17}$$

$$\frac{d^2\bar{w}_r}{d\mu^2} = \frac{(\sum_{i=1}^{R} \frac{\lambda_i(1+C_i^2)}{\mu_i^2})((\mu - \sum_{i=1}^{r-1} \rho_i')^2 + (\mu - \sum_{i=1}^{r-1} \rho_i')(\mu - \sum_{i=1}^{r} \rho_i') + (\mu - \sum_{i=1}^{r} \rho_i')^2)}{(\mu - \sum_{i=1}^{r-1} \rho_i')^3 \ (\mu - \sum_{i=1}^{r} \rho_i')^3}$$

$$+ \frac{2}{\mu^3 \mu_r} \tag{3.18}$$

As before we see that $\bar{w}_r$ is a convex function of $\mu$.

We have shown for all six disciplines that the waiting times by class are convex functions of the processing rate. We are also interested in

the waiting time independent of customer class. Again from well known results (M1) we know that the expected waiting time, $\bar{w}$, is

$$\bar{w} = \sum_{i=1}^{R} \frac{\lambda_i}{\lambda} \bar{w}_i \tag{3.19}$$

where $\lambda = \sum_{i=1}^{R} \lambda_i$. Since a linear combination of convex functions is also a convex function (W3), $\bar{w}$ is a convex function of $\mu$.

## 3.3 Convexity of Times in the Network

We can consider networks of queues of the class described above. Let N be the number of queues in the network. Class r customers leaving queue n arrive at queue n' in class r' with probability $p_{(nr),(n'r')}$, n'=1,...,N, r'=1,...,R. Otherwise the customer leaves the network.

Those customers of class r, r=1,...,R, which depart queue n, n=1,...,N, in a non-Poisson manner must leave the network when they leave the queue. It is well known that the departure processes of customers leaving FCFS queues having class independent exponential service times are Poisson, as are the departure processes of class 1 customers leaving a queue with preemptive priority and exponential service times for class 1 customers. From (M5) we know that the departure processes of customers leaving IS, PS or LCFSPR queues are Poisson. Departure processes of other queues and classes of customers we have considered are not necessarily Poisson. We assume that all customers arrive from a Poisson source with rate $\lambda$, and that they arrive at queue n in class r with probability $p_{nr}$.

In addition to these restrictions, we must make restrictions on feedback of customers in the network. We allow two kinds of subnetworks,

which we call "Poisson Feed Forward" and "Product Form", and certain interconnections of these networks. The Poisson Feed Forward subnetworks are arbitrary networks as described above which do not have feedback. Figure 3.1 is an example of such a subnetwork. Product Form subnetworks may have only FCFS queues with class independent exponential service times, IS queues, PS queues and LCFSPR queues. We may allow arbitrary feedback of customers in Product Form subnetworks. We call these "Product Form" subnetworks because the solutions for these subnetworks have a product form (B2). Figure 3.2 is an example of such a subnetwork. Though the arrival processes of customers at queues in Product Form subnetworks may not be Poisson, we know from (C3) that the waiting times of customers in such subnetworks will be the same as if the arrival processes were Poisson. Further, we know from (M5) that the combined departure process of each class of customers from a Product Form subnetwork is Poisson. We may allow the combined output of one or more classes of customers from a Product Form subnetwork to feed either kind of subnetwork as long as there is no feedback of customers except within Product Form subnetworks. We may allow any Poisson outputs from Poisson Feed Forward networks to feed either kind of subnetwork as long as there is no feedback of customers except within Product Form subnetworks. For example, we can allow a network consisting of the networks of Figs. 3.1 and 3.2 as subnetworks where the output of the network of Figure 3.2 feeds the network of Figure 3.1.

We are interested in the expected number, $e_{nr}$, of times a queue n is visited by customers of class r. Following (R1), $e_{nr}$ is defined by NR linear equations of the form

$$e_{nr} = p_{nr} + \sum_{n'=1}^{N} \sum_{r'=1}^{R} e_{n'r'} p_{(n'r'),(nr)} \tag{3.20}$$

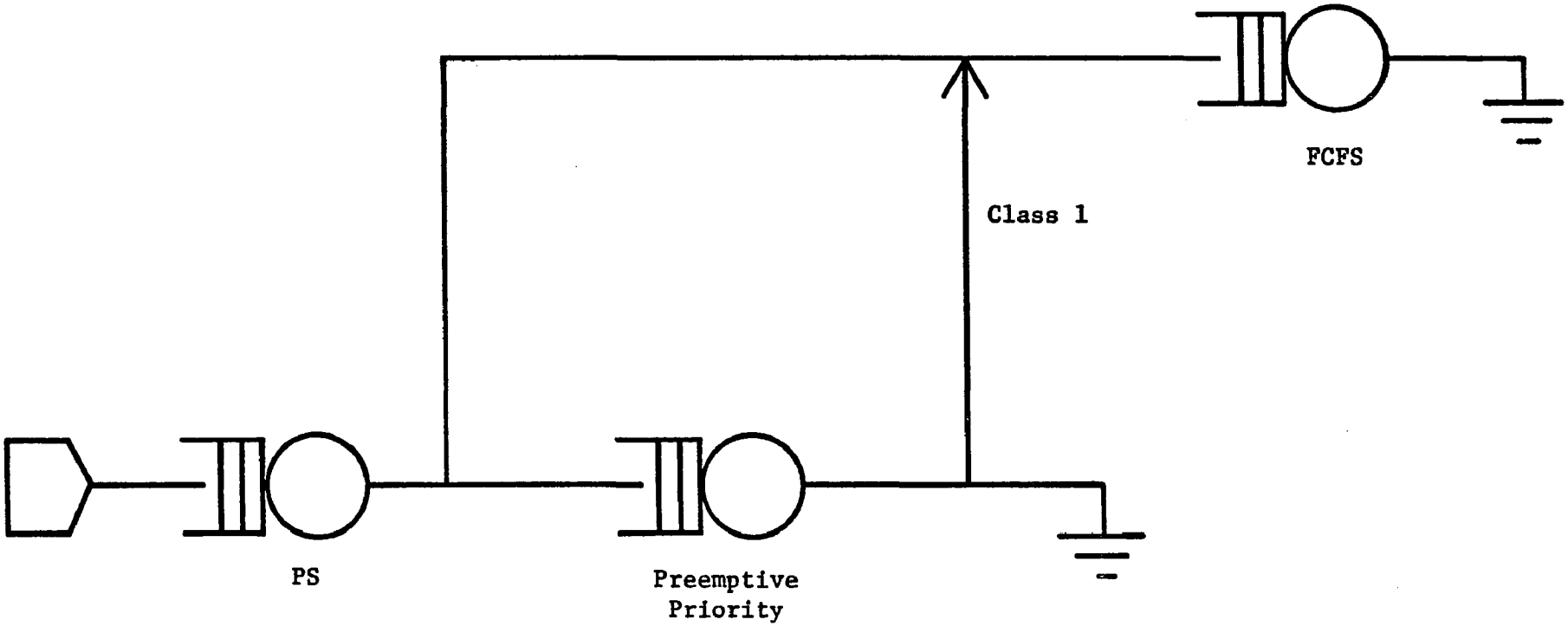



Figure 3.1

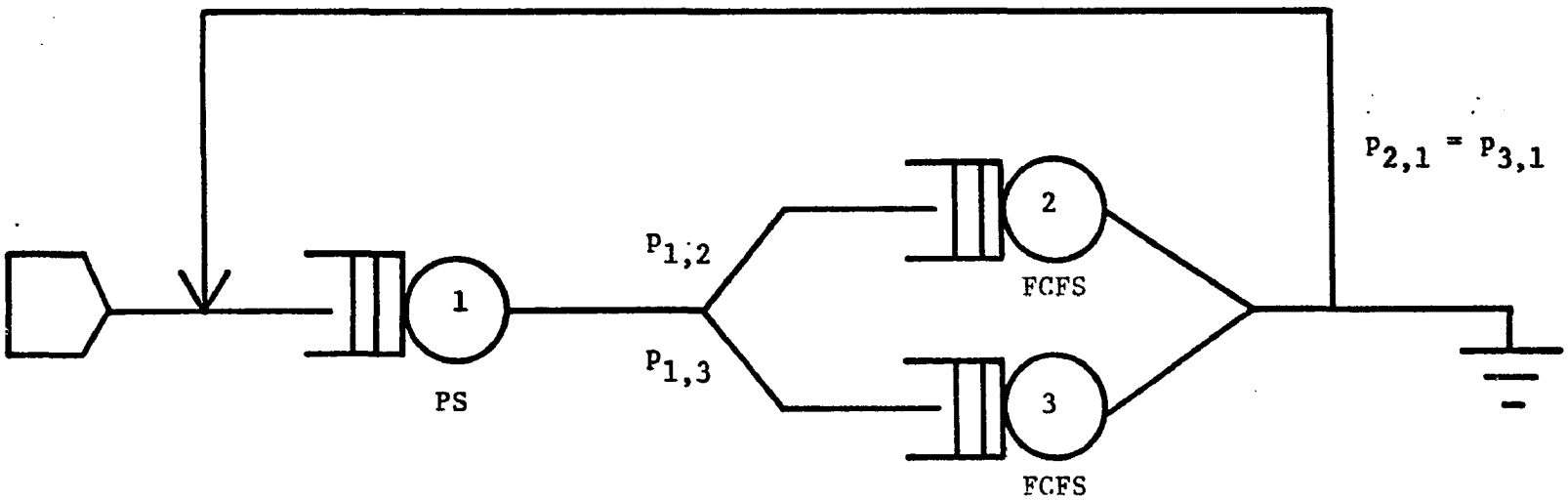



Figure 3.2

The effective arrival rate of customers of class r at queue n is $\lambda e_{nr}$.

As an example, consider the network in Figure 3.2. For simplicity, we assume a single class of customers and omit subscripts indicating customer class. The probabilities associated with paths not shown are assumed to be zero. The equations defined by (3.20) are

$$e_1 = 1 + e_2 p_{2,1} + e_3 p_{3,1}$$

$$e_2 = e_1 p_{1,2}$$

$$e_3 = e_1 p_{1,3}$$

Solving these equations we determine that

$$e_1 = 1/(1 - p_{1,2}p_{2,1} - p_{1,3}p_{3,1})$$

$$e_2 = p_{1,2}/(1 - p_{1,2}p_{2,1} - p_{1,3}p_{3,1})$$

$$e_3 = p_{1,3}/(1 - p_{1,2}p_{2,1} - p_{1,3}p_{3,1}).$$

Clearly the expected total waiting time of a customer at a queue is the expected number of visits multiplied by the expected waiting time per visit, and the total time a customer spends in the network is the sum of the times spent in each queue. So we have

$$\bar{W}_r = \sum_{n=1}^{N} e_{nr}\bar{w}_{nr} \tag{3.21}$$

where $\bar{W}_r$ is the expected total time a customer spends in the network in class r, and $\bar{w}_{nr}$ is the waiting time for class r customers at queue n. We let $\mu_n$ be the overall processing rate at queue n, and let $\mu_{nr}$ be the relative processing rate for class r customers at queue n. We let m be the vector $(\mu_1,\mu_2,\ldots,\mu_N)^T$. We wish to show that $\bar{W}_r$ is a convex function of m. Consider the Hessian of $\bar{W}_r$. For n = 1,...,N

$$\frac{\partial \bar{W}_r}{\partial \mu_n} = e_{nr}\frac{\partial \bar{w}_{nr}}{\partial \mu_n} \tag{3.22}$$

$$\frac{\partial^2 \bar{W}_r}{\partial \mu_n^2} = e_{nr} \frac{\partial^2 \bar{w}_{nr}}{\partial \mu_n^2} \geq 0 \tag{3.23}$$

$$\frac{\partial^2 \bar{w}_r}{\partial \mu_n \partial \mu_j} = 0 , \quad j \neq n \tag{3.24}$$

(The inequality of (3.23) follows immediately from the results of Section 3.2). So the Hessian of $\bar{W}_r$ is positive semi-definite and $\bar{W}_r$ is a convex function of m.

We have treated the general case where customers may change class when leaving a queue. The restricted case where customers do not change class is also of interest. In this case we are interested in the time a class r customer spends in the network, $\bar{W}'_r$. We have

$$\bar{W}'_r = \frac{\bar{W}_r}{\sum_{n=1}^{N} p_{nr}} \tag{3.25}$$

which is clearly a convex function of m since $\bar{W}_r$ is a convex function of m.

Finally, we wish to show that the expected time a customer spends in the network, $\bar{W}$, is a convex function of m. But

$$\bar{W} = \sum_{r=1}^{R} \sum_{n=1}^{N} e_{nr} \bar{w}_{nr} = \sum_{r=1}^{R} \bar{W}_r \tag{3.26}$$

So $\bar{W}$ is a linear combination of convex functions and also a convex function.

## 3.4 Optimization Problem Statements, Procedures

### 3.4.1 Cost Functions

Having shown convexity of the various wait times, we are now ready to consider optimization problem statements and suggest procedures for

solution of the problems. The problems we formulate are instances of general problems which have been previously solved, so we will not present the procedures themselves, but will refer the reader to previous work.

We can consider a variety of optimization problems, depending on the cost functions involved. We restrict attention to a class of continuous and discrete cost functions based on Grosch's Law as discussed by Bell and Newell (B3). They suggest that the function

$$C_n = k_n \sqrt[g]{\mu_n} \tag{3.27}$$

is a good approximation to the cost of a device for queue n, where $C_n$ is the approximate cost, $k_n$ is a constant associated with queue n, and g is a constant. It has been suggested (B3) that in most cases g is in the interval [.5,2], and usually in the interval (1,2). We will assume that g is the same for all queues in the system, and consider two cases. In Section 3.4.2 we consider the case where g is not greater than one and in Section 3.4.3 we consider the case where g is greater than one. In both sections we consider both continuous and then discrete subcases.

### 3.4.2 Convex Cost Functions ($g \leq 1$)

When $g \leq 1$, it is easy to show that $C_n$ is a convex function of $\mu_n$, and if we assume that the total cost, C, is the sum of the costs at the individual queues, then C is a convex function of m. So we have R+2 convex functions, $\overline{W}$, $\overline{W}_r$, r = 1,...,R, and C, defined over the convex set of processing rates, M. $(M = \{m \mid \forall n\ \mu_n > \lambda \sum_{r=1}^{R} e_{nr}\})$.

We now can consider R+2 optimization problems, each one corresponding to minimization of one of the functions $\overline{W}$, $\overline{W}_r$, r = 1,...,R, or C over a subset

of M. This subset consists of the points in M such that the functions not being minimized do not exceed some corresponding maximal values. For example, one such problem would be:

$$\text{minimize } \overline{W} \text{ over the set M}$$

$$\text{such that } \overline{W}_r \leq \omega_r,\ r = 1,\ldots,R$$

$$\text{and } C \leq \gamma.$$

The subset of M constrained in this manner is a convex set (R2). This subset is clearly bounded, and so if a feasible solution exists a local minimum of the objective function ($\overline{W}$ in the example) is also a global minimum. Thus we can use standard procedures (H3,W3) for solution of these optimization problems.

These continuous problems are not realistic representations of many actual systems. Often the choices of processing rates of devices are not continuous, but discrete. For example, in computer systems one usually has a choice of a few central processing units (CPUs) of different rates, not a continuum of choices. Thus the set M in these cases is not convex. However, we may state the optimization problems in the same manner as before. Further, most of the techniques for solving the discrete problem, such as branch and bound techniques (H4), require solution of the continuous problem. So our results are important in the solution of discrete problems.

### 3.4.3 Concave Cost Functions ($g > 1$)

When $g > 1$, the cost function C as defined above is not convex, but concave. (A function f is said to be concave if -f is convex). We may state the problems of minimizing the times in the network as before, but we no longer have the property that a local minimum is necessarily a global

minimum. This complicates the solution techniques considerably, but existing techniques (H3) may be applied and our results are necessary for the application of these techniques. We may solve the continuous forms of these problems using the iterative linear approximation techniques of Hogarth (H3). To apply these linear approximation techniques, we must be able to solve the problems of Section 3.4.2 with linear cost functions (g=1). Again, we will often need to consider discrete cost functions; the solutions of the continuous problems may be used as input to a branch and bound algorithm.

## 3.5 Application of Open Queueing Network Models

Open queueing network models such as these have been applied to design of communications networks, computer networks and computing systems. Since general results for queueing networks with different classes of customers have only recently been obtained, the models used have usually assumed that all customers have identical behavior.

In modeling communication networks (K2) the service devices considered are the transmission lines. Customers have messages to be transmitted and lines of different capacities transmit the messages at different rates. The branching probabilities may be used to represent different routings of messages. Different customers may require different routings, and we may represent this by using customer classes.

Queueing network models have found wide application in modeling computing systems since the early work of Smith (S5), Gaver (G1) and Buzen (B6). The model in Figure 3.2 is very similar to the central server models used by Buzen. Here queue 1 represents the CPU and queues 2 and 3 represent input/output (I/O) devices. Programs arrive at the system, alternately

receive service from the CPU and an I/O device, and eventually leave the system. Typically the CPU will have a round robin scheduling discipline which we may represent by the PS queueing discipline. Different programs often have markedly different CPU request distributions (J2), and we can represent this by class dependent service times. We let the queueing discipline at the I/O's be FCFS with class independent exponential service times, and let the branching probabilities be class dependent. These assumptions are also based on empirical studies (J2). This model assumes that contention for memory is not a factor in the system, which may or may not be correct, depending on the individual system. Similar models (B4,D6, Chapters V, VI) can include memory contention; it would be desirable to extend the results of this chapter to such models. (Some of Hogarth's results (H3) are applicable to some such models with a single class of customers; similar results for multiple-class models and other more general models would be very useful.)

Models similar to these have also been applied to networks of computers (P1); these applications combine the computer system models and communication network models described above.

# CHAPTER IV

## EFFICIENT NUMERICAL SOLUTION OF QUEUEING NETWORKS

### 4.1 Introduction

Queueing network models are used to effectively model the performance of computing systems (B1,B4,B6,F2,G1,S4). Closed form solutions or efficient numerical solutions for these models have been difficult to obtain except for models which may be solved by techniques of local balance (B2,C4,R1). We consider a general class of continuous transition Markovian models with finite state spaces (D1). We will look at two queue models with characteristics of computing systems such as First-Come-First-Served (FCFS) queueing disciplines, priority queueing disciplines, non-exponential service time distributions, customer dependent service distributions, and multiple identical servers. These models are useful by themselves in modeling computing systems; they are especially useful in determining approximate solutions for more general models (see Chapter V). We present algorithms for solution of these models and demonstrate the use of the models in computer system simulation.

Since the models considered have finite state spaces, we can determine equilibrium state probabilities by solution of the balance equations for this state space (D1). If we let P(i) be the equilibrium probability of state i, $i = 1,\ldots,N$, and $\lambda_{i,j}$ be the rate of transitions from state i to state j, $i,j = 1,\ldots,N$, then the ith balance equation, $i = 1,\ldots,N$, will be

$$\lambda_{1,i}P(1) + \ldots + \lambda_{i-1,i}P(i-1) - \sum_{j\neq i} \lambda_{i,j}P(i) + \lambda_{i+1,i}P(i+1)$$

$$+ \ldots + \lambda_{N,i}P(N) = 0 \qquad (4.1)$$

In addition to these N equations we know that

$$P(1) + \ldots + P(N) = 1 \qquad (4.2)$$

If we substitute this equation for one of the balance equations, we can solve the resulting set of equations for the equilibrium state probabilities. From the equilibrium state probabilities we can determine model statistics such as customer throughput, server utilizations, queue lengths and wait times.

In general, the state space may be very large and solutions obtained by direct numerical techniques may require excessive memory and computation. Iterative numerical techniques (W1) have been successfully used for models such as these, but these techniques still require large amounts of memory and computation. Generally, these techniques will require large amounts of memory because all equilibrium state probabilities are determined and stored before model statistics are determined.

Herzog, Chandy and Woo (H1) have developed a general approach to numerical solution of Markovian models with structured state spaces. This approach determines the equilibrium state probabilities of a small subset of the state space. From these probabilities the probabilities of the other states can be directly obtained, if those probabilities are desired. In practice, the model statistics can be determined in terms of the probabilities of the states of the subset while the probabilities of these states are determined. The probabilities of most of the states are neither determined nor stored. Thus this approach does not require a large amount of memory compared to other techniques; this approach is also efficient in terms of computation.

In section 4.2 we illustrate this approach for a very simple model. In section 4.3 we consider a general representation of a large class of

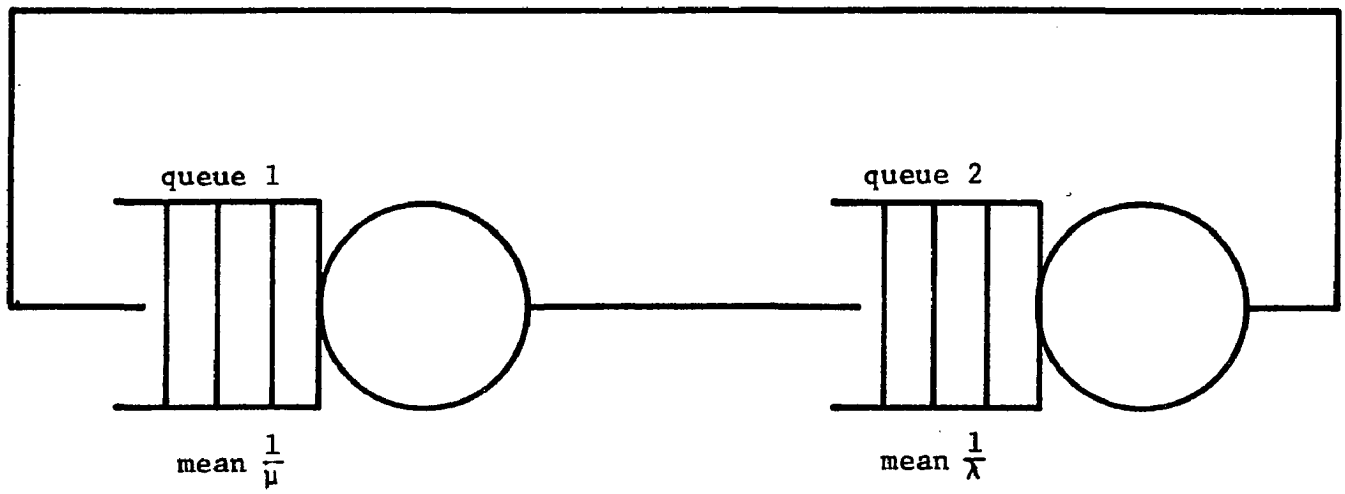



Figure 4.1

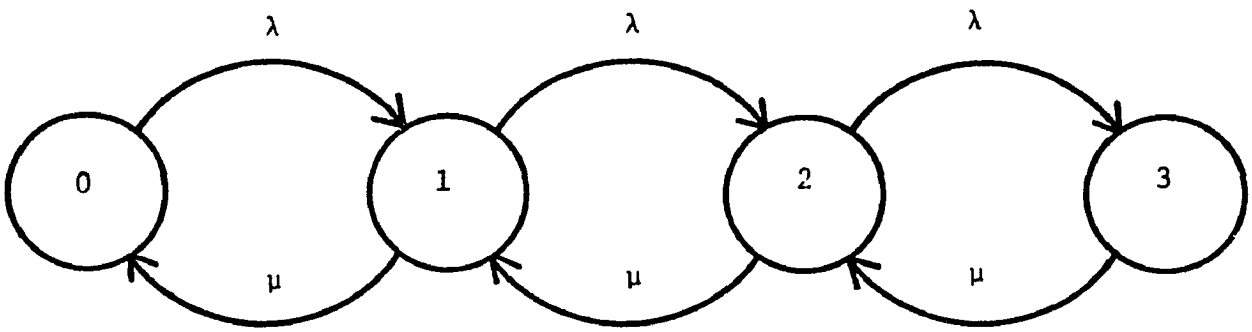



Figure 4.2

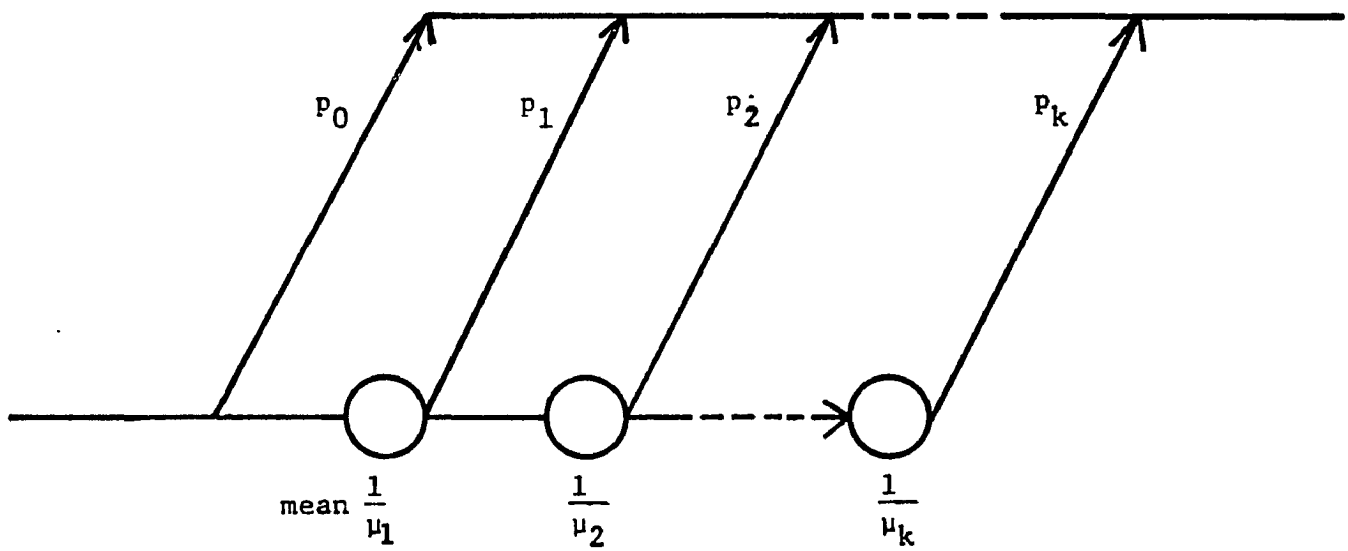



Figure 4.3

service time distributions. This representation is appropriate to both the solution approach and the models of interest. In section 4.4 we illustrate techniques for more general models. In section 4.5 we apply the approach to several important classes of models, and in section 4.6 we consider application to computer system modeling.

## 4.2 Two Exponential Queues - The General Approach

Consider a model with two FCFS queues, a single class of customers, and exponential service time distributions. Assume that customers completing service at one queue always proceed to the other queue. See Figure 4.1. Let the mean service at queue 1 be $1/\mu$ and the mean service at queue 2 be $1/\lambda$. Assume there are N customers in the network. The state of the network can be determined by the number of customers in the first queue. Let P(i), $i = 0,\ldots,n$, be the probability of state i. The state transition diagram for $N = 3$ is given in Figure 4.2. Let $\overline{T}$, $\overline{U}$, $\overline{Q}$, and $\overline{W}$ be the throughput, utilization, mean queue length and mean wait time, respectively, of queue. Let $\overline{C}$ be the cycle time required for a customer to make a complete cycle through the network. The following algorithm will determine P(0), $\overline{C}$, $\overline{T}$, $\overline{U}$, $\overline{Q}$, and $\overline{W}$. Other moments of the wait time distribution and the queue length distribution can be determined in a similar manner. The statistics for queue 2 may be determined in a similar manner, or derived from the statistics for queue 1.

Algorithm 4.1

1. Initialization

$$
\begin{aligned}
P(0) &= 1 \\
P(1) &= \lambda/\mu \\
S &= 1 + \lambda/\mu \\
U &= \lambda/\mu \\
Q &= \lambda/\mu \\
W &= \lambda/\mu
\end{aligned}
$$

2. Iteration. Do step 2 for n = 2, N

$P(i) = ((\lambda+\mu)P(n-1) - \lambda P(n-2))/\mu$ ←(Note: this follows directly from balance equation n-1.)
$S = S + P(n)$
$U = U + P(n)$
$Q = Q + nP(n)$
$W = W + n\lambda P(n-1)/\mu$

Note: After each application of step 2, we may reclaim the storage used for P(n-2).

3. Determination of statistics.

$\underline{P}(0) = 1/S$
$\underline{U} = U\underline{P}(0)$
$\underline{T} = \mu\underline{U}$
$\underline{C} = N/\underline{T}$
$\underline{Q} = QP(0)$
$\underline{W} = WP(0)/\underline{T}$

Note: Many simplifications of this algorithm are possible. It is presented in this form for the sake of clarity and so that extension to more complex models be straightforward. In particular, note that it will often be desirable for $\mu$ and/or $\lambda$ to be dependent on n. Note that we may alternatively calculate $\overline{W}$ as $\overline{Q}/\overline{T}$ (Little's Rule (L3)). If $\mu$ and $\lambda$ are not state dependent, then we can easily determine solutions for N + 1 from the values determined for N; this is significant in parametric analysis.

## 4.3 Generalized Erlang Distributions

Erlang developed a distribution form consisting of a sequence of exponential stages, each of which must be completed by a customer before proceeding to the next. This form is less skewed then the exponential. Cox (C5) proposed a generalization of Erlang's form which allows a customer to bypass the remaining service stages according to a fixed probability

dependent only on the stage about to be entered. Figure 4.3 illustrates this form. Assuming that there are K stages, let $p_i$, $i = 0,\ldots,K$ be the probability of bypassing the remaining stages after completing stage i, where it is understood that $p_o$ is the probability of bypassing all stages initially, and that $p_K$ is identically 1. Let $\bar{p}_i = 1-p_i$, $i = 0,\ldots,K$ and let $1/\mu_i$ be the mean holding time of the ith stage, $i = 1,\ldots,K$. This form has the Laplace transform

$$f^*(s) = p_o + \sum_{i=1}^{K} \bar{p}_o \cdots \bar{p}_{i-1}\, p_i \prod_{j=1}^{i} (\mu_j/(\mu_j + s)) \qquad (4.3)$$

Cox showed that this distribution form can represent arbitrary distributions with rational Laplace transforms, provided that K is sufficiently large and that we allow the probabilities and holding times to be complex valued.

Though the algorithms we present can be used with this general form, there are several difficulties:

1) When $p_o \neq 0$ the solution techniques may become more complex. As we shall demonstrate, the Cox form is still quite general when we assume that $p_o$ is identically zero. We shall make this assumption.

2) It is difficult to apply intuition to complex holding times, complex probabilities and/or real probabilities outside the interval [0,1]. Further, we do not know how to simulate non-standard probabilities. (We will refer to real probabilities in the interval [0,1] as "standard" probabilities.) Thus we cannot directly compare analytic results with simulation results when we allow non-standard probabilities. We shall show that

many interesting and important distributions may be represented by the Cox form even when we only allow standard probabilities.

3. When we wish to represent distributions with small coefficient of variation, we must use many stages when we use the Cox form or any form consisting of a network of exponential stages. We shall discuss what we call "pseudo-PDF's" which may accurately approximate distributions with arbitrarily small coefficient of variation using as few as two exponential stages. By pseudo-PDF we mean negative valued functions similar to probability density functions.

We will now discuss in detail the second and then the third problem mentioned above.

The hyper-exponential distribution (M4) is often used in modeling service times in computer systems. See Figure 4.4. With two stages, positive real holding times and standard probabilities, this distribution form allows arbitrarily large coefficient of variation (the standard deviation divided by the mean). Additional stages may be used to more accurately reflect the higher moments of a given distribution. If there are K stages, with standard probabilities $q_i$, such that $\sum_{i=1}^{K} q_i = 1$, and mean holding times $1/\mu_i$, i=1,...,K then the Laplace transform is

$$f^*(s) = \sum_{i=1}^{K} q_i \mu_i / (\mu_i + s) \qquad (4.4)$$

Theorem 4.1: For K = 2,3, a hyper-exponential distribution of the above form may be represented by a generalized Erlang (GE) distribution with the same number of stages, the same mean holding times for corresponding stages, and standard probabilities.

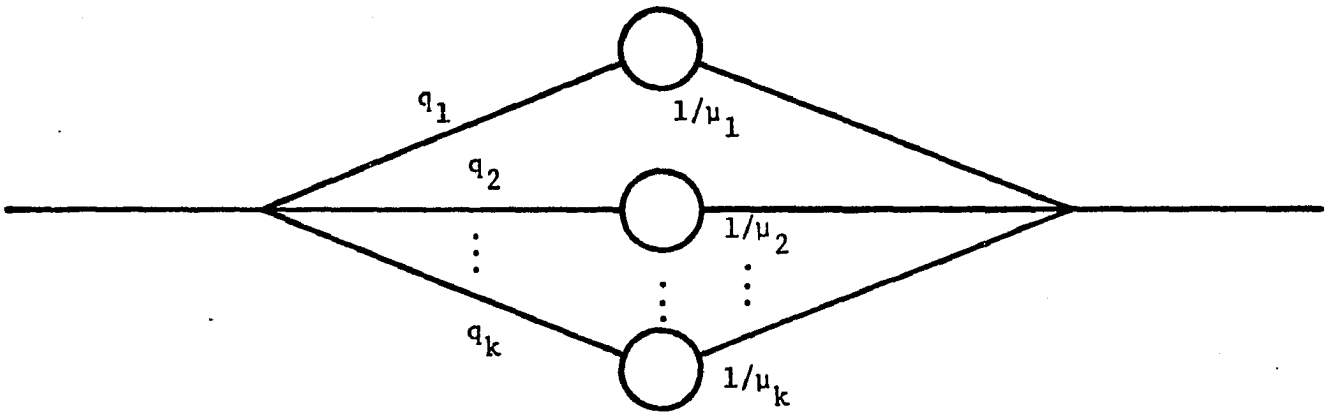



Figure 4.4

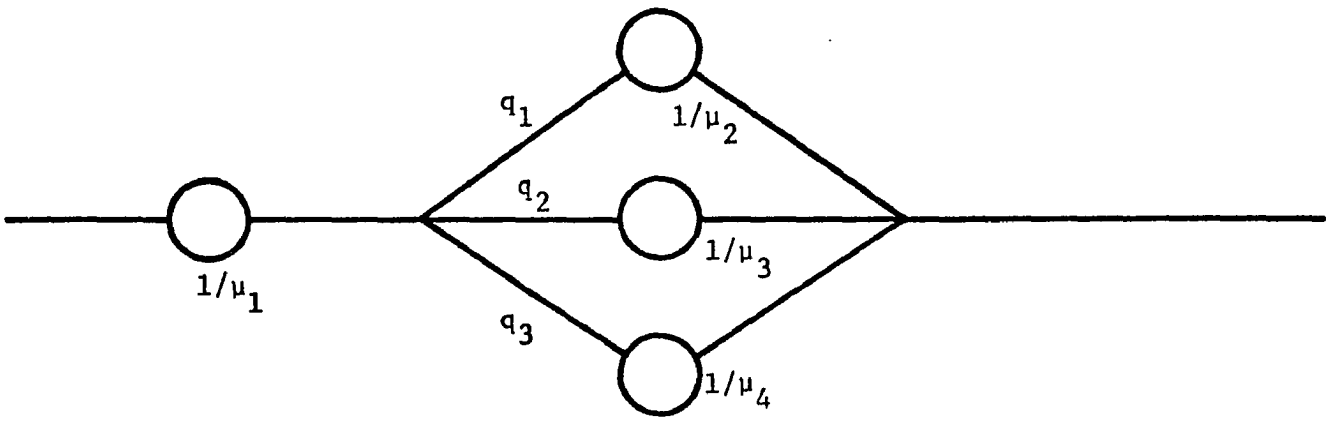



Figure 4.5

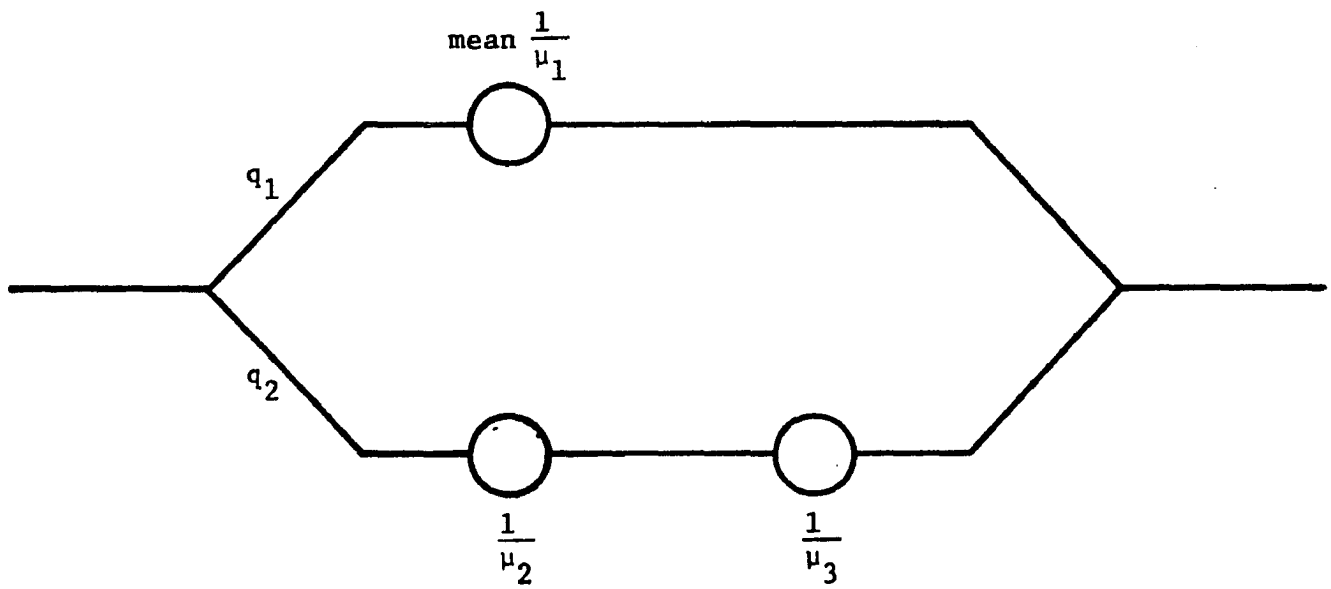



Figure 4.6

**Proof:** We consider the case for K = 3. A similar argument holds for K = 2. Assume without loss of generality that $\mu_1 \geq \mu_2 \geq \mu_3$ and that $q_i > 0$, $i = 1,2,3$. Let

$$p_1 = q_1 + (q_2\mu_2 + q_3\mu_3)/\mu_1$$

$$p_2 = \frac{q_2\mu_1\mu_2 + q_3\mu_1\mu_3 - q_2\mu_2^2 - q_3\mu_3^2}{q_2\mu_1\mu_2 + q_3\mu_1\mu_2 - q_2\mu_2^2 - q_3\mu_2\mu_3}$$

It is easy to show that $p_1$ and $p_2$ lie in the interval [0,1]. By direct substitution of these values in (4.3) one can obtain an expression equivalent to (4.4). Thus the GE distribution with K = 3 is equivalent to the hyper-exponential formulation.

**Conjecture:** The above result holds for arbitrary $K \geq 2$.

Many other interesting distributions can be represented by the restricted GE distribution limited to standard probabilities. For example, distributions with two or three modes may be represented by distributions of the form in Figure 4.5. It is clear from the proof of Theorem 4.1 that the form of Figure 4.5 is equivalent to a GE distribution with standard probabilities. Of course, Erlang and hypo-exponential (M1) forms are cases of the restricted form. Distributions of the form in Figure 4.6 may be represented by the restricted GE form with standard probabilities if $\mu_2$ or $\mu_3 \leq \mu_1$. Otherwise, the restricted form with real probabilities may be used. From the above results, it is plausible that arbitrary distributions consisting of networks of exponential stages may be represented by the Cox form with real probabilities. Since networks of exponential stages may be used to represent arbitrary distributions with rational Laplace transforms (C1),

correctness of this conjecture would imply that we may restrict the Cox form to real probabilities and holding times and still be able to represent arbitrary distributions with rational Laplace transforms.

We now consider some surprising results obtained by using "pseudo probability density functions" (pseudo-PDF's). We do not fully understand these functions or their application, but they seem potentially useful in computer systems modeling and worthy of further study. A general problem with any distribution form consisting of exponential stages is that the minimum obtainable coefficient of variation is $1/\sqrt{K}$. Thus many stages must be used to represent distributions with little variance, and an infinite number of stages must be used to represent a constant distribution. The pseudo-PDF's we consider may have arbitrarily small coefficient of variation with as few as two stages.

Consider a Cox form with $K = 2$, $p_o = 0$, $\mu_1 = \mu_2 = 2 + \sqrt{2}$ and $p_1 = -\sqrt{2}$. A graph of this function is shown in Figure 4.7. This function has "mean" 1 and "variance" of 0. It is clearly not a PDF because it is not strictly non-negative. However, it is like a PDF in that its integral from zero to infinity is 1. Using this function with algorithm 4.3 we obtained results nearly identical to those obtained by Gaver (G1) for utilization of a CPU with constant service times. The maximum disagreement with the results obtained by Gaver was .004. Using the techniques of Chapter V, we used this form to analyze more general models with constant CPU service times. The results of this analysis were in good agreement with simulation results for CPU utilization, mean and standard deviation of queue length and wait times. (See Tables 5.1, 5.2).

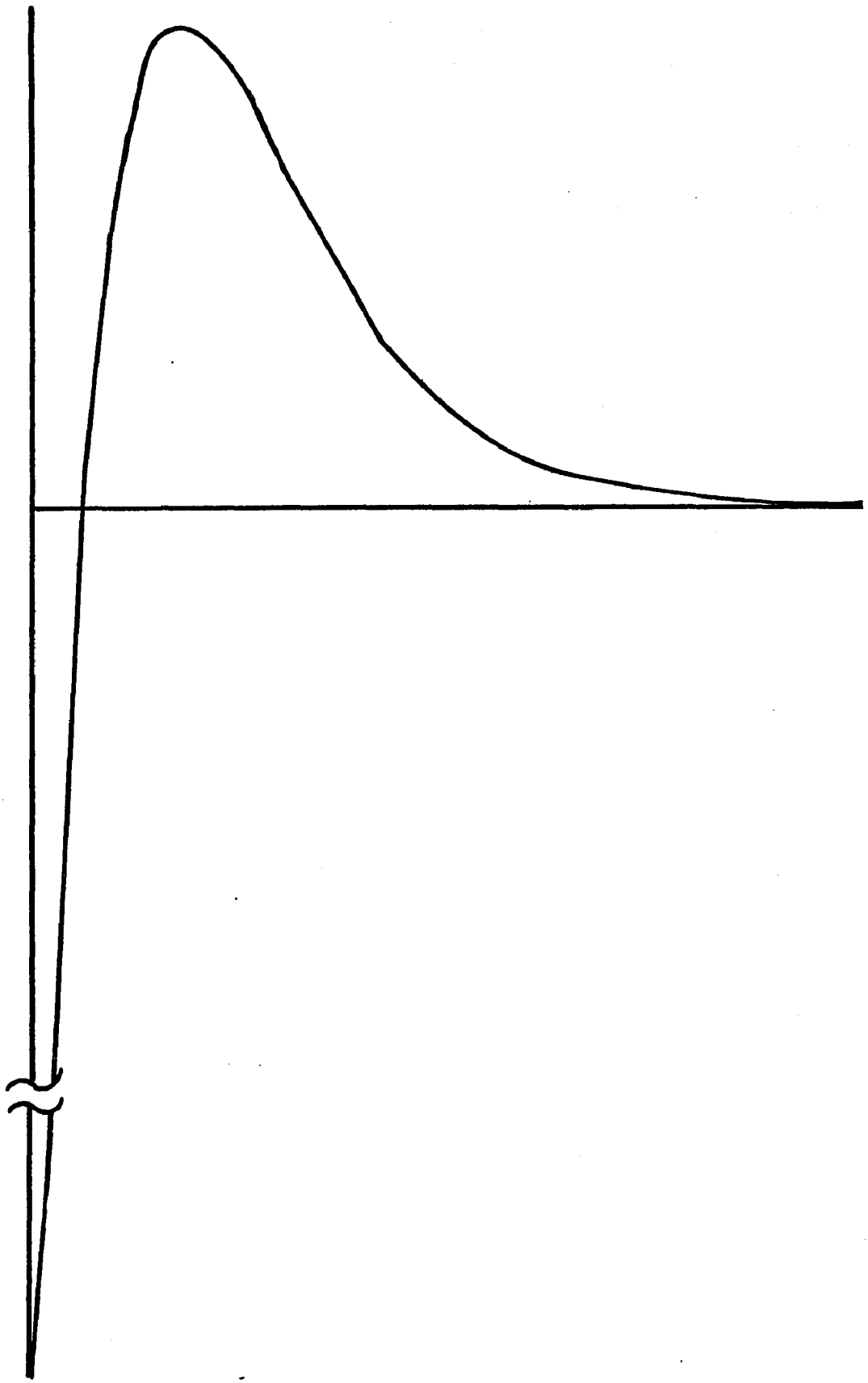


Figure 4.7

These results are exciting, but pseudo-PDF's must be used with caution until they are better understood. It is possible to obtain physically impossible results using the form described above. Consider a model similar to that of section 4.2 with two customers, a pseudo-PDF for the first queue service time, and an exponential distribution with mean $1/\lambda$ for the second queue service time. For the pseudo-PDF, let $\mu_1 = \mu_2 = 1$ and $p_o = -\sqrt{2}$. This form has mean $2 + \sqrt{2}$ and variance 0. If $\lambda \neq 1/\sqrt{2}$, it is easily shown (see Algorithm 4.3) that the utilization of server 1 is

$$1 - (1 - \lambda\sqrt{2})/(1 + 2\lambda + (4 + 2\sqrt{2})\lambda^2 + (2 + \sqrt{2})\lambda^3) \qquad (4.5)$$

Clearly this value will be greater than 1, a physically impossible situation, if $\lambda$ is greater than $1/\sqrt{2}$. However disconcerting this may be, it can be interpreted as a reasonable amount of error. For example, if $\lambda = 1$, then the value of (4.5) will be approximately 1.03. Using the analysis of (S4) we find that the correct value for a model with constant service of $2 + \sqrt{2}$ is approximately .99. We will not pursue these forms further here, but suggest that they be further studied in the future.

## 4.4 Two Queues - One GE, One Exponential

We now illustrate techniques for models with one queue having the GE form of service distribution and a second queue having an exponential distribution; otherwise, the model is as in section 4.2. We will often omit the subscript when referring to $p_1$. It is straightforward to extend the algorithm to allow customers to feedback to the queue they are departing from according to fixed probabilities. We also assume that K = 2; extension to other values of K is straightforward.

Let the state of the model be represented by the ordered pair (i,n) where there are n customers in queue 1 and the customer being served is in

stage i of the service distribution. For notational convenience, let i ≡ 1 when n = 0. Figure 4.8 gives the state transition diagram for N = 3. We present two algorithms; the first determines P(1,N) and P(2,N) and the second determines P(1,0). Though the second algorithm is preferable for this model, the techniques of both algorithms are applied to more complex models. Determination of model statistics is not included, these may be determined in a straightforward manner similar to Algorithm 4.1.

Figure 4.8 is partitioned into three groups of states by dashed lines. Algorithm 4.2 "sweeps through" the state diagram according to these partitions. Step 1 determines values for the states in the top group. Step 2 determines values for the middle group. Step 3 determines values for the bottom group of states and uses these to determine the model solutions.

Algorithm 4.2 (assume N > 1)

Throughout most of the algorithm we represent state probabilities as two-element column vectors; at the end of step 3 P(1,N) and P(2,N) are determined as scalars.

1. Initialization

$$P(1,N) = (1,0)^T$$
$$P(2,N) = (0,1)^T$$
$$P(1,N-1) = (\mu_1/\lambda)P(1,N)$$
$$P(2,N-1) = (\mu_2/\lambda)P(2,N) - (\bar{p}\,\mu_1/\lambda)P(1,N)$$
$$S = P(1,N) + P(2,N) + P(1,N-1) + P(2,N-1)$$

2. Iteration. For n = N-2,...,1 do step 2.

$$P(1,i) = (1 + \mu_1/\lambda)P(1,n+1) - (p\,\mu_1/\lambda)P(1,n+2) - (\mu_2/\lambda)P(2,n+2)$$
$$P(2,i) = (1 + \mu_2/\lambda)P(2,n+1) - (\bar{p}\,\mu_1/\lambda)P(1,n+1)$$
$$S = S + P(1,n) + P(2,n)$$

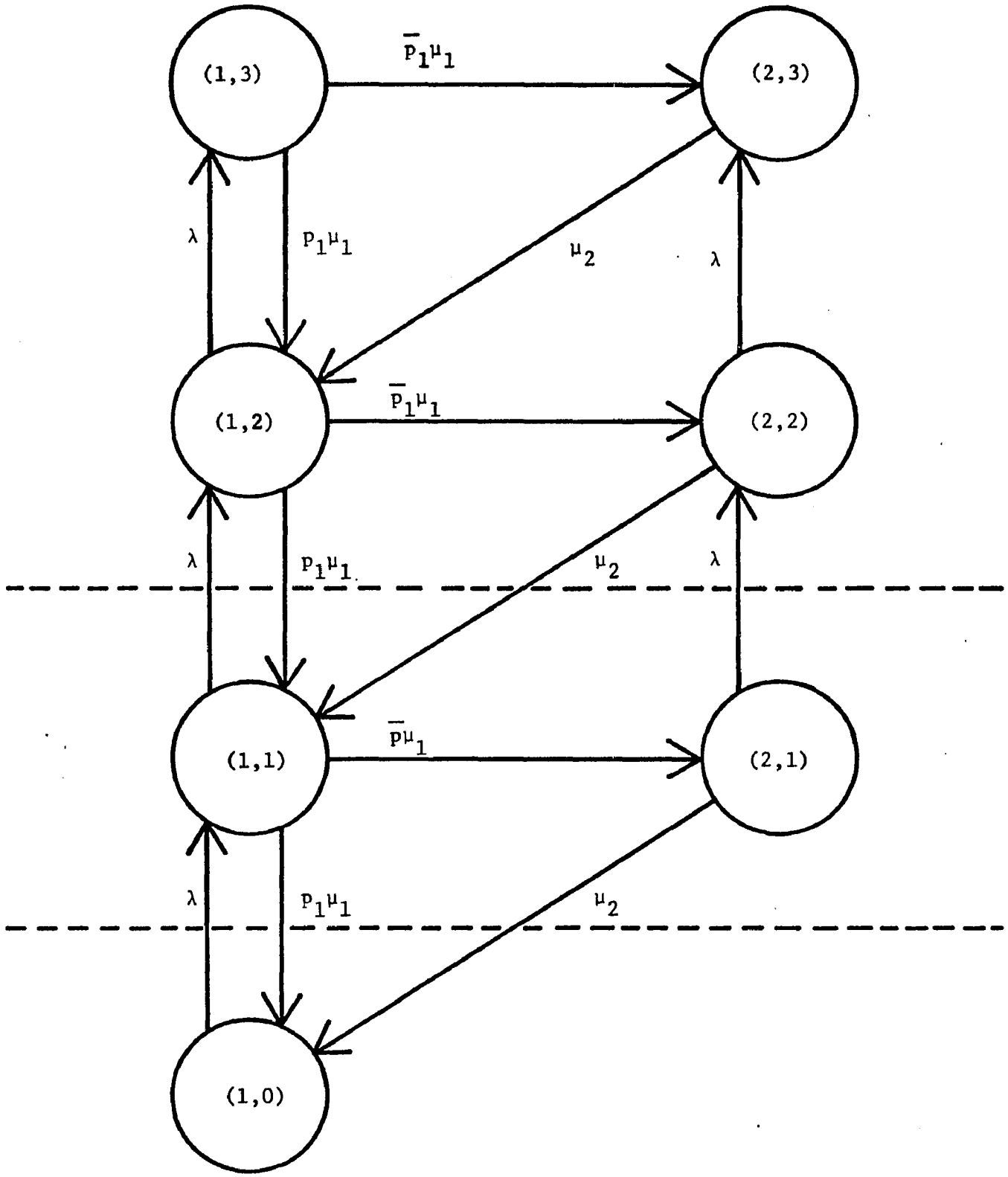



**Figure 4.8**

3. Determination of P(1,N), P(2,N) as scalars

$P(1,0) = (1 + \mu_1/\lambda)P(1,1) - (p\mu_1/\lambda)P(1,2) - (\mu_2/\lambda)P(2,2)$
$S = S + P(1,0)$
$D = P(2,1) - (\bar{p}\ \mu_1/(\mu_2 + \lambda))P(1,1)$

Note: D is the difference of the value of P(2,1) as determined from the balance equations for (2,2) and the value of P(2,1) as determined from the balance equations for (2,1). Thus the inner product of D and the vector consisting of the scalar values of P(1,N) and P(2,N) must be 0.

Solve $\begin{pmatrix} D^T \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ for x

$P(1,N) = \zeta_1 \qquad P(2,N) = \zeta_2$

where $\zeta_i$ is the ith element of x.

Algorithm 4.3 (assume N > 1)

1. Initialization

$P(1,0) = 1$
$P(1,1) = (\lambda/(p\mu_1 + p\mu_1\mu_2/(\mu_2 + \lambda)))P(1,0)$

This expression is obtained from the balance equations for state (1,0) and (2,1) as follows:

$$p\mu_1 P(1,1) + \mu_2 P(2,1) = \lambda P(1,0) \tag{4.6}$$

$$(\mu_2 + \lambda)P(2,1) = \bar{p}\mu_1 P(1,1) \tag{4.7}$$

dividing each side of (4.7) by $\mu_2 + \lambda$ and substituting into (4.6) yields

$$p\mu_1 P(1,1) + \frac{\mu_2 \bar{p}\mu_1}{\mu_2+\lambda} P(1,1) = \lambda P(1,0) \tag{4.8}$$

collecting terms and dividing by $p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda)$ yields

$$P(1,1) = \frac{\lambda}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda)} P(1,0) \tag{4.9}$$

$P(2,1) = \bar{p}\mu_1 P(1,1)/(\mu_2 + \lambda)$

$S = P(1,0) + P(1,1) + P(2,1)$

2. Iteration. Do step 2 for n = 2,N-1

$$P(1,n) = \frac{(\lambda + \mu_1)P(1,n-1) - \lambda P(1,n-2) - \mu_2\lambda P(2,n-1)/(\mu_2 + \lambda)}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda)}$$

This expression is obtained from the balance equation for states (1,n-1) and (2,n) as follows

$$p\mu_1 P(1,n) + \mu_2 P(2,n) + \lambda P(1,n-2) = (\lambda + \mu_1)P(1,n-1) \tag{4.10}$$

$$(\mu_2 + \lambda)P(2,n) = \bar{p}\mu_1 P(1,n) + \lambda P(2,n-1) \tag{4.11}$$

$$P(2,n) = \frac{\bar{p}\mu_1}{\mu_2+\lambda} P(1,n) + \frac{\lambda}{\mu_2+\lambda} P(2,n-1) \tag{4.12}$$

$$p\mu_1 P(1,n) + \frac{\bar{p}\mu_1\mu_2}{\mu_2+\lambda} P(1,n) + \frac{\lambda\mu_2}{\mu_2+\lambda} P(2,n-1) + \lambda P(1,n-2) =$$

$$(\lambda + \mu_1)P(1,n-1) \tag{4.13}$$

$$P(1,n) = \frac{\lambda+\mu_1}{p\mu_1+\bar{p}\mu_1\mu_2/(\mu_2+\lambda)} P(1,n-1)$$

$$- \frac{\lambda}{p\mu_1+\bar{p}\mu_1\mu_2/(\mu_2+\lambda)} P(1,n-2)$$

$$- \frac{\lambda\mu_2}{(\mu_2+\lambda)p\mu_1+\bar{p}\mu_1\mu_2} P(2,n-1) \tag{4.14}$$

$P(2,n) = (\bar{p}\mu_1 P(1,n) + \lambda P(2,n-1))/(\mu_2 + \lambda)$

$S = S + P(1,n) + P(2,n)$

3. Determination of P(1,0)

$P(2,N) = ((\lambda + \mu_1)P(1,N-1) - \lambda P(1,N-2) - \lambda P(2,N-1))/\mu_1$

$P(2,N) = (\bar{p}\mu_1 P(1,N) + \lambda P(2,N-1))/\mu_2$

$S = S + P(1,N) + P(2,N)$

$P(1,0) = 1/S$

## 4.5 Application to More General Models

We now present algorithms for a variety of important models. These algorithms are not as general as possible. They are intended to illustrate technique. We have implemented more general versions of each of these algorithms, in Fortran for a CDC 6600. These algorithms may be combined to consider models with several of the features considered below.

### 4.5.1 Two Non-Exponential Queues

Consider a two queue network as in Figure 4.1, with N identical customers, with FCFS disciplines at both queues, and GE distributions at both queues. Assume that each distribution has two stages, with rates $\mu_1$ and $\mu_2$ and probability $p_1$ for queue 1, and with rates $\lambda_1$ and $\lambda_2$ and probability $q_1$ for queue 2. As before, we will often omit the subscripts on p and q. We may define a state of this system as a triple (i,j,n), where n = 0,...,N is the number of customers in queue 1, i = 1,2 is the service stage of the customer being served at queue 1, and j = 1,2 is the service stage of the customer being served at queue 2. For notational convenience, let i be 1 when n = 0, and let j be 1 when n = N. Let P(i,j,n) be probability of state (i,j,n). The state transition diagram for N = 3 is given in Figure 4.9.

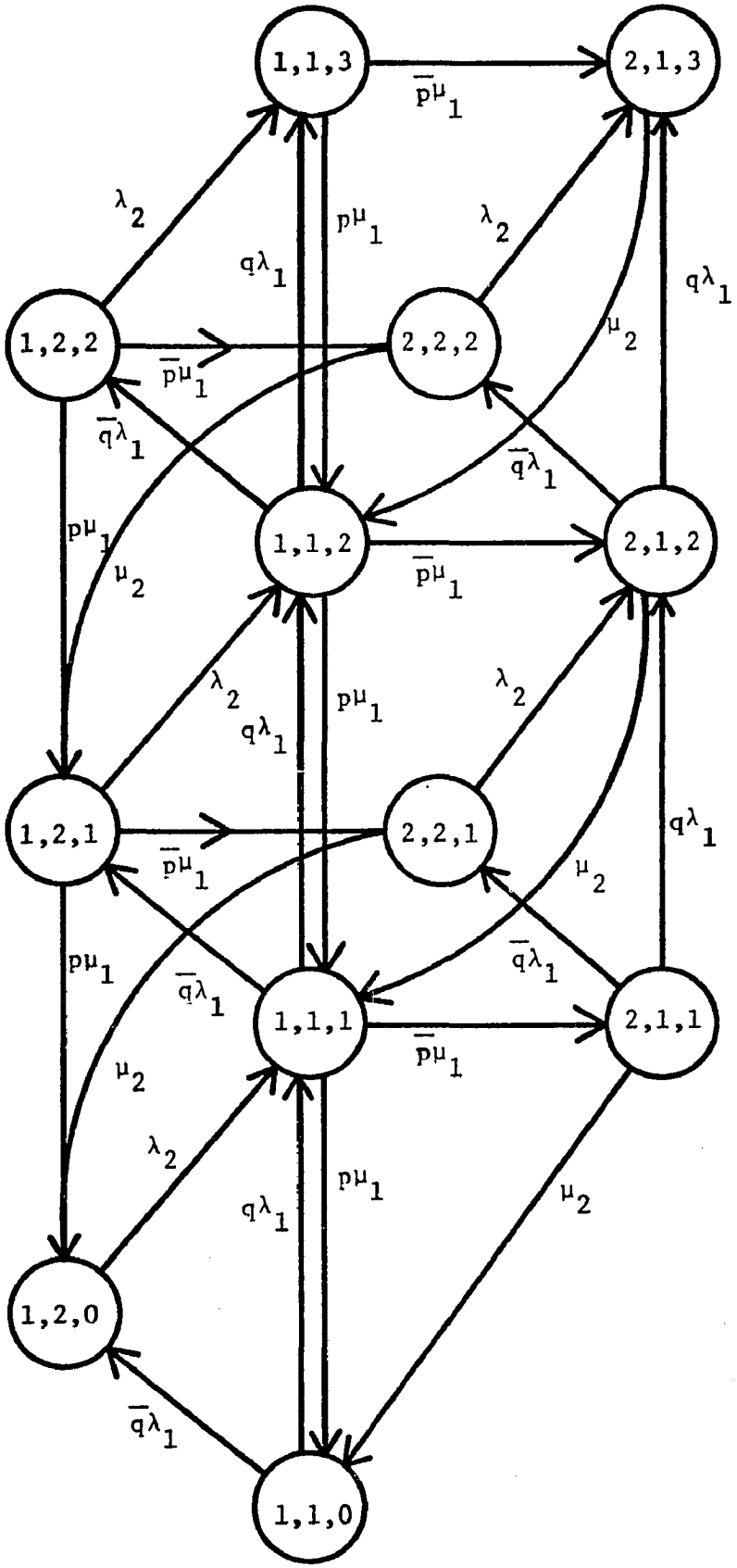



**Figure 4.9**

**Algorithm 4.4** Determination of P(1,1,0) and P(1,2,0) (assume N > 1)

1. Initialization

$$P(1,1,0) = (1,0)^T$$

$$P(1,2,0) = (0,1)^T$$

$$P(1,1,1) = \frac{\lambda_1}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda_1)} P(1,1,0)$$

$$P(2,1,1) = \frac{\bar{p}\mu_1}{\mu_2 + \lambda_1} P(1,1,1)$$

$$P(1,2,1) = \frac{\lambda_2}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda_2)} P(1,2,0)$$

$$- \frac{\bar{q}\lambda_1}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda_2)} P(1,1,0)$$

$$- \frac{\bar{q}\mu_2\lambda_1}{p\mu_1(\mu_2 + \lambda_2) + \bar{p}\mu_1\mu_2} P(2,1,1)$$

$$P(2,2,1) = \frac{\bar{p}\mu_1}{\mu_2 + \lambda_2} P(1,2,1) + \frac{\bar{q}\lambda_1}{\mu_2 + \lambda_2}$$

$$S = P(1,1,0)+P(1,2,0)+P(1,1,1)+P(1,2,1)+P(2,1,1)+P(2,2,1)$$

2. Iteration. Do step 2 for n = 2,3,...,N-1

$$P(1,1,n) = \frac{\lambda_1+\mu_1}{p\mu_1+\bar{p}\mu_1\mu_2/(\mu_2+\lambda_1)} P(1,1,n-1) - \frac{\mu_2 q\lambda_1}{(\mu_2+\lambda_1)p\mu_1+\bar{p}\mu_1\mu_2} \cdot$$

$$P(2,1,n-1) - \frac{\mu_2\lambda_2}{(\mu_2+\lambda_1)p\mu_1+\bar{p}\mu_1\mu_2} P(2,2,n-1) - \frac{q\lambda_1}{p\mu_1+p\mu_1\mu_2/(\mu_2+\lambda_1)} \cdot$$

$$P(1,1,n-2) - \frac{\lambda_2}{p\mu_1+p\mu_1\mu_2/(\mu_2+\lambda_1)} P(1,2,n-2)$$

$$P(2,1,n) = \frac{\bar{p}\mu_1}{\mu_2+\lambda_1} P(1,1,n) + \frac{q\lambda_1}{\mu_2+\lambda_1} P(2,1,n-1) + \frac{\lambda_2}{\mu_2+\lambda_1} P(2,2,n-1)$$

$$P(1,2,n) = \frac{\mu_2 + \lambda_1}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda_2)} P(1,2,n-1)$$

$$- \frac{\mu_2\bar{q}\lambda_1}{(\mu_2 + \lambda_2)p\mu_1 + \bar{p}\mu_1\mu_2} P(2,1,n)$$

$$- \frac{\bar{q}\lambda_1}{p\mu_1 + \bar{p}\mu_1\mu_2/(\mu_2 + \lambda_2)} P(1,1,n-1)$$

$$P(2,2,n) = \frac{\bar{p}\mu_1}{\mu_2 + \lambda_2} P(1,2,n) + \frac{\bar{q}\lambda_1}{\mu_2 + \lambda_2} P(2,1,n)$$

$$S = S + P(1,1,n) + P(2,1,n) + P(1,2,n) + P(2,2,n)$$

3. Determination of P(1,1,0), P(1,2,0) as scalars

$$P(1,1,N) = \frac{\lambda_1 + \mu_1}{\mu_1} P(1,1,N-1) - \frac{q\lambda_1}{\mu_1} P(2,1,N-1)$$

$$- \frac{\lambda_2}{\mu_1} P(2,2,N-1) - \frac{q\lambda_1}{\mu_1} P(1,1,N-2)$$

$$- \frac{\lambda_2}{\mu_1} P(1,2,N-2)$$

$$P(2,1,N) = \frac{\bar{p}\mu_1}{\mu_2} P(1,1,N) + \frac{q\lambda_1}{\mu_2} P(2,1,N-1) + \frac{\lambda_2}{\mu_2} P(2,2,N-1)$$

$$S = S + P(1,1,N) + P(2,1,N)$$

$$D = P(2,1,N-1) - \frac{\bar{q}\lambda_1}{\mu_1 + \lambda_2} P(1,1,N-1)$$

$$\text{solve} \begin{pmatrix} D^T \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{for } x$$

$$P(1,1,0) = \zeta_1 \qquad P(1,2,0) = \zeta_2$$

Our computer implementation of this algorithm allows the number of stages at each queue to be 1, 2 or 3, independent of the number of stages at the other queue. In this implementation, the rates of each stage of the distribution for the second queue may be a function of the number of customers in that queue; when these rates are not queue length dependent, the program can determine results for a range of numbers of customers in the model without redetermining intermediate results.

### 4.5.2 Multiple Identical Servers

We now consider models with two identical servers at queue 1. The algorithm we present can be extended to more than two servers. We assume the service time distribution for queue 1 is of the above form with two stages and the second queue service distribution is exponential with mean $1/\lambda$; extension to both distributions of the Cox form is straightforward. We assume that one server is idle when only one customer is in queue 1; very minor changes in the algorithm are required to consider cooperation of the two servers when there is only one customer in the queue. The state of the model can be described as a triple (i,j,n) where there are n customers in queue 1, the customer being served at one server is in stage i, and the customer being served at the other server is in stage j, where $i \leq j$. For notational convenience, we have i identically 1 when n = 1, and i and j identically 1 when n = 0. The state transition diagram for N = 3 is given in Figure 4.10.

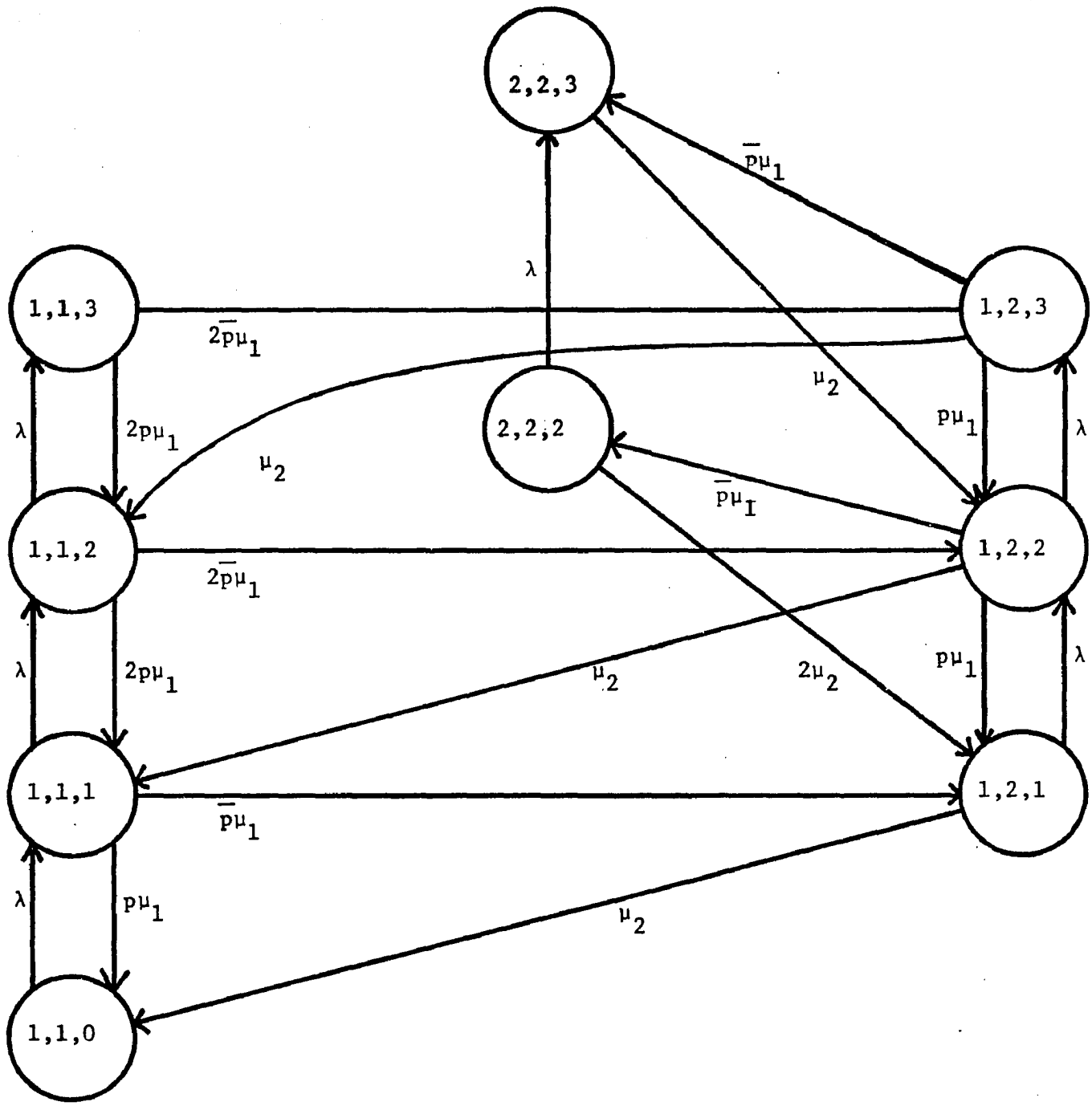



Figure 4.10

**Algorithm 4.5** Determination of P(1,1,1) and P(1,2,1) (assume N > 2)

1. **Initialization**

$$P(1,1,1) = (1,0)^T$$

$$P(1,2,1) = (0,1)^T$$

$$P(1,1,0) = \frac{p\mu_1}{\lambda} P(1,1,1) + \frac{\mu_2}{\lambda} P(1,2,1)$$

$$P(1,2,2) = \frac{\lambda + \mu_2}{p\mu_1 + 2\bar{p}\mu_1\mu_2/(2\mu_2 + \lambda)} P(1,2,1)$$

$$- \frac{\bar{p}\mu_1}{p\mu_1 + 2\bar{p}\mu_1\mu_2/(2\mu_2 + \lambda)} P(1,1,1)$$

$$P(2,2,2) = \frac{\bar{p}\mu_1}{2\mu_2 + \lambda} P(1,2,2)$$

$$P(1,1,2) = \frac{\lambda + \mu_1}{2p\mu_1} P(1,1,1) - \frac{\mu_2}{2p\mu_1} P(1,2,2) - \frac{\lambda}{2p\mu_1} P(1,1,0)$$

$$S = P(1,1,1) + P(1,1,0) + P(1,2,2) + P(2,2,2) + P(1,1,2)$$

2. **Iteration.** Do step 2 for n = 3,N-1

$$P(1,2,n) = \frac{\lambda+\mu_1+\mu_2}{p\mu_1 + 2\bar{p}\mu_1\mu_2/(2\mu_2 + \lambda)} P(1,2,n-1)$$

$$- \frac{2\mu_2\lambda}{(2\mu_2 + \lambda)p\mu_1 + 2\bar{p}\mu_1\mu_2} P(2,2,n-1)$$

$$- \frac{2\bar{p}\mu_1}{p\mu_1 + 2\bar{p}\mu_1\mu_2/(2\mu_2 + \lambda)} p(1,1,n-1)$$

$$- \frac{\lambda}{p\mu_1 + 2\bar{p}\mu_1\mu_2/(2\mu_2 + \lambda)} P(1,2,n-2)$$

$$P(2,2,n) = \frac{\bar{p}\mu_1}{2\mu_2 + \lambda} P(1,2,n) + \frac{\lambda}{2\mu_2 + \lambda} P(2,2,n-1)$$

$$P(1,1,n) = \frac{\lambda + 2\mu_1}{2p\mu_1} P(1,1,n-1) - \frac{\mu_2}{2p\mu_1} P(1,2,n)$$

$$- \frac{\lambda}{2p\mu_1} P(1,1,n-2)$$

$$S = S + P(1,2,n) + P(2,2,n) + P(1,1,n)$$

3. Determination of P(1,1,1) and P(1,2,1) as scalars

$$P(1,2,N) = \frac{\lambda+\mu_1+\mu_2}{\mu_1} P(1,2,N-1) - \frac{\lambda}{\mu_1} P(2,2,N-1)$$

$$- 2\bar{p} P(1,1,N-1) - \frac{\lambda}{\mu_1} P(1,2,N-2)$$

$$P(2,2,N) = \frac{\bar{p}\mu_1}{2\mu_2} P(1,2,N) + \frac{\lambda}{2\mu_2} P(2,2,N-1)$$

$$P(1,1,N) = \frac{\lambda + 2\mu_1}{2p\mu_1} P(1,1,N-1) - \frac{\mu_2}{2p\mu_1} P(1,2,N)$$

$$- \frac{\lambda}{2p\mu_1} P(1,1,N-2)$$

$$S = S + P(1,2,N) + P(2,2,N) + P(1,1,N)$$

$$D = P(1,1,N) - \frac{\lambda}{2\mu_1} P(1,1,N-1)$$

$$\text{Solve} \begin{pmatrix} D^T \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$P(1,1,1) = \zeta_1$$

$$P(1,2,1) = \zeta_2$$

Our computer implementation of the above algorithm allows $\lambda$ to be a function of the number of customers in queue 2. It will handle models where both servers in queue 1 cooperate when a single customer is present in that queue.

Note that algorithm 4.5 is not defined for p = 0 because of the division by $2p\mu_1$ in step 2. Of course the algorithm will be unstable when p is near zero. Algorithm 4.5E (Erlang) will handle the case where p is zero. It would be straightforward to develop an algorithm which determines P(1,1,N), P(1,2,N) and P(2,2,N) and would not be sensitive to the value of p, but this would require more memory, especially when extended to multiple classes. We expect that Algorithm 4.5E could be modified to consider the general case, but we have not done so. This modified algorithm would likely require memory comparable to Algorithm 4.5.

Algorithm 4.5E Determination of P(1,1,0)
(assume p = 0, N > 2)

1. Initialization

$$P(1,1,0) = 1$$

$$P(1,2,1) = \frac{\lambda}{\mu_2} P(1,1,0)$$

$$P(1,2,2) = \frac{\lambda + \mu_2}{2\mu_1\mu_2/(2\mu_2 + \lambda) + \mu_1\mu_2/(\mu_1 + \lambda)} P(1,2,1) - \frac{\mu_1\lambda}{(\mu_1 + \lambda)2\mu_1\mu_2/(2\mu_2 + \lambda) + \mu_1\mu_2} P(1,1,0)$$

$$P(2,2,2) = \frac{\mu_1}{2\mu_2 + \lambda} P(1,2,2)$$

$$P(1,1,1) = \frac{\mu_2}{\mu_1 + \lambda} P(1,2,2) + \frac{\lambda}{\mu_1 + \lambda} P(1,1,0)$$

$$S = P(1,1,0) + P(1,2,1) + P(1,2,2) + P(2,2,2) + P(1,1,1)$$

2. Iteration. Do step 2 for n = 3,...,N-1

$$P(1,2,n) = \frac{\lambda+\mu_1+\mu_2}{2\mu_1\mu_2/(2\mu_2+\lambda) + 2\mu_1\mu_2/(2\mu_1+\lambda)} P(1,2,n-1)$$

$$- \frac{2\mu_1\lambda}{(2\mu_1+\lambda)2\mu_1\mu_2/(2\mu_2+\lambda) + 2\mu_1\mu_2} P(1,1,n-2)$$

$$- \frac{2\mu_2\lambda}{2\mu_1\mu_2 + (2\mu_2+\lambda)2\mu_1\mu_2/(2\mu_1+\lambda)} P(2,2,n-1)$$

$$P(2,2,n) = \frac{\mu_1}{2\mu_2+\lambda} P(1,2,n) + \frac{\lambda}{2\mu_2+\lambda} P(2,2,n-1)$$

$$P(1,1,n-1) = \frac{\mu_2}{2\mu_1+\lambda} P(1,2,n) + \frac{\lambda}{2\mu_1+\lambda} P(1,1,n-2)$$

$$S = S + P(1,2,n) + P(2,2,n) + P(1,1,n-1)$$

3. Determination of P(1,1,0)

$$P(1,2,N) = \frac{\lambda+\mu_1+\mu_2}{\mu_1 + 2\mu_1\mu_2/(2\mu_1+\lambda)} P(1,2,N-1)$$

$$- \frac{2\mu_1\lambda}{(2\mu_1+\lambda)\mu_1 + 2\mu_1\mu_2} P(1,1,N-2)$$

$$- \frac{\lambda}{\mu_1 + 2\mu_1\mu_2/(2\mu_1+\lambda)} P(2,2,N-1)$$

$$P(2,2,N) = \frac{\mu_1}{2\mu_2} P(1,2,N) + \frac{\lambda}{2\mu_2} P(2,2,N-1)$$

$$P(1,1,N-1) = \frac{\mu_2}{2\mu_1+\lambda} P(1,2,N) + \frac{\lambda}{2\mu_1+\lambda} P(1,1,N-2)$$

$$P(1,1,N) = \frac{\lambda}{2\mu_1} P(1,1,N-1)$$

$$S = S + P(1,2,N) + P(2,2,N) + P(1,1,N-1) + P(1,1,N)$$

$$P(1,1,0) = 1/S$$

### 4.5.3 Different Classes of Customers - FCFS

In this section we present an algorithm for models with FCFS discipline at queue 1 and 2 classes of customers with class dependent service times. In subsequent sections we consider models with priority disciplines at queue 1. We assume that queue 2 is such that customers of different classes are served in parallel. In Algorithm 4.6 we assume that the mean service time for each class of customers is exponential with mean $1/\lambda_j$, j=1,2, where j indicates customer class. We assume that the service time at queue 1 is exponentially distributed with mean $1/\mu_j$, j=1,2; extension to non-exponential distributions of the Cox form is straightforward. We will assume that there is exactly 1 customer of class 1 in the model; extension to models where customers may change class when leaving a queue so long as there is at most one class 1 customer at any time is straightforward. Extension to models with more than one customer of each class and more than 2 classes is possible but more difficult. Extension to models with multiple identical servers at queue 1 is straightforward. We represent a state of the model by an ordered pair (i,n) where n is the number of customers in queue 1 and i is the number of class 2 customers at the head of queue 1. When there is no class 1 customer in queue 1, i and n will have the same value. Figure 4.11 gives the state transition diagram for a model with 4 customers.

Algorithm 4.6 Determination of $P(0,0),\ldots,P(0,N-1)$
(assume $N > 1$)

Note that this is the first algorithm we present where the number of states to be finally determined depends on the number of customers. The vectors we deal with have length N. We will represent a column vector with all elements 0 except the ith element 1 as $e_i$.

1. Initialization.

$P(0,0) = e_1$
$P(0,1) = e_2$

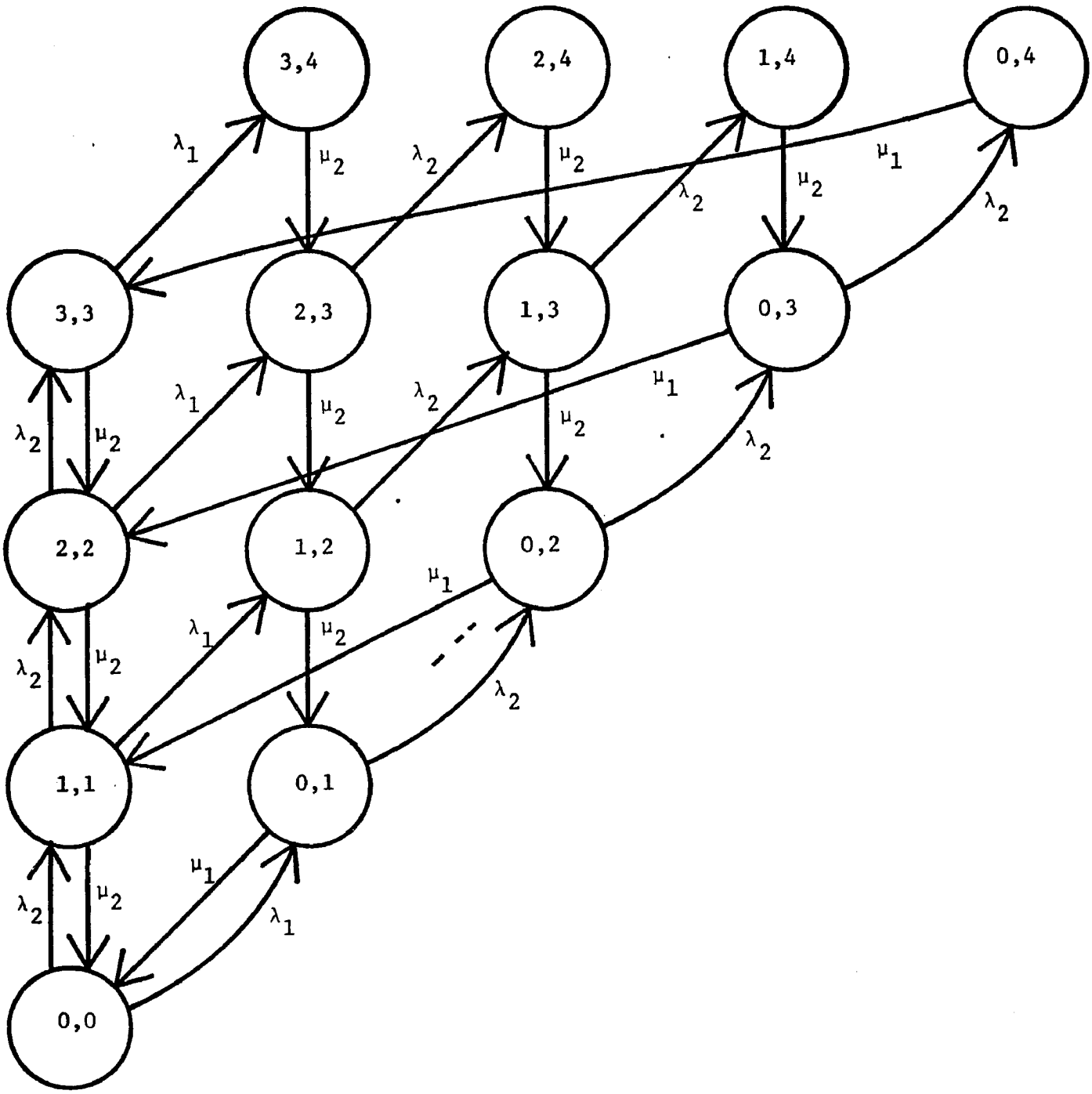



Figure 4.11

$$P(1,1) = \frac{\lambda_1 + \lambda_2}{\mu_2} P(0,0) - \frac{\mu_1}{\mu_2} P(0,1)$$

$$S = P(0,0) + P(0,1) + P(1,1)$$

2. Iteration. Do step 2 for n = 2,...,N-1

   a. $P(0,n) = e_{n+1}$

   b. Iteration. Do step 2b for i = 1,...,n-1

$$P(i,n) = \frac{\lambda_2+\mu_{\min(2,i)}}{\mu_2} P(i-1,n-1) - \frac{\lambda_{\min(2,n-i)}}{\mu_2} P(i-1,n-2)$$

   c. $$P(n,n) = \frac{\lambda_1+\lambda_2+\mu_2}{\mu_2} P(n-1,n-1) - \frac{\lambda_2}{\mu_2} P(n-2,n-2) - \frac{\mu_1}{\mu_2} P(0,n)$$

   d. $S = S + P(0,n) + \ldots + P(n,n)$

3. Determination of P(0,0),...,P(0,N-1) as scalars

   a. $$P(0,N) = \frac{\lambda_1+\mu_2}{\mu_1} P(N-1,N-1) - \frac{\lambda_2}{\mu_1} P(N-2,N-2)$$

   $S = S + P(0,N)$

   b. Iteration. Do step 3b for i = 1,...,N-1

$$P(i,n) = \frac{\lambda_2+\mu_{\min(2,i)}}{\mu_2} P(i-1,N-1) - \frac{\lambda_{\min(2,N-i)}}{\mu_2} P(i-1,N-2)$$

$$D_i = P(i,n) - \frac{\lambda_{\min(2,N-i)}}{\mu_2} P(i,N-1)$$

   $S = S + P(i,n)$

   c. Solve

$$\begin{pmatrix} D_1^T \\ \vdots \\ D_{N-1}^T \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

   For n = 0,...,N-1 $\quad P(0,n) = \zeta_{n+1}$

Our computer implementation uses a similar algorithm which solves for states with all customers in queue 1 and allows queue 1 service distributions of the Cox form with arbitrary numbers of stages. When we allow non-exponential distributions at queue 1, an extended version of Algorithm 4.6 will use considerably less memory and be more efficient than the algorithm we implemented. Our implementation allows $\lambda_1$ and $\lambda_2$ to be dependent on the numbers of customers of each type queue 2.

### 4.5.4 Preemptive Priority Based on Customer Class

Let queue 1 of the two queue models we have been considering have a preemptive priority discipline with class 1 customers having priority over class 2 customers. Let queue 2 have a parallel service discipline as in section 4.5.3. We assume that there are $N_1$ class 1 customers and $N_2$ class 2 customers, and that all service time distributions are exponential with means as before. Extension to more classes of customers is straightforward and extension to non-exponential distributions is also possible. We represent a state of the model by the ordered pair $(n_1,n_2)$ where $n_i$ is the number of class i customers in queue 1. Figure 4.12 gives the state transition diagram for $N_1 = 3$ and $N_2 = 2$.

Algorithm 4.7 Determination of $P(N_1,0),\ldots,P(N_1,N_2)$
(assume $N_1 > 0$, $N_2 > 0$)

Note that the vectors we deal with will have $N_2 + 1$ elements

1. Initialization

   a. For $n_2 = 0,\ldots,N_2$, $P(N_1,n_2) = e_{n_2+1}$

   b. $$P(N_1-1,0) = \frac{\mu_1 + \lambda_2}{\mu_1} P(N_1,0)$$

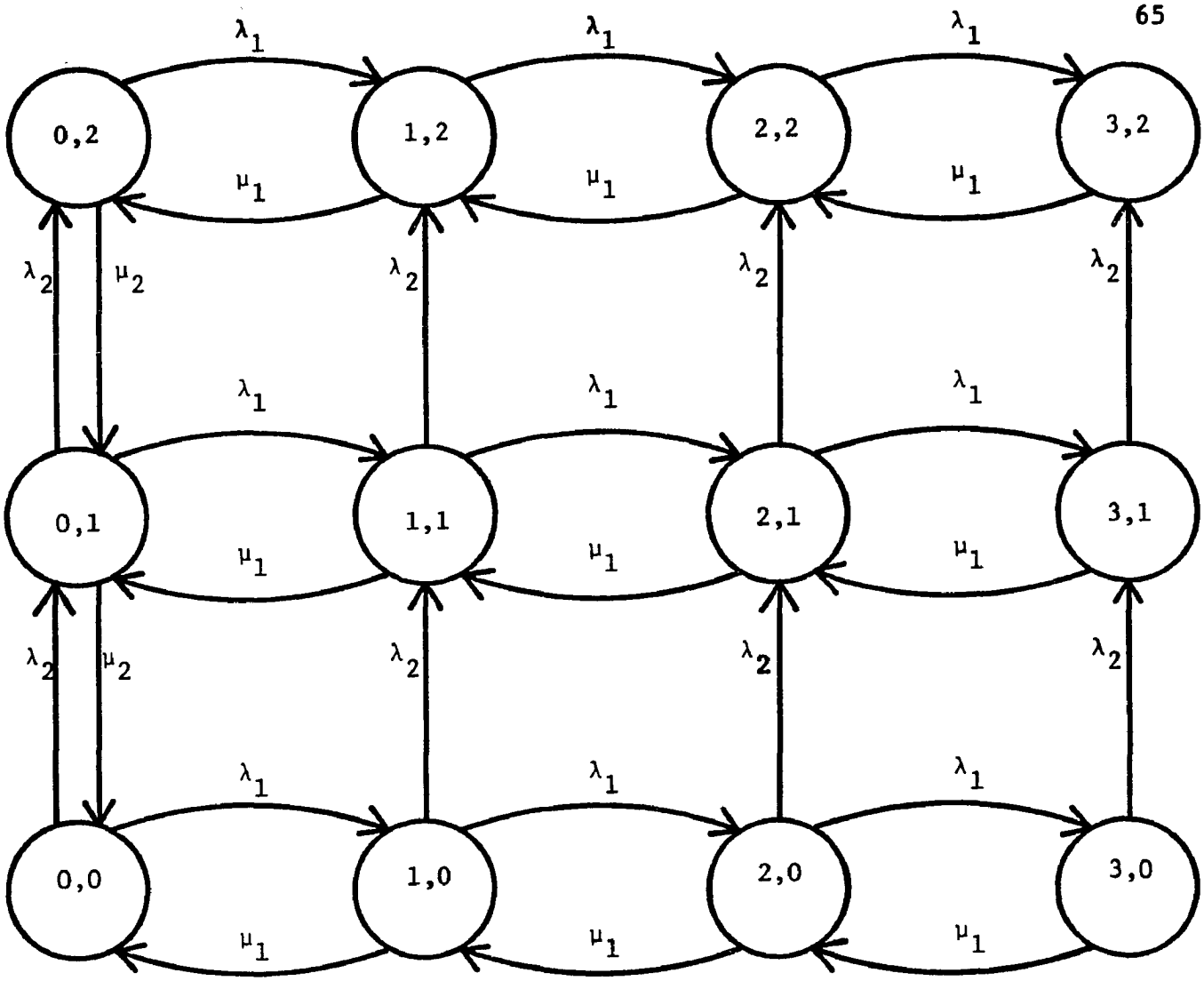



**Figure 4.12**

c. Iteration. Do step 1.c for $n_2 = 1,\ldots,N_2-1$

$$P(N_1-1,n_2) = \frac{\mu_1 + \lambda_2}{\lambda_1} P(N_1,n_2) - \frac{\lambda_2}{\lambda_1} P(N_1,n_2-1)$$

d. $$P(N_1-1,N_2) = \frac{\mu_1}{\lambda_1} P(N_1,N_2) - \frac{\lambda_2}{\lambda_1} P(N_1,N_2-1)$$

e. $$S = \sum_{n_1=N_1-1}^{N_1} \sum_{n_2=0}^{N_2} P(n_1,n_2)$$

2. Iteration. Do step 2 for $n_1 = N_1-2,\ldots,0$

a. $$P(n_1,0) = \frac{\mu_1 + \lambda_2}{\lambda_1} P(n_1+1,0) - \frac{\mu_1}{\lambda_1} P(n_1+2,0)$$

b. Iteration. Do step 2 for $n_2 = 1,\ldots,N_2-1$

$$P(n_1,n_2) = \frac{\mu_1+\lambda_2}{\lambda_1} P(n_1+1,n_2) - \frac{\lambda_2}{\lambda_1} P(n_1+1,n_2-1) - \frac{\mu_1}{\lambda} P(n_1+2,n_2)$$

c. $$P(n_1,N_2) = \frac{\mu_1}{\lambda_1} P(n_1+1,n_2) - \frac{\lambda_2}{\lambda_1} P(n_1+1,n_2-1) - \frac{\mu_1}{\lambda} P(n_1+2,n_2)$$

d. $$S = S + \sum_{n_2=0}^{N_2} P(n_1,n_2)$$

3. Determination of $P(N_1,0),\ldots,P(N_1,N_2)$ as scalars

a. Iteration. Do step 3.a for $n_2 = 1,\ldots,N_2-1$

$$D_{n_2} = P(0,n_2) - \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_2} P(0,n_2-1) - \frac{\mu_1}{\lambda_1+\lambda_2+\mu_2} P(1,n_2)$$

$$- \frac{\mu_2}{\lambda_1+\lambda_2+\mu_2} P(0,n_2+1)$$

b. $$D_{N_2} = P(0,N_2) - \frac{\lambda_2}{\lambda_1+\mu_2} P(0,N_2-1) - \frac{\mu_1}{\lambda_1+\mu_2} P(1,N_2)$$

c. Solve $\begin{pmatrix} D_1^T \\ \vdots \\ D_{N_2}^T \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$

d. For $n_2 = 0,\ldots,N_2$ $\quad P(N_1,n_2) = \zeta_{n_2+1}$

Our computer implementation of an algorithm based on 4.7 allows three classes of customers and assumes exponential distributions. The values of $\lambda_i$, $i = 1,2,3$, may be dependent on the numbers of customers of each class in queue 2. Our algorithm decides which states to solve for on the basis of minimizing the length of the vectors used; it solves for the states with $n_i = N_i$, where i is the minimum value such that $n_i = \max(n_1,n_2,n_3)$.

### 4.5.5 Non-Preemptive Priority Based on Customer Class

We now consider models similar to those considered in 4.5.4, but with non-preemptive priority at queue 1. The state of the model is represented by an ordered triple $(i,n_1,n_2)$ where i is the class of the customer being served at queue 1 and $n_1$ and $n_2$ are as before. (Let $i = 2$ when $n_1 = n_2 = 0$). Figure 4.13 gives the state transition diagram for $N_1 = 3$ and $N_2 = 2$.

Algorithm 4.8 Determination of $P(1,N_1,0),P(1,N_1,1),P(1,N_1,2),\ldots,P(1,N_1N_2)$, $P(2,N_1,N_2)$
(assume $N_1 > 1$, $N_2 > 0$)

Note the vectors we deal with have length $2N_2+1$

1. Initialization.

a. $P(1,N_1,0) = e_1$ $\quad$ For $n_2 = 1,\ldots,N_2$,

For $i = 1,2$, $\quad P(i,N_1,n_2) = e_{2n_2+i-1}$

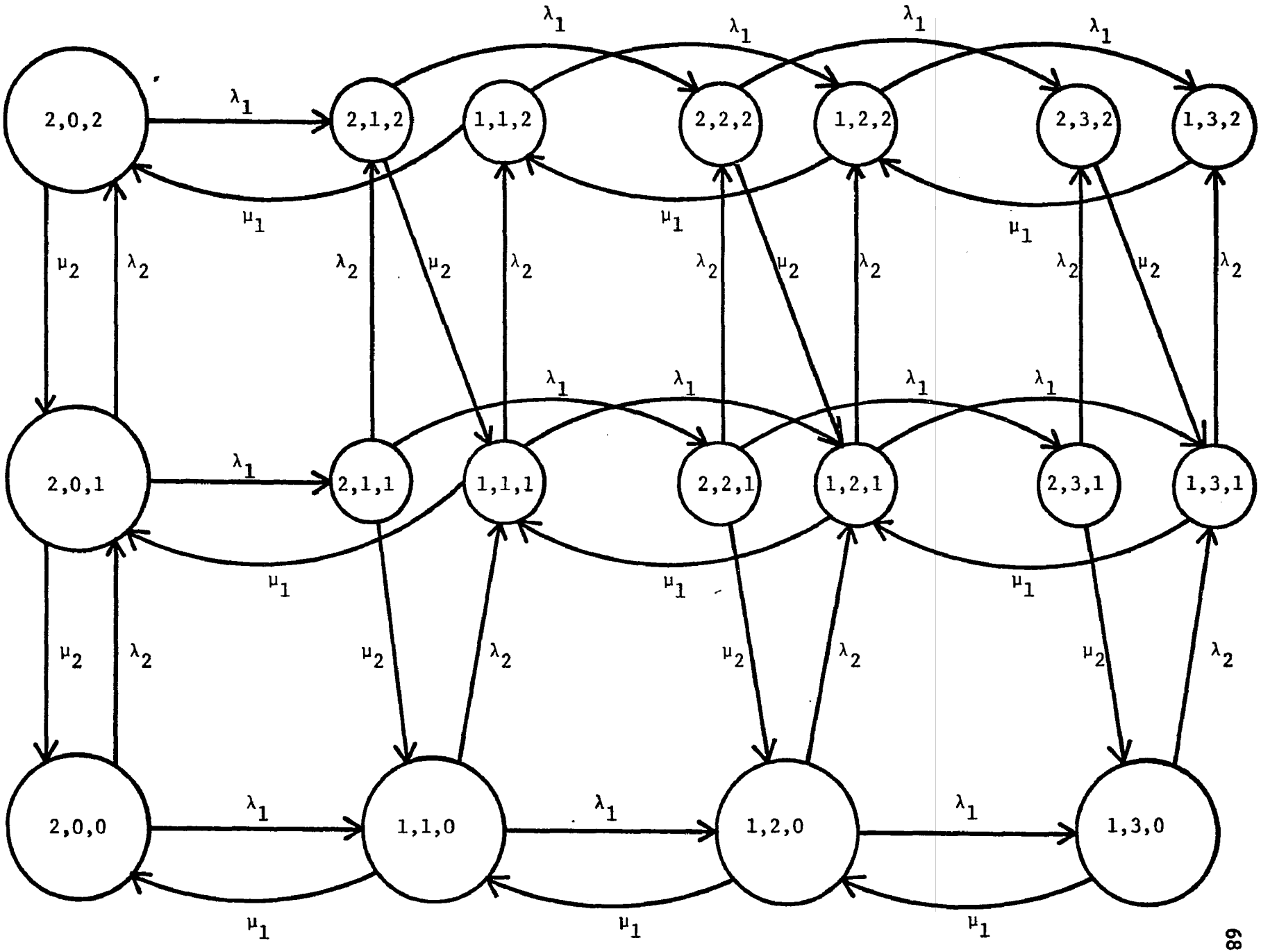



Figure 4.13

b. $$P(1,N_1-1,0) = \frac{\lambda_2+\mu_1}{\lambda_1} P(1,N_1,0) - \frac{\mu_2}{\lambda_1} P(2,N_1,1)$$

c. Iteration. Do step 1.c for $n_2 = 1,N_2-1$

$$P(1,N_1-1,n_2) = \frac{\lambda_2+\mu_1}{\lambda_1} P(1,N_1,n_2) - \frac{\mu_2}{\lambda_1} P(2,N_1,n_2+1) - \frac{\lambda_2}{\lambda_1} P(1,N_1,n_2-1)$$

$$P(2,N_1-1,n_2) = \frac{\lambda_2+\mu_2}{\lambda_1} P(2,N_1,n_2) - \frac{(1-\delta_{1,n_2})\lambda_2}{\lambda_1} P(2,N_1,n_2-1)$$

where $\delta$ is the Kronecker $\delta$.

d. $$P(1,N_1-1,N_2) = \frac{\mu_1}{\lambda_1} P(1,N_1,N_2) - \frac{\lambda_2}{\lambda_1} P(1,N_1,N_2-1)$$

$$P(2,N_1-1,N_2) = \frac{\mu_2}{\lambda_1} P(2,N_1,N_2) - \frac{(1-\delta_{1,N_2})\lambda_2}{\lambda_1} P(2,N_1,N_2-1)$$

e. $$S = P(1,N_1,0) + P(1,N_1-1,0) = \sum_{i=1}^{2} \sum_{n_1=N_1-1}^{N_1} \sum_{n_2=1}^{N_2} P(i,n_1,n_2)$$

2. Iteration. Do step 2 for $n_1 = N_1-2,\ldots,1$

a. $$P(1,n_1,0) = \frac{\lambda_1+\lambda_2+\mu_1}{\lambda_1} P(1,n_1+1,0) - \frac{\mu_2}{\lambda_1} P(2,n_1+1,1) - \frac{\mu_1}{\lambda_1} P(1,n_1+2,0)$$

b. Iteration. Do step 2 for $n_2 = 1,\ldots,N_2-1$

$$P(1,n_1,n_2) = \frac{\lambda_1+\lambda_2+\mu_1}{\lambda_1} P(1,n_1+1,n_2) - \frac{\mu_2}{\lambda_1} P(2,n_1+1,n_2+1) - \frac{\lambda_2}{\lambda_1} P(1,n_1+1,n_2-1) - \frac{\mu_1}{\lambda_1} P(1,n_1+2,n_2)$$

$$P(2,n_1,n_2) = \frac{\lambda_1+\lambda_2+\mu_2}{\lambda_1} P(2,n_1+1,n_2) - \frac{(1-\delta_{1,n_2})\lambda_2}{\lambda_1} P(2,n_1+1,n_2-1)$$

c. $$P(1,n_1,N_2) = \frac{\lambda_1+\mu_1}{\lambda_1} P(1,n_1+1,N_2) - \frac{\lambda_2}{\lambda_1} P(1,n_1+1,N_2-1)$$

$$- \frac{\mu_1}{\lambda_1} P(1,n_1+2,n_2)$$

$$P(2,n_1,N_2) = \frac{\lambda_1+\mu_2}{\lambda_1} P(2,n_1+1,N_2) - \frac{(1-\delta_{1,N_2})\lambda_2}{\lambda_1} P(2,n_1+1,N_2)$$

d. $$S = S + P(1,n_1,0) + \sum_{i=1}^{2} \sum_{n_2=1}^{N_2} P(i,n_1,n_2)$$

3. Determination of $P(1,N_1,0),\ldots,P(2,N_1,N_2)$ as scalars

a. Iteration. Do step 3.a for $n_2 = 1,\ldots,N_2-1$

$$D_{n_2} = P(1,1,n_2) - \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_1} P(1,1,n_2-1) - \frac{\mu_2}{\lambda_1+\lambda_2+\mu_1} P(2,1,n_2+1)$$

$$- \frac{\mu_1}{\lambda_1+\lambda_2+\mu_1} P(1,2,n_2)$$

b. $$D_{N_2} = P(1,1,N_2) - \frac{\lambda_2}{\lambda_1+\mu_1} P(1,1,N_2-1) - \frac{\mu_1}{\lambda_1+\mu_1} P(1,2,n_2)$$

c. $$P(2,0,0) = \frac{\lambda_1+\lambda_2+\mu_1}{\lambda_1} P(1,1,0) - \frac{\mu_2}{\lambda_1} P(2,1,1) - \frac{\mu_1}{\lambda_1} P(1,2,0)$$

d. Iteration. Do step 3.d for $n_2 = 1,\ldots,N_2-1$

$$P(2,0,n_2) = \frac{\lambda_1+\lambda_2+\mu_2}{\lambda_1} P(2,1,n_2) - \frac{(1-\delta_{1,n_2})\lambda_2}{\lambda_1} P(2,1,n_2-1)$$

e. $$P(2,0,N_2) = \frac{\lambda_1+\mu_2}{\lambda_1} P(2,1,N_2) - \frac{(1-\delta_{1,N_2})\lambda_2}{\lambda_1} P(2,1,N_2-1)$$

f. $$S = S + \sum_{n_2=0}^{N_2} P(2,0,n_2)$$

g. Iteration. Do step 3.g for $n_2 = 1,\ldots,N_2-1$

$$D_{N_2+n_2} = P(2,0,n_2) - \frac{\lambda_2}{\lambda_1+\lambda_2+\mu_2} P(2,0,n_2-1) - \frac{\mu_2}{\lambda_1+\lambda_2+\mu_2} P(2,0$$

$$- \frac{\mu_1}{\lambda_1+\lambda_2+\mu_2} P(2,1,n_2)$$

h. $$D_{2N_2} = P(2,0,N_2) - \frac{\lambda_2}{\lambda_1+\mu_2} P(2,0,N_2-1) - \frac{\mu_1}{\lambda_1+\mu_2} P(1,1,N_2)$$

i. Solve $$\begin{pmatrix} D_1^T \\ \vdots \\ D_{2N_2}^T \\ \\ S^T \end{pmatrix} x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \\ 1 \end{pmatrix}$$

$P(1,N_1,0) = \zeta_1$

for $n_2 = 1,\ldots,N_2$, for $i = 1,2$, $P(i,N_1,n_2) = \zeta_{2n_2+i-1}$

The algorithm we implemented allows three classes of customers, and allows queue length dependent service times for queue 2.

### 4.5.6 Other Applications

We have applied these techniques to two queue models with random scheduling at queue 1 or with priority disciplines at both queues, and to models with more than two queues, but have not implemented computer programs for these models. Models with priority disciplines at both queues seem limited in applicability to computer systems. Algorithms for more than two queues will tend to have large memory requirements, but may still be of some value.

## 4.6 Application to Computer System Modeling

### 4.6.1 General Approaches

Many approximate analysis techniques for general models of computing systems are dependent on solution of the models we have considered (C2, Chapter V). We now consider direct use of these models in computer system analysis. The models used here are based on those of Gaver (G1).

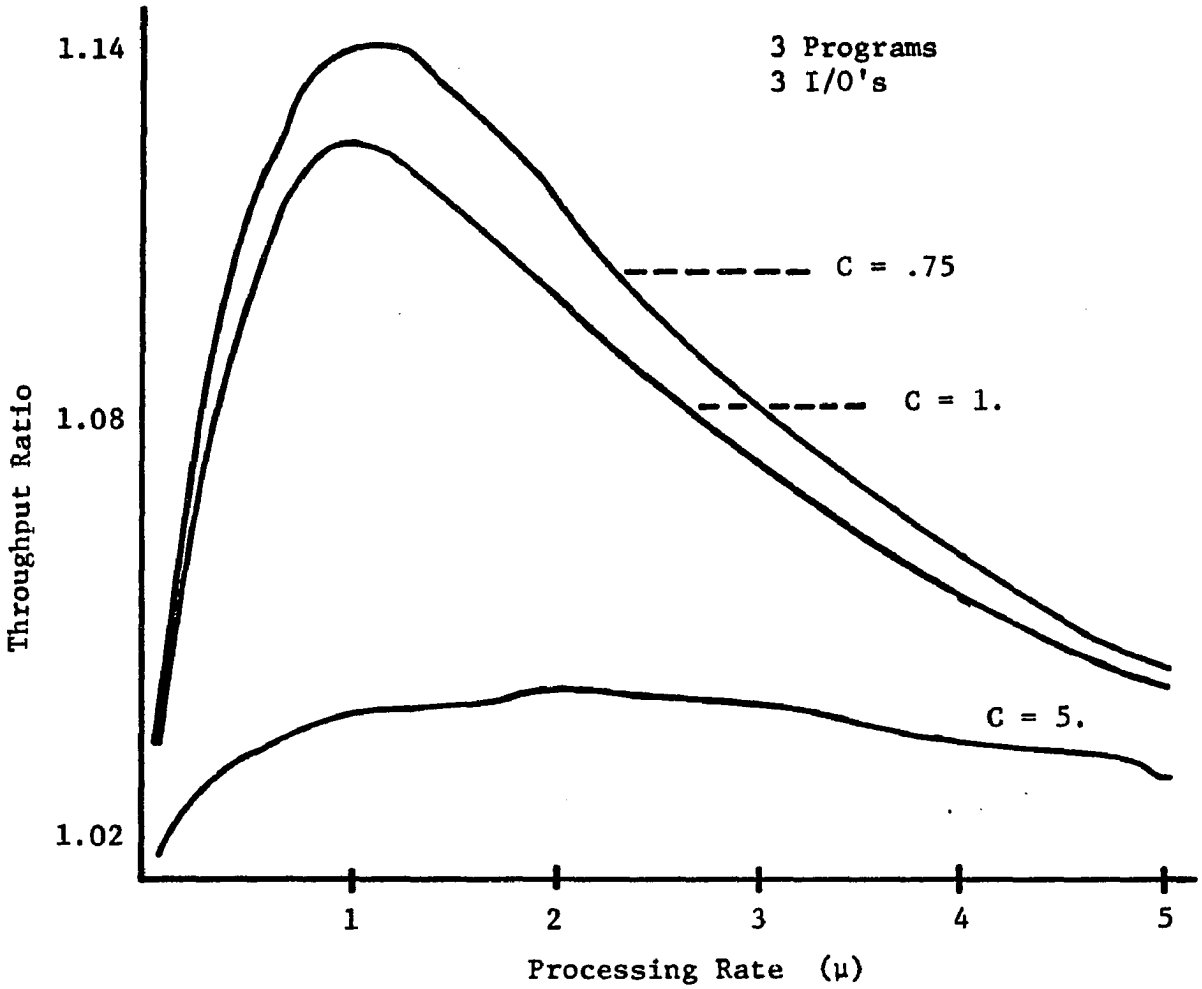
We can use these queueing network models to help illustrate some of the effects of having single or multiple processing units, and of having multitasking when there are multiple processing units. We will restrict attention to a simple class of models; the analysis can be extended to much more general models using the techniques of (B4,C2,Chapter V). We will assume that there is a fixed number of programs in memory and that these programs have identical behavior. The programs alternately request service from a central processing unit (CPU) and an input/output (I/O) device. When there are no free CPU's, or no free I/O's, programs must wait in the respective CPU and I/O queues, both with FCFS queueing discipline. We assume the service times at the I/O devices are exponentially distributed with mean L, where L is the number of I/O devices, and that the CPU service times have a standardized distribution (Chapter V) with mean $1/\mu$ and coefficient of variation C.

The queueing network models we use are those of sections 4.4 and 4.5.2. We let queue 1 represent the CPU queue and let queue 2 represent the I/O queue. For queue 2 we let $\lambda$ be a function of the length of queue 2, with $\lambda_i = \min(i,L)/L$, $i = 1,\ldots,N$, where N is the number of programs. We use this representation so that the effective combined rate of the I/O devices will be 1 when all devices are busy.

In Section 4.6.2 we consider the relative advantages of having a single CPU of given speed and of having two CPU's half as fast as the single processor, in section 4.6.3 we study the improvement in throughment obtained by multitasking with two processors, and in section 4.6.4 we consider improvements in throughput which may be obtained by adding or upgrading CPU's.

### 4.6.2 Single CPU vs. Two Slower CPU's

It would be reasonable to expect that a single CPU would be better than two CPU's with half the processing rate of the single CPU, since one of the slower CPU's in the two CPU case would be idle when there is only a single program needing a CPU. Figure 4.14 shows the ratio of throughput with one CPU to the throughput with two CPU's half as fast as a function of $\mu$ for the fast CPU. Curves for three values of C are given. The number of customers and the number of I/O's are both held constant at 3. Notice that there is little difference between one fast CPU and two slower CPU's when the CPU distribution is skewed. When the distribution is so skewed, a single CPU may be occupied for long periods by a single program; the other programs must wait in the CPU queue while the I/O's are idle. In contrast, with multiple CPU's programs with smaller requests for service can continue to circulate through the system. As C decreases, the ratio increases because this blocking effect decreases. The blocking effect becomes strong enough to favor two slower CPU's when the number of programs and I/O's is raised to 5 (Figure 4.15), or when there is contention for I/O devices (Figure 4.16). Figure 4.17 shows throughput as a function of the number of programs. Curves are shown for 1 and 2 CPU's. L is the same as the number of programs, $\mu$ for the fast CPU is 1, $\mu$ for the 2 slow CPU's is .5, and C is 5.




C - Coefficient of Variation

Figure 4.14 - Throughput Ratio - 1 Fast/2 Slow CPU's

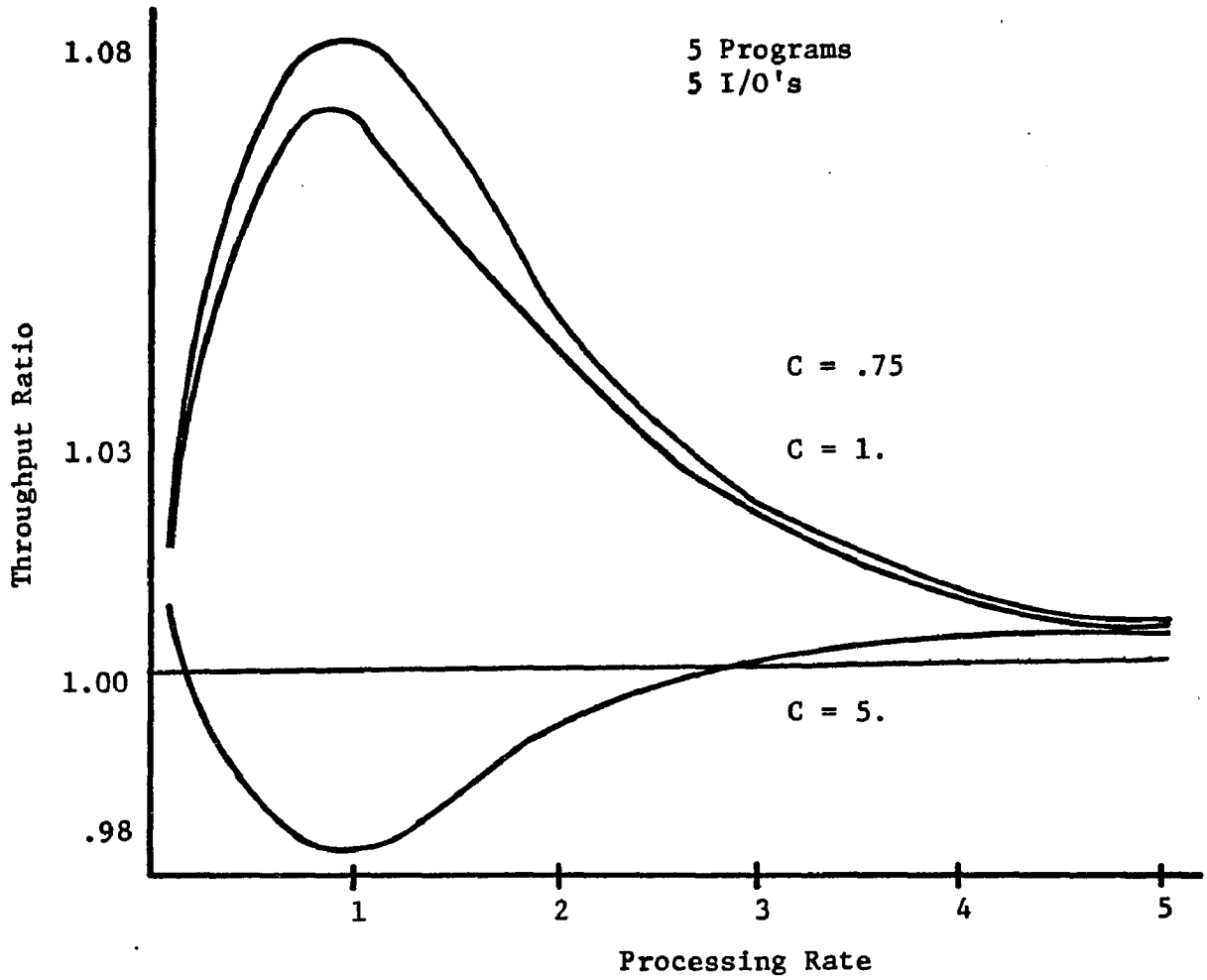



Figure 4.15 - Throughput Ratio - 1 Fast/2 Slow CPU's

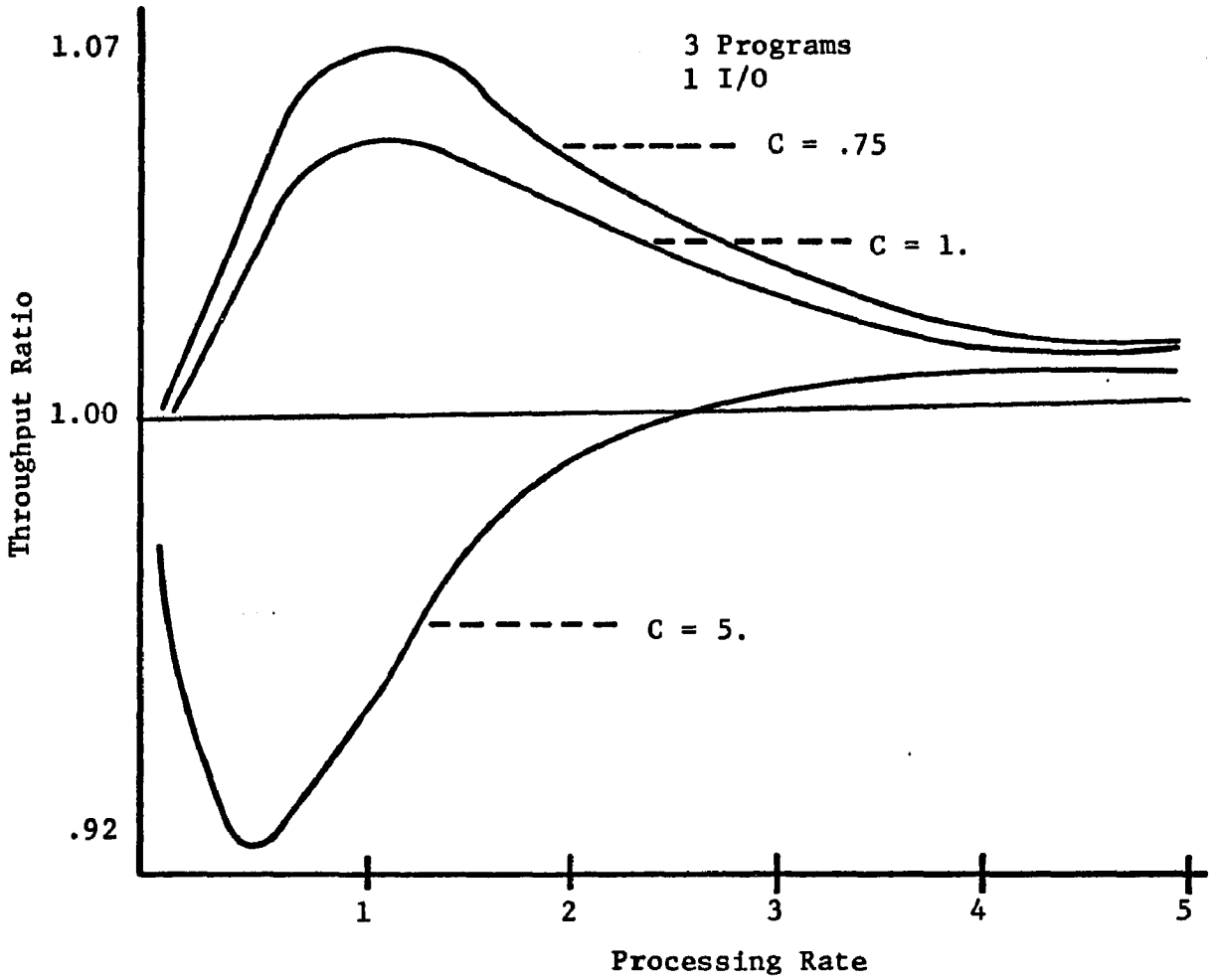



Figure 4.16 - Throughput Ratio - 1 Fast/2 Slow CPU's

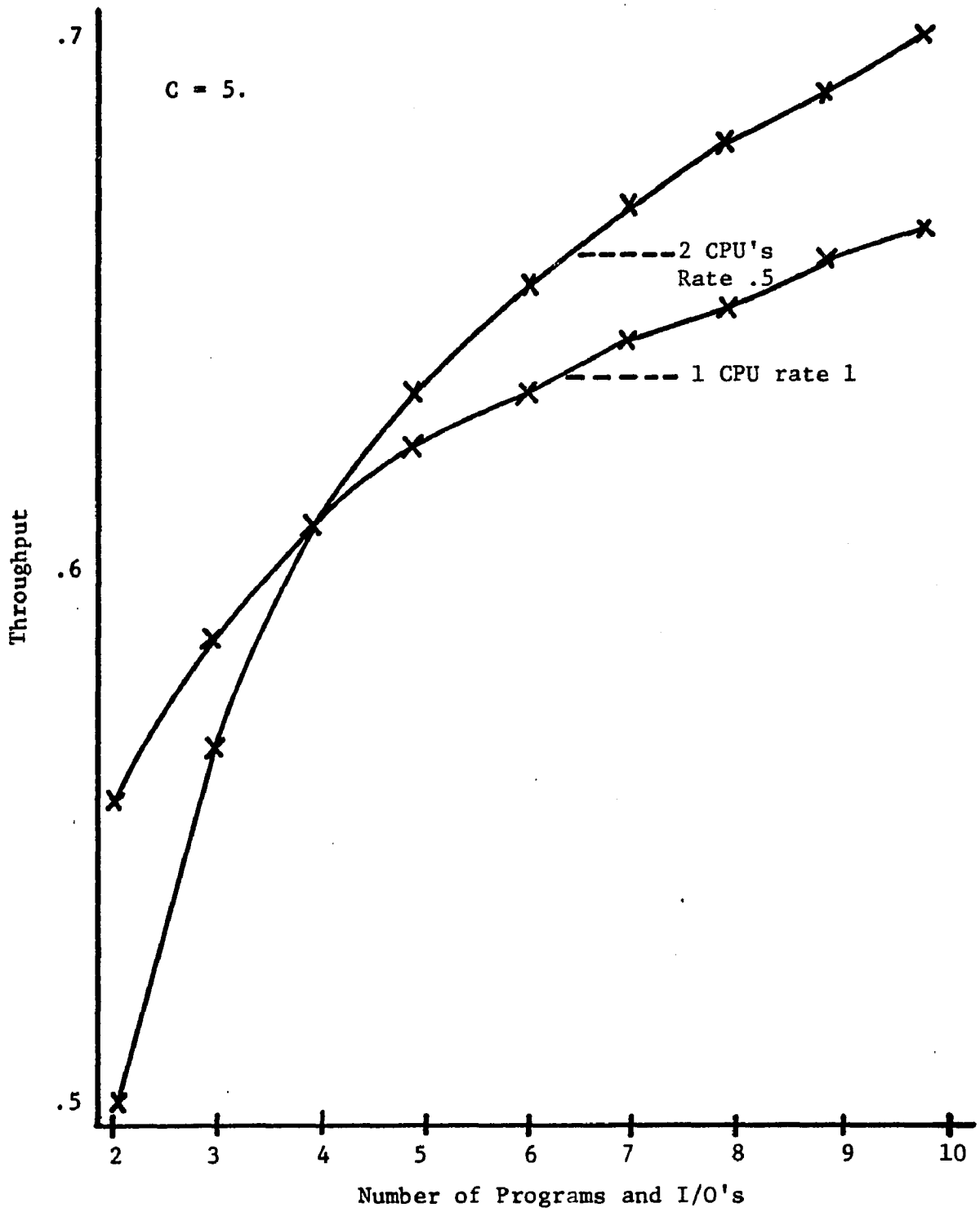



Figure 4.17 - Throughput with Skewed CPU Distribution

### 4.6.3 Improvement Obtained by Multitasking

When a system has more than one CPU, it may be possible to improve performance by dividing a program into tasks which may execute in parallel (H3). Usually there will be interference between the tasks, but for the sake of illustration we will assume that it is possible for two processors to cooperate fully on a single program without interference. If processors always cooperate on a single program without interference, then this is equivalent to a single CPU with rate equal to the combined rate of the individual CPU's. The analysis of Sec. 4.6.2 would be approximate for the case. We assume that processors cooperate only when there is exactly one program needing a CPU. Figure 4.18 shows the ratio of throughput with and without cooperation as a function of $2\mu$. Curves are shown for the same distributions as before, for a system with three programs and three I/O's. Figure 4.19 gives results for a system with 5 programs and 5 I/O's. Notice that the potential improvement is less; since there are more programs, it is less likely that there will be only one program needing the CPU. Also notice in both figures that maximum potential improvement exists when the system is fairly well balanced. When the system is CPU bound, it is unlikely that only one program will need a CPU; when the system is I/O bound, improvement in CPU performance has little overall effect.

### 4.6.4 Improvement Obtained by Adding or Upgrading CPU's

In general, the rates of CPU's actually obtainable do not increase continuously, but in discrete steps. For the sake of example, we assume that we have a choice of four CPU's with mean processing rates ($\mu$) 1/3, 1, 3, and 9, that a system can support at most 2 CPU's, and that CPU's of different

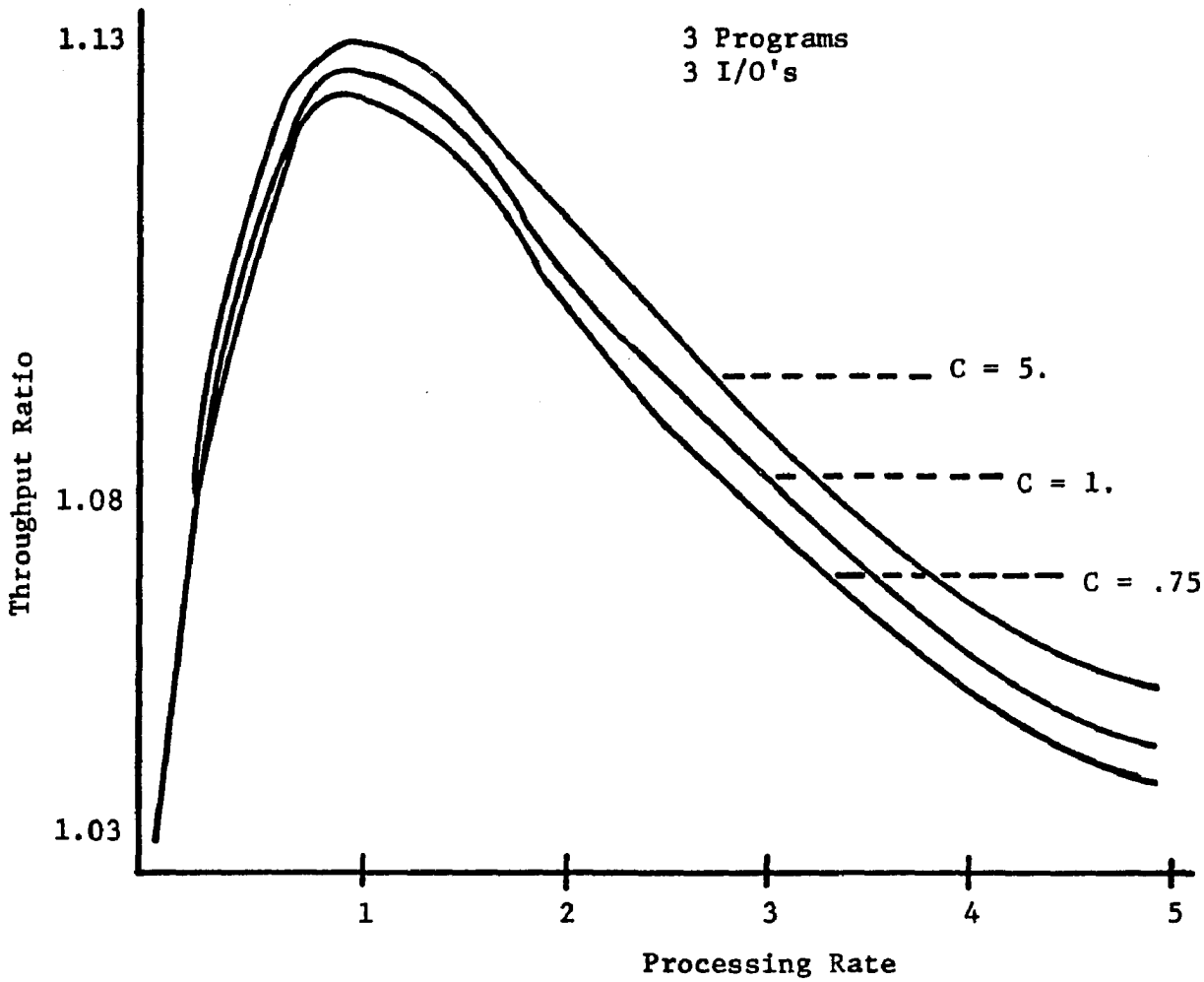



Figure 4.18 - Throughput Ratio - Multi-tasking/Uni-tasking

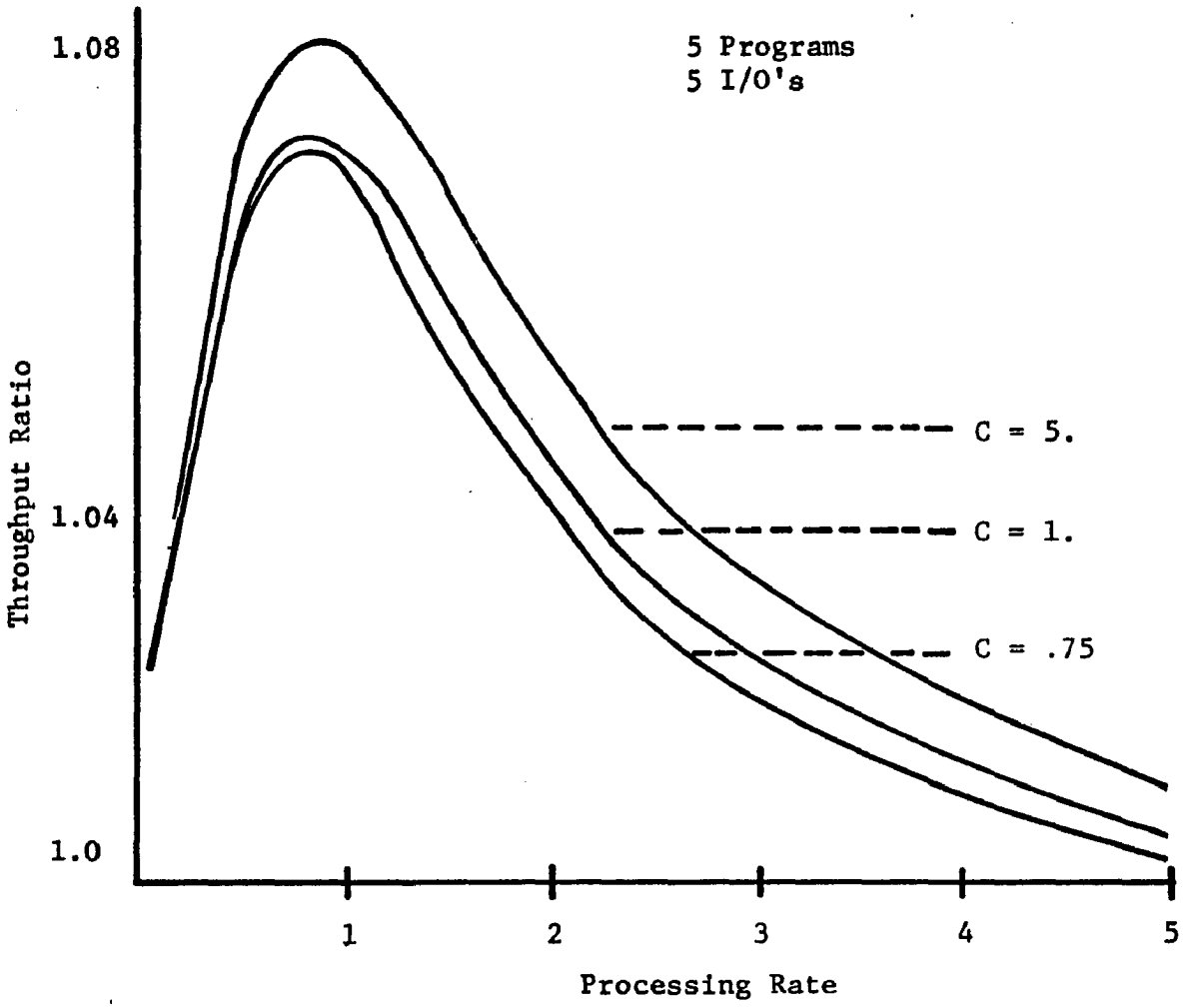



Figure 4.19 - Throughput Ratio - Multi-tasking/Uni-tasking

rates may not be used together. Thus a system with a single CPU may increase processing power by adding a similar CPU, but a system with dual CPU's must increase power by getting a faster CPU. We would expect that throughput would increase monotonically as potential processing power is increased, as in Figure 4.20. In this figure we have 3 programs, 3 I/O's, and C is 5. Some unexpected behavior does occur under other circumstances. Figure 4.21 is for a system with five customers and one I/O. Replacing two CPU's of rate 3 with one of rate 9 actually decreases throughput; this can be explained by the effect of the skewed distribution as with Figures 4.15 and 4.16. Figure 4.22 is for a system with 5 customers, 5 I/O's, and C = .75. Adding a second CPU of rate 3 or rate 9 produces negligible improvement because it is unlikely that more than one program will need the CPU at the same time. These anomalies are not of great significance; since the system is so I/O bound where the anomalies occur, it is unlikely that one would try to improve system performance by increasing CPU power.

### 4.6.5 Summary of Model Results

These models suggest several areas of consideration in choice of CPU's and CPU Schedulers. From section 4.6.2 we see that the choice of the number of processors depends heavily on the coefficient of variation of the distribution of CPU service requests. If several CPU's are used, the system is well-balanced, the programs can be divided into non-interfering tasks, and the coefficient of variation of the CPU service distribution is small, then it may be advantageous to have the CPU's cooperate on a single program rather than work on separate programs. From section 4.6.3 we see that multi-tasking may also be desirable when only a single program needs a CPU, if the system

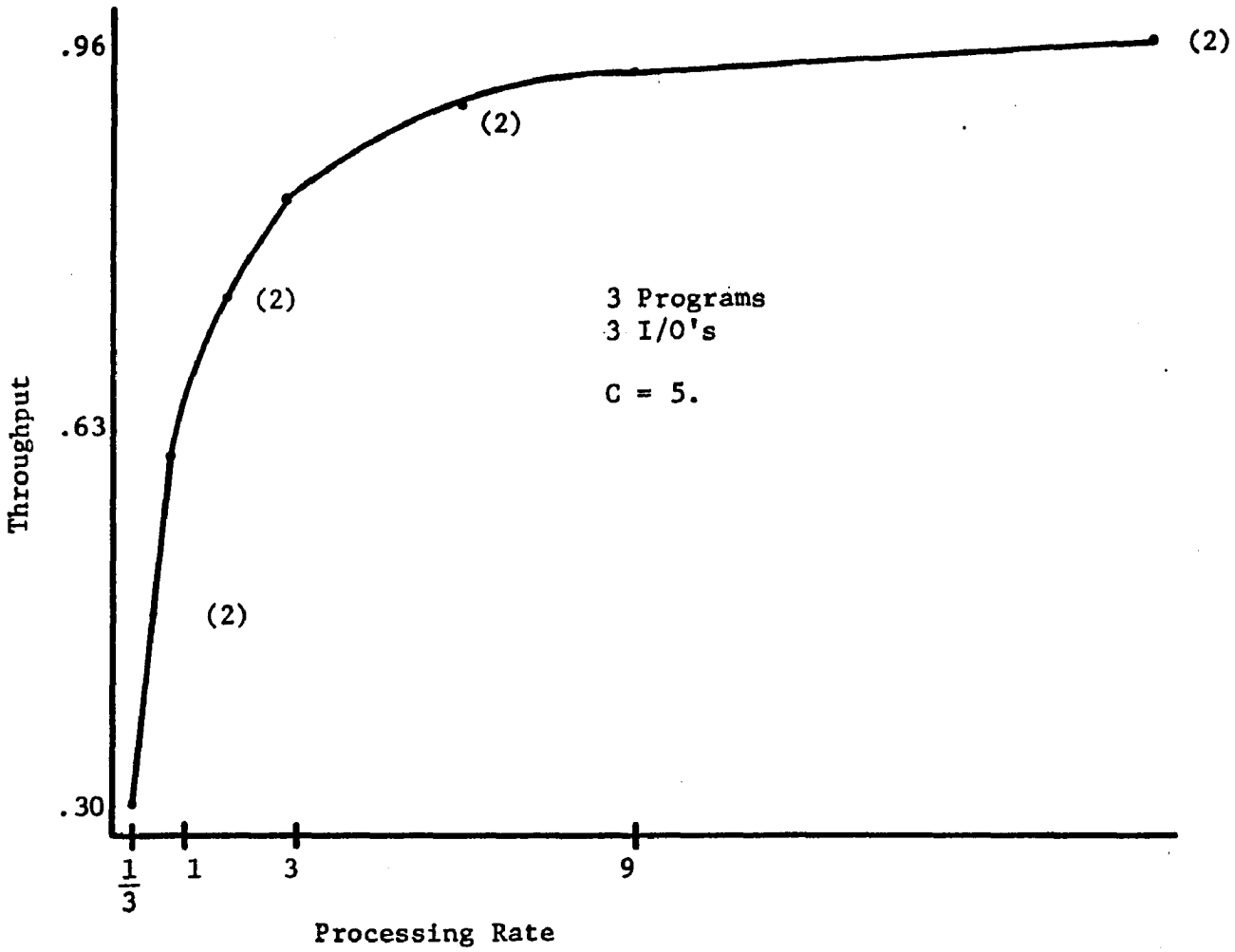



Figure 4.20 - Throughput Obtained by Upgrading CPU's

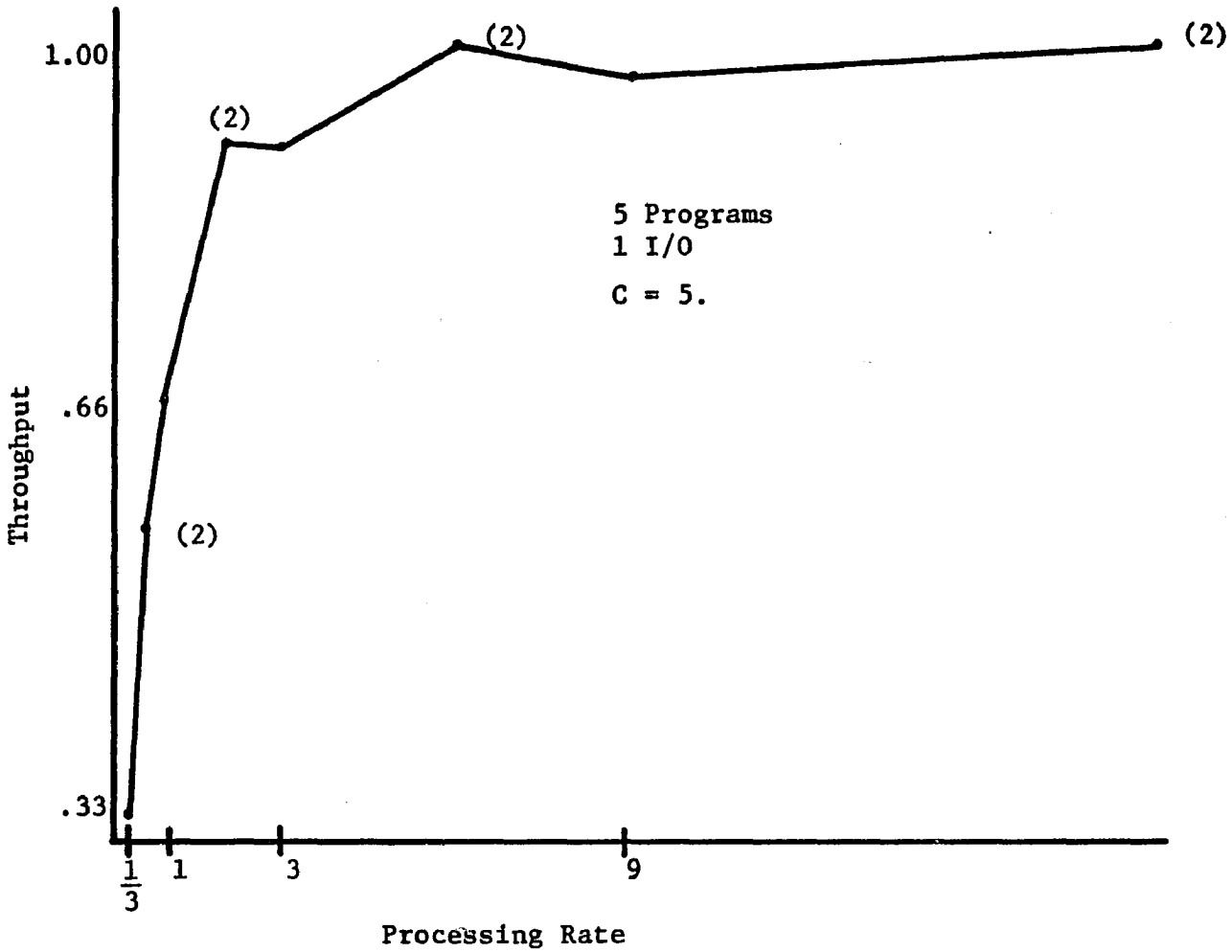



Figure 4.21 - Throughput Obtained by Upgrading CPU's

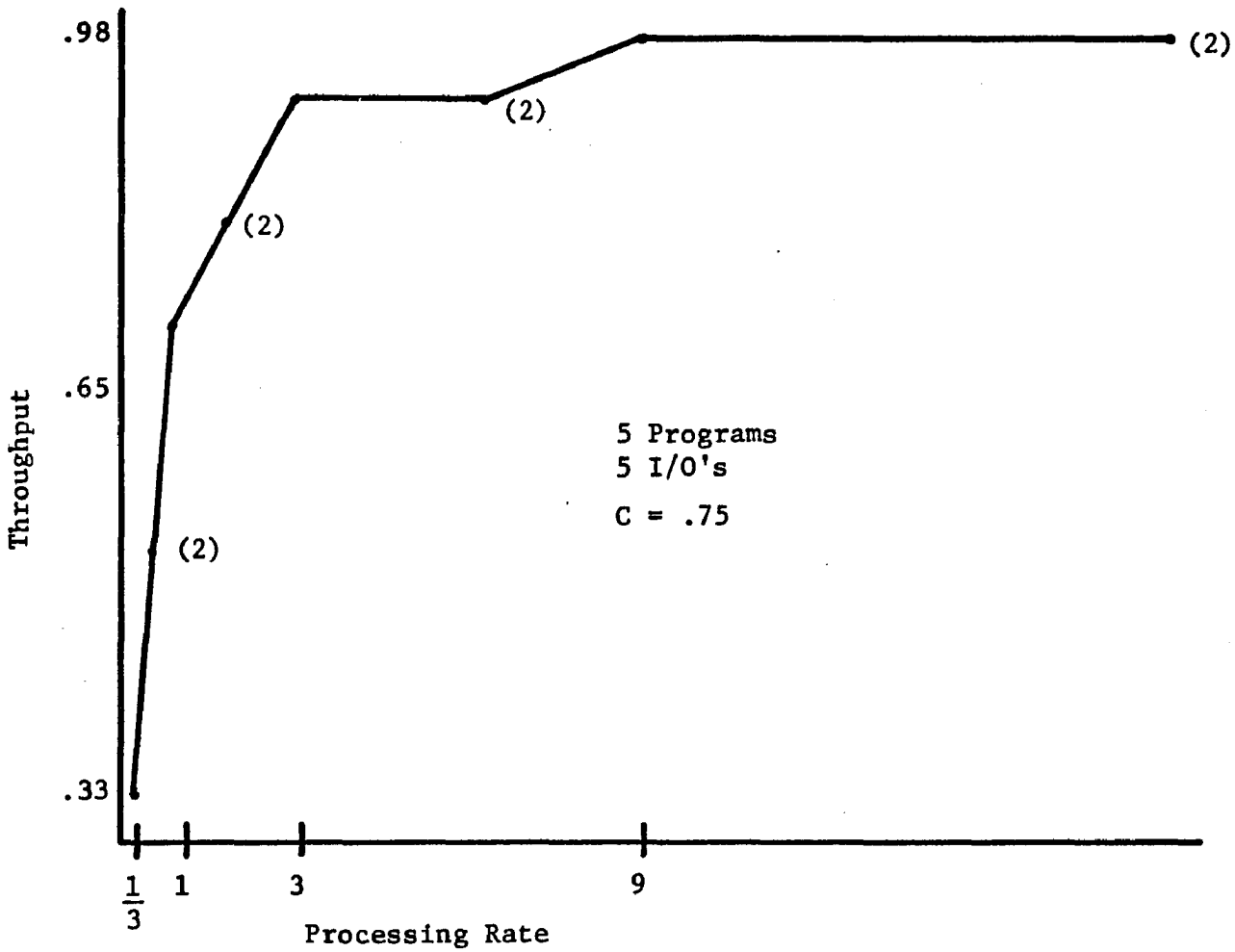



Figure 4.22 - Throughput Obtained by Upgrading CPU's

is well balanced and programs may be divided into non-interfering tasks. Finally, from section 4.6.4 we find that the models of increasing CPU power agree with intuition as long as the system is not severely I/O bound.

# CHAPTER V

# APPROXIMATE ANALYSIS OF CENTRAL SERVER MODELS

## 5.1 Introduction

Central server queueing network models have been widely used in the analysis of computing systems (B1,B6,G2,L2,S4,S5). These models assume that a fixed number of customers (programs) traverse a closed network consisting of the central processor (CPU) and the input/output (I/O) devices. A customer alternately receives service from the CPU and one of the I/O devices. A customer may have to wait in a queue if the server is busy. After completing service at the CPU, a customer selects an I/O device according to probabilities associated with that device and the given customer. These probabilities are independent of the state of the system. The service time of a customer on a device may depend upon the device, the customer, and the queue lengths for that device, but is otherwise independent of the state of the system. Figure 5.1 illustrates a central server model with three I/O devices. Central server models have also been used as sub-models in detailed models of complex systems (B4).

Often the models used are such that solutions for the equilibrium behavior can be determined using the techniques of local balance (B2,C1). If the model is to have first come first served (FCFS) queueing disciplines, and if the techniques of local balance are to be used in the solution of the model, then it must be assumed that, at the servers with FCFS disciplines, the service distributions are exponential and independent of the customer being served. Local balance techniques do not allow priority queueing disciplines.

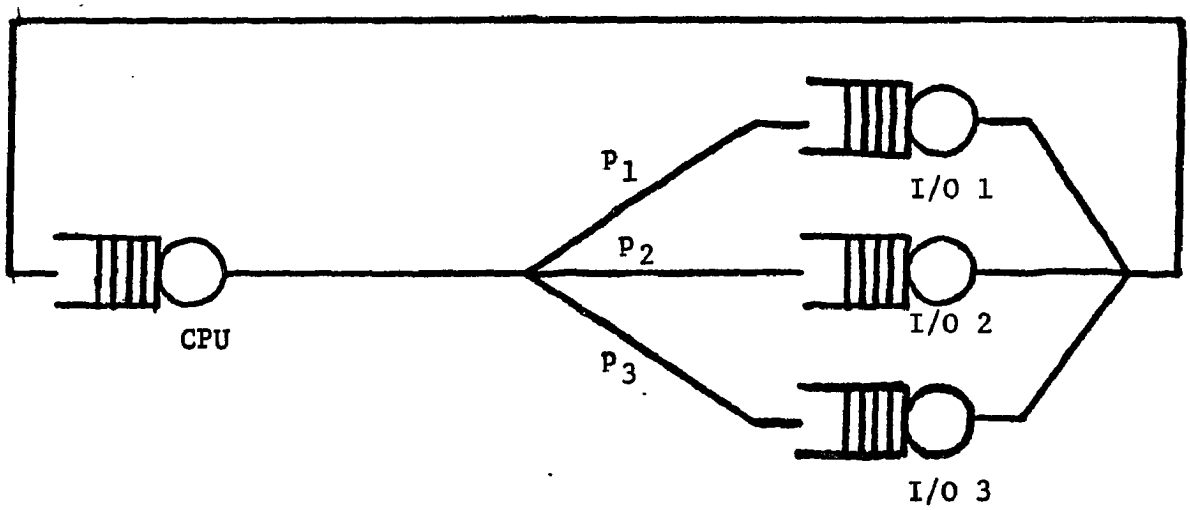



Figure 5.1

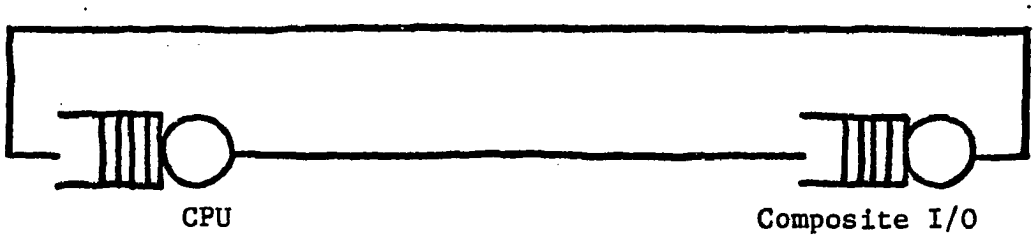



Figure 5.2

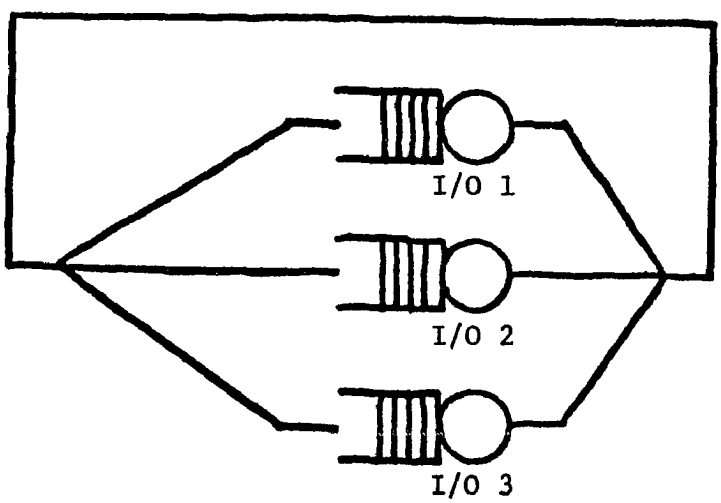



Figure 5.3

Empirical studies on real computing systems show that CPU service distributions are often hyper-exponential (the standard deviation is greater than the mean) and that I/O device service distributions may be hypo-exponential (the standard deviation is less than the mean). Studies (J2) have shown that mean service times and service distributions are dependent on the customer being served. When one makes assumptions that distributions are exponential and all customers have the same distributions, significant inaccuracy may be introduced into the model. Clearly, distinctions must be made between customers if priority CPU distributions are considered. Therefore, (a) many realistic problems do not satisfy local balance and (b) customer differentiation is often required for realistic models.

Chandy, Herzog and Woo (C2) have developed accurate approximate iterative techniques for analysis of general queueing networks with non-exponential service distributions and distributions dependent on customer class. The iterative techniques of Wallace and Rosenberg (W1) may also be used to obtain exact solutions for models with non-exponential distributions. The techniques of Crane and Iglehart (C6,C7) may be used to obtain confidence intervals for simulation results for these models and thus to obtain accurate simulation results. However, these techniques are relatively expensive to apply. In many instances it will not be practical to survey a large variety of models using these techniques.

We present here approximate solution techniques specifically intended for, but not limited to, central server models of computing systems. Our techniques are considerably less expensive to apply than the above mentioned techniques, but are sufficiently accurate for the initial stages of computer

system design. Our techniques complement the previous techniques in that ours can be used to study and compare a large variety of models, and then more accurate, more expensive techniques may be used, to study more carefully, a small subset of the original group of models.

Section 5.2 summarizes central server models in local balance and gives examples of inaccuracies of "local balance assumptions." Section 5.3 describes "Norton's Theorem" on locally balanced queueing networks (C3) as applied to central server models. Our approximations are based on the results of Norton's Theorem. Section 5.4 presents the approximations for models with non-exponential distributions, Section 5.5 presents techniques for class dependent service distributions, and Section 5.6 presents techniques for models with priority CPU disciplines based on customer class. In Section 5.7 we compare the results of our techniques with results of simulations; our techniques are validated by comparison with over 125 different simulations.

## 5.2 Local Balance

A central server model will be in local balance (B2) if 1) branching probabilities are dependent only on the device and the customer class, 2) all queueing disciplines are FCFS, processor sharing (PS) or last come first served preemptive resume (LCFSPR), 3) servers with FCFS discipline have exponential distributions independent of customer class (which may depend on queue length), and 4) servers with PS or LCFSPR disciplines have differentiable service distributions (which may be dependent on customer class). In these models the equilibrium state probabilities will have the "product form," and are easily calculated (B2). From the state probabilities one can determine model statistics such as throughput, server utilization, queue length distributions and waiting time distributions.

The following example illustrates the inaccuracy which may be introduced by using local balance solutions for models violating local balance assumptions. This example is by no means a worst case, but illustrates that results of assuming local balance are likely to be unsatisfactory.

Suppose that a system to be modeled has one I/O device and two classes of customers, with one customer per class. Further, both service disciplines are FCFS, all service distributions are exponential, the mean CPU service time for class one is 2, the mean CPU service time for class two is .2, and the mean I/O service time for both classes is 1. Suppose we are interested in the overall throughput of customers through the CPU. This model is small enough that exact solution of the Markov balance equations is convenient. From the solution of these equations the throughput is .5941. If we assume that the results for a similar model with PS CPU discipline will be close enough, the value we get for throughput will be .84, an error of more than 40%. If we apply the techniques of Section 5, the value we get for throughput is .6375, an error of about 7%.

Other examples illustrating the inaccuracy introduced by local balance assumptions are found in (C2).

## 5.3 Norton's Theorem Applied to Central Server Models

This section reviews earlier work on Norton's Theorem in subsection 5.3.1, in 5.3.2 a multiclass example is presented, and computational algorithms are presented in 5.3.3.

### 5.3.1 Norton's Theorem: A Discussion

Norton's Theorem (C3) may be used to transform a central server model in local balance into one with a single "composite" I/O which represents

the combined effects of the I/O devices in the original model at steady state. See Figure 5.2. Values determined for equilibrium cycle times, throughputs, server utilizations, and CPU queue length and waiting time distributions of the two-queue model will be the same as those calculated for the original model. The transformation is independent of the CPU parameters, so if a variety of CPU parameters are to be studied, effort may be saved by applying Norton's Theorem and studying the reduced model as the CPU parameters are varied. The approximation technique presented here is also especially well suited for parametric analysis of the CPU.

In describing Norton's Theorem we shall assume that there are J classes of customers. The composite I/O processes all classes of customers in parallel in the two queue, CPU-composite I/O model. The composite I/O service rate for the first customer of any given class i at any given time depends upon i and upon the number of customers $N_j$ of class j, j = 1,...,J, in the composite I/O queue at that time. These composite I/O service rates are determined by analyzing a modified version of the original network in which the CPU has been "shorted," i.e., the mean CPU service time for all customers is set to zero. See Figure 5.3. The composite I/O service rate for the first customer of any given class i, when there are $N_j$ customers of class j, j = 1,...,J, in the composite I/O queue, is set equal to the throughput of the customers in class i through the shorted CPU when there is a population of $N_j$ customers of class j, j = 1,...,J, in the shorted CPU model. The solutions of the two-queue, CPU-composite I/O model, with the same CPU parameters as in the original model and these queue-dependent composite I/O service rates, will be identical to those of the original model for the equilibrium statistics mentioned above.

### 5.3.2 Example

Consider the following two-class example of a locally-balanced central-server model with a processor-shared CPU and two I/Os labeled 1 and 2, two non-identical customers, one of class A and the other of class B. The class A customer uses I/O's 1 and 2 with equal probability, while the class B customer uses I/O 1 exclusively. The mean service time for each I/O is independent of customer class. The mean service times for I/Os 1 and 2 are 1 and 2, respectively. All I/O service times have negative-exponential distributions.

Both class A and B customers are assumed to be serviced in parallel in the composite I/O queue. The service rates for class A and class B customers depend upon the numbers of class A and B customers in the composite I/O queue. We next discuss the computation of these rates by analyzing the modified version of the original network in which the CPU has been shorted (Figure 5.3). When only the class A customer is present in the CPU-shorted network, the throughput of the class A customer through the shorted CPU is 2/3; when only the class B customer is present the throughput is 1; and when both are present, the throughputs for classes A and B are 1/2 and 3/4, respectively. The composite I/O service rates when there is one customer of class A and none of class B in the composite I/O queue is set to 2/3 for class A (and 0 for class B); when there is one customer of class B and none of class A the rate is set to 1 for class B (and 0 for class A); and when there is one customer of each class the rates are set to 1/2 for class A and and 3/4 for class B. The solution of the CPU-composite-I/O model with the same CPU parameters as in the original model and these queue-dependent composite

I/O service rates, will be identical to the solutions of the original model for the equilibrium statistics mentioned above.

### 5.3.3 Determination of Composite I/O Throughput

In this section we review computational techniques developed by Buzen (B6) and extended by Chandy, Herzog and Woo (C3). We will assume an arbitrary closed network in local balance with R single server queues numbered from 1 to R. We assume that J, the number of customers, is 2, and later consider arbitrary $J \geq 1$. We assume that customers cannot change class, that the mean service rate at queue r for class j is $\lambda_{rj}$, that a class j customer leaving queue r joins queue r' with probability $p_{(rj),(r'j)}$, and that the number of class j customers at queue r is $n_{rj}$. We must have $\sum_{r=1}^{R} n_{rj} = N_j$. We define $e_{rj}$, the expected number of times a customer of class j visits queue r, by J sets of R linear equations of the form

$$e_{rj} = \sum_{r'=1}^{R} e_{r'j} p_{(r'j),(rj)} \tag{5.1}$$

For a given j, $\{e_{rj}\}$ is uniquely determined up to a multiplicative constant.

We know from (B2) that the probability of having $n_{rj}$ customers of class j at queue r, $j = 1,2$, $r = 1,\ldots,R$, $P(n_{11},n_{12},n_{21},n_{22},\ldots,n_{R1},n_{R2})$ is

$$\frac{1}{G} \prod_{r=1}^{R} \binom{n_{r1} + n_{r2}}{n_{r1}} \prod_{j=1}^{2} \left( \frac{e_{rj}}{\lambda_{rj}} \right)^{n_{rj}} \tag{5.2}$$

Here G is a normalizing constant chosen so that the probabilities sum to 1. Of course G is dependent on $N_j$, $j = 1,2$, so we will refer to $G(N_1,N_2)$. So we must have

$$G(N_1,N_2) = \sum_{\substack{\Sigma n_{r1}=N_1 \\ n_{r1} \geq 0}} \sum_{\substack{\Sigma n_{r2}=N_2 \\ n_{r2} \geq 0}} \prod_{r=1}^{R} \binom{n_{r1}+n_{r2}}{n_{r1}} \prod_{j=1}^{2} \left(\frac{e_{rj}}{\lambda_{rj}}\right)^{n_{rj}} \quad (5.3)$$

The reader should be alarmed by the computation required to determine (5.2) and (5.3); though we could determine throughput from these expressions, more efficient algorithms are needed.

It can be shown that the throughput of class 1 customers through queue r when there are $N_j$, j = 1,2, customers in the network, $T_{r1}(N_1,N_2)$, is

$$T_{r1}(N_1,N_2) = \frac{e_{r1}G(N_1-1,N_2)}{G(N_1,N_2)} \quad (5.4)$$

Similarly,

$$T_{r2}(N_1,N_2) = \frac{e_{r2}G(N_1,N_2-1)}{G(N_1,N_2)} \quad (5.5)$$

So we need only find efficient algorithms for determining $G(N_1-1,N_2)$, $G(N_1,N_2-1)$ and $G(N_1,N_2)$. We now consider an algorithm for determining $G(n_1,n_2)$ for $n_1 = 0,\ldots,N_1$ and $n_2 = 0,\ldots,N_2$.

We let G be an array with first subscript ranging from 0 to $N_1$ and second subscript ranging from 0 to $N_2$. We will define arrays $X_r$, r = 1,...,R and a "convolution" operator "*" such that

$$G = X_1 * X_2 * \ldots * X_R. \quad (5.6)$$

The operator "*" is associative and commutative, but for convenience we determine G as $G_R$, where $G_r = G_{r-1} * X_r$, r = 2,...,R, and $G_1 = X_1$. For r = 1,...,R, $n_1 = 0,\ldots,N_1$, $n_2 = 0,\ldots,N_2$, we define

$$X_r(n_1,n_2) = \binom{n_1+n_2}{n_1} \left(\frac{p_{r1}}{\lambda_{r1}}\right)^{n_1} \left(\frac{p_{r2}}{\lambda_{r1}}\right)^{n_2} \quad (5.7)$$

Notice that we do not have to evaluate (5.7) for each element of $X_r$. For example, for $n_1, n_2 > 0$,

$$X_r(n_1,n_2) = \frac{n_1 + n_2}{n_1}\left(\frac{\rho_{r1}}{\lambda_{r1}}\right) X_r(n_1-1,n_2)$$

$$= \frac{n_1 + n_2}{n_2}\left(\frac{\rho_{r2}}{\lambda_{r2}}\right) X_r(n_1,n_2-1) \qquad (5.8)$$

For $r = 1,\ldots,R$, $n_1 = 0,\ldots,N_1$, $n_2 = 0,\ldots,N_2$

$$G_r(n_1,n_2) = \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} G_{r-1}(i_1,i_2)X_r(n_1-i_1,n_2-i_2) \qquad (5.9)$$

We now apply these algorithms to the example of Section 5.3.2. We let class A correspond to class 1 and class B correspond to class 2. So we have $p_{11} = p_{21} = .5$, $p_{12} = 1$, $p_{22} = 0$, $\mu_{11} = \mu_{12} = 1$ and $\mu_{21} = \mu_{22} = .5$. We can let $e_{rj} = p_{rj}$. Then

$$G_1 = X_1 = \begin{pmatrix} 1 & 1 \\ .5 & 1 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

$$G = G_2 = \begin{pmatrix} 1\cdot1 & 1\cdot0 + 1\cdot1 \\ 1\cdot1 + .5\cdot1 & 1\cdot0 + 1\cdot1 + .5\cdot0 + 1\cdot1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 1.5 & 2 \end{pmatrix}$$

If we let $T_j(n_1,n_2) = \sum_{n=1}^{R} T_{rj}(n_1,n_2)$, then

$$T_1(1,0) = \frac{1}{1.5} = \frac{2}{3}$$

$$T_1(1,1) = \frac{1}{2}$$

$$T_2(0,1) = \frac{1}{1} = 1$$

$$T_2(1,1) = \frac{1.5}{2} = \frac{3}{4}$$

When $j \geq 1$, then (5.7) and (5.9) become

$$X_r(n_1,n_2,\ldots,n_j) = \binom{n_1+n_2+\ldots+n_j}{n_1 n_2 \ldots n_j} \left(\frac{p_{r1}}{r1}\right)^{n_1} \left(\frac{p_{r2}}{r2}\right)^{n_2} \ldots \left(\frac{p_{rj}}{rj}\right)^{n_j} \quad (5.10)$$

and

$$G_r(n_1,n_2,\ldots,n_j) = \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} \ldots \sum_{i_j=0}^{n_j} G_{r-1}(i_1,i_2,\ldots,i_j) X_r(n_1-i_1, n_2-i_2,\ldots,n_j-i_j) \quad (5.11)$$

These algorithms also apply to networks with queue length dependent service rates.

Assume queue r is FCFS with queue length dependent service rates. Let $\lambda(n)$ be the rate with n customers in the queue. (Queue length dependent service rates are useful for representing multiple servers at a single queue. For example, if we have k servers each with rate $\mu$, we let $\lambda(n) = \min(n,k)\mu$.) We can let

$$X_r(n_1,n_2) = \binom{n_1+n_2}{n_1} \frac{p_{r1}^{n_1} \; p_{r2}^{n_2}}{\lambda(1)\lambda(2)\ldots\lambda(n_1+n_2)} \quad (5.12)$$

If we define $X_r$ as in (5.12), then G and the throughput will be as before.

## 5.4 FCFS Central Server Models with Non-Exponential Service Times

We first discuss the overall technique generally (5.4.1), then study composite I/O representations (5.4.2), present the detailed algorithm (5.4.3), and finally work out an example (5.4.4).

### 5.4.1 Overview

We now restrict our attention to central server models with all customers identical, FCFS disciplines at all servers, and arbitrary service distributions having rational Laplace transforms. For sake of discussion, we will assume that the system being modeled has a single CPU. These techniques have also been applied to models with multiple identical CPU's. Even though this class of models is not in local balance except when all service distributions are exponential, we shall apply Norton's Theorem and show that the composite I/O model yields solutions close to those of the original model. (In making the composite I/O transformation we assume that the I/O devices have exponential distributions with the same means as the actual distributions. See example below.) Chandy, Herzog and Woo (C2) use an approximate application of Norton's Theorem in their iterative method. In order to compensate for the inaccuracy introduced, we adjust the distributions for the composite I/O to reflect the non-exponential character of the actual distributions.

After applying Norton's Theorem and adjusting the distributions, we have a central server model with a single composite I/O, with both service distributions non-exponential. This model is solved by an efficient recursive technique which is an application of the technique developed by Herzog, Woo and Chandy (H1). Their technique assumes distributions of the generalized Erlang form developed by Cox (C5). This generalized form includes arbitrary distributions with rational Laplace transform. Our technique assumes that both the CPU and the I/O distributions are of this general form. Details of our two queue analysis are given in Chapter IV.

Our adjustment for the non-exponential nature of the I/O distributions is simple and effective. More sophisticated adjustments could potentially

increase the accuracy of the final results. We characterize each I/O distribution by its mean and coefficient of variation (standard deviation divided by the mean). For the means of the composite I/O distributions we use the queue length dependent values as shown earlier. We assume that the composite I/O coefficient of variation is the weighted sum of the coefficients of variation of the individual distributions, with the weights being the I/O branching probabilities. The composite I/O coefficient of variation is a constant, independent of queue length. Of course, the mean and coefficient of variation do not completely specify the distribution. If the composite I/O coefficient of variation is greater than one, we assume that the composite I/O service time is a standard two stage hyper-exponential as in Figure 5.4. If the coefficient of variation is one, we assume the service time is exponential. If the coefficient of variation is less than one, we assume the service time is of the generalized Erlang form with the minimum number of stages necessary to obtain the given coefficient of variation, all stages having the same mean, and all branching probabilities zero, with the possible exception of the branch after the first stage, as in Figure 5.5.

### 5.4.2 The Composite I/O Distribution

We desire that the composite I/O distribution represent the aggregate of all the individual I/O distributions. Intuitively, we expect the distribution of a given I/O to influence the composite I/O distribution more than distributions of other I/Os, if the given I/O processes more customers than other I/Os. We decided to restrict attention to the first two moments to keep computation simple. The means of composite I/O service times are obtained by aggregating individual I/O mean service times via Norton's Theorem. The

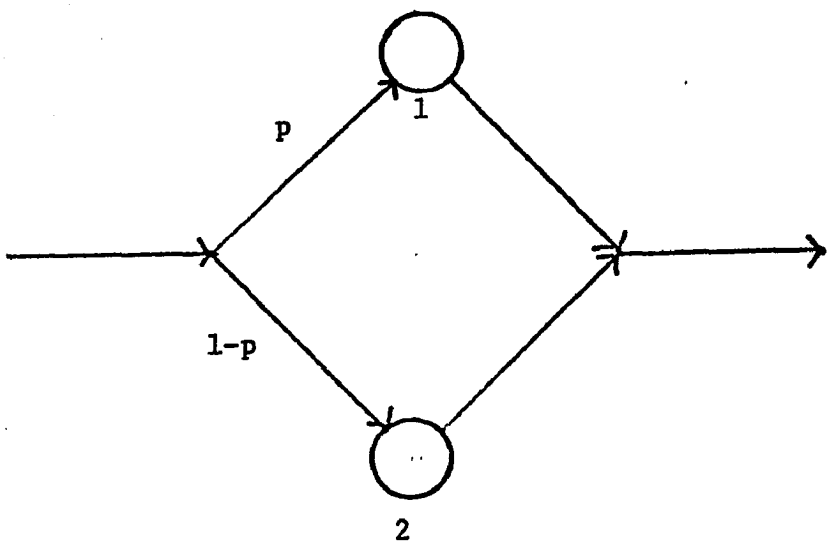



Figure 5.4

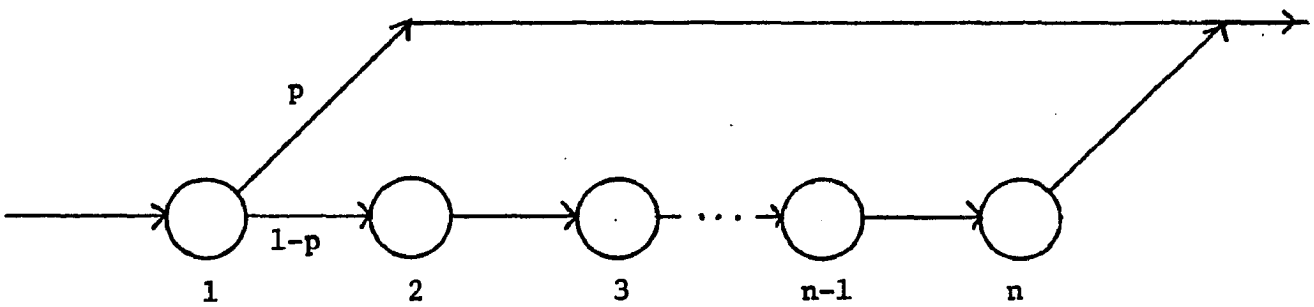



Figure 5.5

composite coefficient of variation is obtained by aggregating individual coefficients of variation, weighting each I/O by its branching probability since I/O branching probabilities are directly proportional to I/O throughputs. Note that though the mean composite service time is queue-length dependent, the coefficient of variation is not dependent on queue length. Note also that if all the I/Os have the same coefficient of variation, then the composite I/O will have that coefficient of variation too.

The first two moments do not completely specify a distribution. We decided to model composite service times using either two-stage hyper-exponential (Figure 4.4) or generalized Erlang (Figure 4.5) random variables since these are common ways of representing service times in computing systems. Note that the particular forms of the hyper-exponential and generalized Erlang random variables are such that the first two moments uniquely specify the distributions. The selection of these particular composite I/O distributions were made with modeling convenience and reasonability in mind; clearly other choices could also have been made. However, note that if the original model satisfies local balance, then our technique gives exact results, since the composite I/O distribution obtained via our technique is the same as that obtained via Norton's Theorem.

The Hyper-Exponential

Let $k_c$ be the coefficient of variation of the composite I/O. We shall use a standard hyper-exponential random variable to model composite I/O service times if $k_c > 1$. The relationship between $k_c$ and parameter p (Figure 5.4) of the hyper-exponential is shown below:

$$p = \frac{k_c^2 + 1 - \sqrt{k_c^4 - 1}}{2(k_c^2 + 1)} \tag{5.13}$$

Note that $k_c$ uniquely specifies p. The means for each stage of this hyper-exponential are uniquely specified by p and the mean composite service time.

$$\text{Mean of stage 1} = \frac{\text{mean composite service time}}{2p} \tag{5.14}$$

$$\text{Mean of stage 2} = \frac{\text{mean composite service time}}{2(1-p)} \tag{5.15}$$

Generalized Erlang

Consider the generalized Erlang (Figure 5.5) with n stages, n = 2,3,4,.... After a customer completes the first stage, he may finish service with probability p, or he may continue through the remaining n - 1 stages with probability 1-p. All stages have the same mean time, and all stage holding times are independent exponential random variables. By varying p from 0 to 1 the coefficient of variation ranges from $1/\sqrt{n}$ to 1. We wish to keep the number of stages small to minimize computation. Hence, we shall use n stages if and only if, $1/\sqrt{n-1} > k_c \geq 1/\sqrt{n};$. The value of n is directly determined from $k_c$. n and $k_c$ together uniquely specify p. See equation (5.16) below. The means for each stage are uniquely specified by n, $k_c$, p and the means of the composite I/O service times.

$$p = \frac{2nk_c^2 + n-2 - \sqrt{n^2+4 - 4nk^2}}{2(k_c^2 + 1)\ (n-1)} \tag{5.16}$$

$$\text{Mean of each stage} = \frac{\text{mean composite service time}}{n - p(n-1)} \tag{5.17}$$

In conclusion, the generalized Erlang and hyper-exponential random variables shown in Figures 5.4 and 5.5, are completely specified by the first

two moments, and have a wide range of coefficients of variation. The parameters p are independent of composite I/O mean service times and the mean times for all stages in both distributions are directly proportional to the composite I/O mean service time; this simple relationship is an advantage in modeling queue-dependent service rates.

### 5.4.3 The Algorithm

We now present the algorithm after explaining some notation. Let there be R I/O queues indexed 1,...,r,...,R. We shall use the subscript r to denote the rth I/O in the original model and the subscript c to denote the composite I/O in the CPU-composite-I/O model. Let $p_r$ be the probability that a customer branches to the rth I/O device after finishing CPU service. Let k denote the coefficient of variation: $k_c$ for the composite I/O and $k_r$ for the rth I/O device. We shall use the subscript 0 (zero) for the CPU. Let $U_r$ be the utilization and $t_r$ the throughput for the rth queue, r=0,1,...,R. Let $\lambda_r$ be the service rate for the rth I/O device. Let $\bar{q}$ and $\bar{w}$ be the mean CPU queue length and wait times and let $\sigma_q$ and $\sigma_w$ be the corresponding standard deviations. Let C be the cycle time; C is very important since response time in the computer system will be dependent on C.

ALGORITHM 5.1

Step 1. Composite I/O Service Rates

Consider the given (non-locally-balanced) model. Construct the shorted-CPU model in which all I/O service times are assumed to be independent exponential random variables and the CPU service time is set to zero. The shorted-CPU model satisfies local balance and can be analyzed easily.

Determine queue-dependent composite I/O service rates by analyzing the shorted-CPU model.

Step 2. Composite I/O Coefficient of Variation

$$\text{Compute} \qquad k_c = \sum_{r=1}^{R} k_r \cdot p_r$$

Step 3. Determine exponential stage representations for composite I/O service times from $k_c$ and composite I/O mean service times.

If $k_c > 1$ use standard hyper-exponential random variable. (Fig. 5.4)

If $k_c = 1$ use exponential random variable

If $k_c < 1$ use generalized Erlang random variable. (Fig. 5.5)

Step 4. Solve the two queue, CPU-composite I/O model.

The CPU parameters in this model are set to the same values as in the original model. The composite I/O parameters are completely and uniquely specified by step 3. The two-queue model is completely specified. Analyze this model to determine C, $U_0$, $t_0$, $\bar{q}$, $\sigma_q$, $\bar{w}$, and $\sigma_w$.

Step 5. I/O Utilizations

$$\text{Compute} \qquad t_r = t_0 \times p_r \qquad \text{for } r = 1,\ldots,R$$

$$U_r = t_r/\lambda_r \qquad \text{for } r = 1,\ldots,R$$

stop.

5.4.4 Example

Consider a two I/O model with 2 customers where I/O 1 has an exponential service time with mean 4. I/O 2 has a generalized Erlangian service time with a coefficient of variation of .414 and mean 2 and the CPU

has a standard hyper-exponential service time with a coefficient of variation of 2 and a mean of 2. Customers use each I/O with equal probability. We shall now follow through the five steps of the algorithm.

Step 1. The composite I/O service rates (from Section 5.3.3) when there are j customers, j = 1,2, in the composite I/O queue are 1/3 and 3/7, respectively.

Step 2. $k_c = (0.5 \times 1.0) + (0.5 \times 0.414) = 0.707$

Step 3. Since $k_c < 1$ the generalized Erlang representation is used. In this case n will be 2 and p will be zero. (The rate for each stage is clearly twice the composite I/O service rate.)

Step 4. We now have a two-queue model where the CPU service time is a two-stage hyper-exponential and the composite I/O service time is a two-stage Erlang. The balance equations for the resulting Markov states are solved to obtain $C = 6.99$, $U_0 = .571$, $t_0 = .286$, $\bar{q}_0 = .837$, $\bar{w}_0 = 2.93$

Step 5. $t_1 = t_0 \times 0.5 = .143$, $t_2 = t_0 \times 0.5 = .143$

$U_1 = t_1/\lambda_1 = .571$, $U_2 = t_2/\lambda_2 = .286$

stop.

## 5.5 FCFS Central Server Models with Class Dependent Service Rates

This section is divided into three subsections. In 5.5.1 we discuss the technique generally, in 5.5.2, the algorithm is presented and an example is worked out in 5.5.3.

### 5.5.1 Discussion

In this section, we restrict ourselves to models with several classes

of customers, FCFS, all service distributions exponential, all I/O service rates independent of customer class, and the CPU service rates dependent on customer class. The assumption of class independent I/O service rates can be justified by observing that the largest portion of most I/O services is spent on primarily program independent operations such as acquiring channels, positioning disk arms, and waiting for device rotation. The techniques presented here have been extended to non-exponential CPU distributions and can also easily be extended to non-exponential I/O distributions. They are extended to priority disciplines in the next section, using the techniques of the previous section. Our techniques may also be extended to other, more general models.

Multiple classes severely complicate analysis. Even the reduced model obtained by applying the Norton's Theorem approximation to the I/O subnetwork is difficult to analyze. As the number of classes and/or the numbers of customers per class attain even moderate values, e.g., 4, the analysis becomes too complex to be of practical value.

To reduce the complexity of analysis, we transform the more general original model to an approximately equivalent one with only two classes of customers: a designated class with only one customer and a composite class representing all of the other customers in the network. This further reduced model can be analyzed relatively easily, by applying the Norton's Theorem approximation. We designate each class in the original model and in turn analyze the corresponding reduced model, thus we obtain approximate values for the interesting statistics by each customer class in the original model.

In transforming the original model to the one with only two classes, the customer of the designated class is given the same I/O branching

probabilities and CPU service distribution as in the original model. For each I/O device, the composite class branching probability is determined as a weighted sum of the branching probabilities of the classes coalescing from the original model. The weights used are the relative throughputs of the corresponding customers in a model identical to the original model, except that the CPU is processor-shared; this PS model satisfies local balance and is easily analyzed. The CPU service distribution for the composite class is chosen to be the standard two stage hyper-exponential distribution with mean and second moment determined from weighted sums of the means and second moments of the CPU service distributions of the classes being coalesced from the original model.

After this transformation is applied, the Norton's Theorem approximation is applied. The resulting model, with the composite class and composite I/O queue is analyzed by techniques similar to those used in Section 5.4.

### 5.5.2 Algorithms

In this subsection we describe two algorithms, the main program, algorithm 5.2, is presented in 5.2.1., and a subprogram, algorithm 5.3, which approximates an N-class problem by a two-class problem, is in 5.5.2.2.

#### 5.5.2.1 ALGORITHM 5.2

Assume that there are N classes of customers. For purposes of exposition, we assume (without loss of generality) that there is only one customer in each class.

Step 1. For each class i in turn, i = 1,...,N, do steps 2-i through 5-i and thus compute the throughputs and utilizations for all queues for class i, and

also the means and variances of CPU queue lengths and wait times for class i. The algorithm stops after all N classes have been considered.

Step 2-i. Use algorithm 5.3 to approximate the given N-class problem by a two-class problem where the two classes are the designated class and a "composite class" which represents all customers except those in the designated class. We shall refer to the original central-server model as model A and this two-class approximation as model B. Note that B and A have exactly the same central-server network structure; only the number of classes is changed. The parameters for the designated class are the same in A and B. CPU service time for the composite class is assumed to be hyperexponential in B. I/O service times are identical in A and B.

Step 3-i. Compute composite I/O service rates for the designated and composite classes of model B in the usual manner (i.e., by computing throughputs through the shorted CPU of model B and assuming all I/O service times are exponential).

Step 4-i. Consider the resulting two-queue, two-class network consisting of the CPU and I/O queues and the designated and composite classes; we shall refer to this network as model C. Solve Markov balance equations to determine steady-state probabilities of model C. Determine CPU throughput $t_{0i}$, utilization $U_{0i}$, mean and variance of CPU queue length and wait time for designated class i from the equilibrium state probabilities of model C. (Statistics for the composite class are not computed).

Step 5-i. Determine I/O throughputs $t_{ri}$, and utilizations $U_{ri}$, for each I/O r, r = 1,...,R, for the designated class i. Let $p_{ri}$ be the probability

that a customer of class i branches to I/O r after CPU service. Then

$$t_{ri} = t_{0i} \times p_{ri} \qquad \text{for } r = 1,\ldots,R$$

and

$$U_{ri} = t_{ti}/\lambda_i \qquad \text{for } r = 1,\ldots,R$$

Statistics for the composite class are not computed.

Figure 5.6 shows the relationships between models A, B and C.

5.5.2.2 ALGORITHM 5.3

For determining CPU service distributions and I/O branching p probabilities for the coalesced class.

Step 1. Consider a network identical to the given network (model A) except that the CPU is processor-shared; we shall refer to this network as model D. Model D satisfies local balance and is easily analyzable. (See Section 5.3.3.) For the purposes of Algorithm 5.3 only, we shall approximate the CPU throughputs of model A by those of model D. Compute $t_j'$, the CPU throughput of class j in model D, for j = 1,...,N.

Step 2. Compute the conditional probability $V_j$ that a random customer who finishes I/O service in model D is in class j given that he is not in designated class i.

$$V_j = t_j' \Big/ \sum_{h \neq i} t_h' \qquad \text{for } j \neq i$$

$$= 0 \qquad \text{for } j = i$$

Step 3. Compute the first two moments of the CPU service time for the composite class. Let $E[S^n]$ and $E[S_j^n]$ be the nth moment of the CPU service time for the coalesced class and class j respectively, j = 1,...,N. Then:

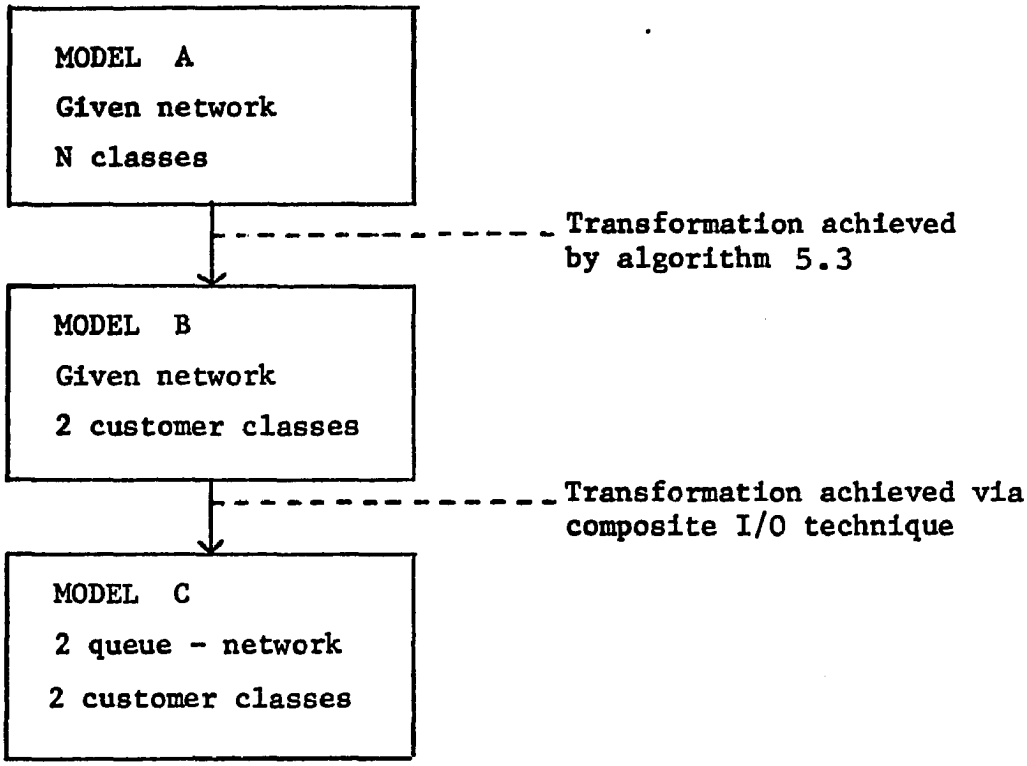



FIGURE 5.6

$$E[S] = \sum_j E[S_j] \cdot V_j$$

$$E[S^2] = \sum_j E[S_j^2] \cdot V_j$$

Represent CPU service time for the composite class by a standard hyper-exponential random variable (Figure 5.4) with the above first two moments.

Step 4. Approximate I/O branching probabilities for the composite class by

$$p_{rc} = \sum_j p_{rj} \cdot V_j$$

Stop.

### 5.5.3 Example

Consider a model with two I/Os and three classes of customers. The mean service times for I/O 1 and I/O 2 are both 2 time units. The branching probabilities for the first I/O are 1., 0, .5, for classes 1, 2 and 3, respectively, and 0, 1., .5, for the second I/O. CPU mean times for classes 1,2,3 are 1,2,3, respectively. All service times are assumed to be independent, exponential random variables.

ALGORITHM 5.2 - Step 1. We shall carry out steps 2-i through 5-i, for i=1.

We first call Algorithm 5.3 to obtain the 2-class approximation.

ALGORITHM 5.3 - Step 1. Analyzing model D we get

$$t_2' = .159 \qquad t_3' = .111$$

ALGORITHM 5.3 - Step 2. $V_1 = 0$, $V_2 = .589$, $V_3 = .411$

ALGORITHM 5.3 - Step 3. $E[S] = (2 \times .589) + (3 \times .411) = 2.41$

$E[S^2] = (8 \times .589) + (18 \times .411) = 12.12$

The hyper-exponential representation for the CPU service time has parameter $p = 0.398$.

ALGORITHM 5.3 - Step 4.

$$p_{1c} = (0 \times .589) + (.5 \times .411) = .206$$

$$p_{2c} = (1 \times .589) + (.5 \times .411) = .795$$

We now have a two-class problem the CPU service time for the composite class is hyper-exponential with mean 2.41 and I/O branching probabilities for device 1 is .206 and for device 2 is .795.

ALGORITHM 5.2 - Step 3.1. The composite I/O service rates for class 1 and the composite class for different queue conditions are shown below.

| Number of class 1 customers | Number of composite class customers | Total rate for class 1 | Total rate for composite class |
|---|---|---|---|
| 1 | 0 | .5 | 0 |
| 0 | 1 | 0 | .5 |
| 0 | 2 | 0 | .598 |
| 1 | 1 | .415 | .415 |
| 1 | 2 | .386 | .556 |

ALGORITHM 5.2 - Step 4.1. Model C is analyzed to obtain $t_{01} = .173$, $U_{01} = .173$, $\bar{q}_1 = .59$, $w_1 = 3.43$, $\sigma_{q_1} = .49$, $\sigma_{w_1} = 3.09$.

## 5.6 Approximations for Models with Priority CPU Disciplines

Now we consider central server models with the same characteristics as in the previous section, except that the CPU discipline will be a priority

discipline with priority based on customer class. We will restrict consideration to preemptive and non-preemptive priority based on customer class, but these techniques are directly applicable to other priority disciplines.

Again, we do not try to apply Norton's Theorem approximation directly, but rather combine the classes of customers in the original model to simplify the analysis. The reduced model we consider has three classes of customers: a designated class, which we do not restrict to a single customer as in the FCFS model, and two composite classes, one of a higher priority than the designated class, and one of lower priority. The combination of classes into these three classes is similar to the technique used in the previous section. The coalescing is done separately for each of the two composite classes. The CPU distribution used for each of the composite classes is an exponential distribution with mean taken as the weighted sum of the means of the classes coalescing into that composite class. The weights are the relative throughputs of classes within the composite class. In other respects, the analysis is essentially the same as that already described.

## 5.7 Validation, Implementation and Performance

We have constructed a simulator which employs the confidence interval techniques of Crane and Iglehart (C6,C7). This simulator can be used with general queueing networks with a variety of disciplines, hetergeneous, classes of customers, and generalized Erlang service distributions. The simulator determines confidence intervals during the simulation, and continues the simulation until satisfactory intervals are obtained. Details of the simulator are found in Chapter VI. This simulator has been used to determine

results for the various models described below. Crane and Iglehart show how to obtain confidence intervals for results of simulations of Markov models. In general, the confidence intervals we obtained are as follows: For utilization, the 90% intervals are at most .05 wide. For those cases where queue lengths and waiting times are obtained, the, the 90% intervals for the means are at most $\pm$ 6% of the point estimates, and the 80% intervals for the standard deviations are at most $\pm$ 16% of the point estimates. In many of the cases the intervals are considerably tighter. However, we were unable to obtain confidence intervals for the FCFS models with 6 classes of customers. For these models the state space is very large, and we were unable to select a state that the system would return to frequently; this is necessary to apply the Crane and Iglehart techniques. We used predetermined simulation run lengths for the 6 class FCFS models, with the run lengths based on experience with 4 class FCFS models. For the models with multiple CPU's or constant service times we used simulators constructed in QSIM (F1,M2).

We have implemented our approximation techniques as a set of Fortran programs for a CDC 6600. Over 125 models have been validated to assure a thorough sampling of problems.

56 of the models validated are of the class described in Section 5.4, i.e., single class, non-exponential. In general the models were fairly well balanced, but some of the models were strongly CPU bound or I/O bound. See Table 5.1. Error tolerances were determined in the manner used in (5) for CPU utilizations, CPU queue lengths and CPU waiting times. Results are said to be within a tolerance z if 1) the difference in utilization is not more than z, 2) the differences in the means and standard deviations of queue

TABLE 1 - NON-EXPONENTIAL MODEL DESCRIPTIONS

| MODEL | NO. CUST. | NO. CPUS | CPU | | I/O 1(5) | | | I/O 2(6) | | | I/O 3(7) | | | I/O 4(8) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MEAN | C.V. | PROB | MEAN | C.V. | PROB | MEAN | C.V. | PROB | MEAN | C.V. | PROB | MEAN | C.V. |
| 1 | 2 | 2 | 2.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 2 | 4 | 2 | 2.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 3 | 8 | 2 | 2.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 4 | 12 | 2 | 2.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 5 | 2 | 1 | 1.000 | 0.000 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 6 | 4 | 1 | 1.000 | 0.000 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 7 | 8 | 1 | 1.000 | 0.000 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 8 | 12 | 1 | 1.000 | 0.000 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 9 | 2 | 1 | 1.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 10 | 4 | 1 | 1.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 11 | 8 | 1 | 1.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 12 | 12 | 1 | 1.000 | 2.134 | .500 | 2.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | .250 | 1.000 | | | |
| 13 | 2 | 1 | 1.000 | 1.000 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 14 | 4 | 1 | 1.000 | 1.000 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 15 | 8 | 1 | 1.000 | 1.000 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 16 | 12 | 1 | 1.000 | 1.000 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 17 | 2 | 1 | 1.000 | 2.134 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 18 | 4 | 1 | 1.000 | 2.134 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 19 | 8 | 1 | 1.000 | 2.134 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 20 | 12 | 1 | 1.000 | 2.134 | .500 | 2.000 | .707 | .250 | 1.000 | .707 | .250 | .250 | .707 | | | |
| 21 | 2 | 1 | 1.000 | 1.000 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| | | | | | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| 22 | 4 | 1 | 1.000 | 1.000 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| | | | | | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| 23 | 8 | 1 | 1.000 | 1.000 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| | | | | | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| 24 | 12 | 1 | 1.000 | 1.000 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| | | | | | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 | .125 | 4.000 | .707 |
| 25 | 2 | 1 | 1.000 | 5.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 |
| 26 | 4 | 1 | 1.000 | 5.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 |
| 27 | 8 | 1 | 1.000 | 5.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 |
| 28 | 12 | 1 | 1.000 | 5.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | 1.000 |
| 29 | 2 | 1 | 1.000 | 3.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 |
| 30 | 4 | 1 | 1.000 | 3.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 |
| 31 | 8 | 1 | 1.000 | 3.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 |
| 32 | 12 | 1 | 1.000 | 3.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | 1.000 |
| 33 | 2 | 1 | 1.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | .707 | .250 | 8.000 | .707 | .250 | 8.000 | .577 |
| 34 | 4 | 1 | 1.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | .707 | .250 | 8.000 | .707 | .250 | 8.000 | .577 |
| 35 | 8 | 1 | 1.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | .707 | .250 | 8.000 | .707 | .250 | 8.000 | .577 |
| 36 | 12 | 1 | 1.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | .707 | .250 | 8.000 | .707 | .250 | 8.000 | .577 |
| 37 | 2 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 |
| 38 | 4 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 |
| 39 | 8 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 |
| 40 | 12 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 3.000 |
| 41 | 2 | 1 | 1.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 |
| 42 | 4 | 1 | 1.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 | .250 | 4.000 | .577 |
| 43 | 2 | 1 | 1.000 | .577 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 44 | 4 | 1 | 1.000 | .577 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 45 | 2 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 46 | 4 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 47 | 8 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 48 | 12 | 1 | 1.000 | 3.000 | .250 | 4.000 | 3.000 | .250 | 4.000 | 1.000 | .250 | 4.000 | .707 | .250 | 4.000 | .577 |
| 49 | 2 | 1 | 1.000 | 3.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 |
| 50 | 4 | 1 | 1.000 | 3.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 |
| 51 | 8 | 1 | 1.000 | 3.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 |
| 52 | 12 | 1 | 1.000 | 3.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 | .250 | 8.000 | 1.000 |
| 53 | 2 | 1 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | .707 | .250 | 1.000 | .707 | .250 | 1.000 | .577 |
| 54 | 4 | 1 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | .707 | .250 | 1.000 | .707 | .250 | 1.000 | .577 |
| 55 | 8 | 1 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | .707 | .250 | 1.000 | .707 | .250 | 1.000 | .577 |
| 56 | 12 | 1 | 1.000 | 1.000 | .250 | 1.000 | 1.000 | .250 | 1.000 | .707 | .250 | 1.000 | .707 | .250 | 1.000 | .577 |

length are not more than z times the number of customers in the network, and 3) the differences in the means and standard deviations of the wait times are not more than z times the cycle time. For the 56 models studied, the results are generally within a tolerance of .05, with a maximum tolerance of .17. In (C2) a tolerance of .05 is considered to be good, and a tolerance of .10 is considered adequate. By these standards the results are good for 47 of the models and adequate for 51 of the 56 models. The results for similar PS models are adequate for only 25 of the 56 models. For these models, the computer time required per model was negligible, approximately 75 milliseconds per model. Table 5.2 shows results for these models.

44 models of the class described in Section 5.5, i.e., FCFS with different classes of customers, including 4 with hyper-exponential CPU distributions, have been validated. These models include from 2 to 8 customers, with from 2 to 6 classes of customers, and 3 or 4 I/O devices. Utilizations and throughputs, both overall and by class, were validated for all of these models. For 8 of the models, queue lengths and wait times for each class were also validated. See Tables 5.3 and 5.4. We did not explicitly determine tolerances as in the single class models, but in general the results showed good accuracy for utilization and reasonable accuracy overall. For the 44 models, the programs required approximately 400 milliseconds computation per model.

36 priority models were validated, 24 preemptive and 12 non-preemptive. These models included from 4 to 6 customers, with from 3 to 6 classes, and 3 or 4 I/O devices. Again, utilizations and throughputs were validated for all models. CPU queue lengths and mean CPU wait times were validated for

TABLE 2 - NON-EXPONENTIAL MODEL RESULTS

| MODEL | TOLERANCE | | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | S.D.W.T. UTIL | | IO 1.5 | | IO 2.6 | | IO 3.7 | | IO 4.8 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APP | PS | APP | SIM | PS | APP | SIM | APP | SIM | APP | SIM | APP | SI | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 1 | .02 | .02 | 3.6 | 3.6 | 3.6 | 1.10 | 1.12 | 1.10 | 1.12 | .74 | .74 | 2.00 | 2.05 | | 4.35 | .55 | .54 | .14 | .14 | .03 | .03 | | |
| 2 | .01 | .05 | 5.5 | 5.5 | 5.3 | 1.46 | 1.46 | 2.06 | 2.06 | 1.36 | 1.35 | 2.83 | 2.80 | | 4.56 | .73 | .73 | .18 | .19 | .05 | .04 | | |
| 3 | .06 | .05 | 9.7 | 9.4 | 9.1 | 1.64 | 1.70 | 3.93 | 4.06 | 2.70 | 2.62 | 4.79 | 4.87 | | 6.03 | .82 | .82 | .21 | .21 | .05 | .05 | | |
| 4 | .04 | .07 | 14.0 | 13.6 | 13.1 | 1.72 | 1.76 | 5.79 | 6.00 | 4.05 | 3.94 | 6.74 | 6.81 | | 7.24 | .86 | .87 | .21 | .23 | .05 | .05 | | |
| 5 | .01 | .31 | 3.0 | 3.0 | 3.2 | .67 | .66 | .87 | .87 | .71 | .73 | 1.29 | 1.32 | .40 | .37 | .67 | .65 | .17 | .17 | .04 | .04 | | |
| 6 | .03 | .19 | 4.7 | 4.8 | 5.1 | .85 | .83 | 1.79 | 1.88 | 1.17 | 1.26 | 2.12 | 2.27 | .93 | .95 | .85 | .80 | .21 | .21 | .05 | .05 | | |
| 7 | .03 | .11 | 8.6 | 8.7 | 9.1 | .93 | .92 | 3.75 | 3.96 | 2.23 | 2.38 | 4.03 | 4.32 | 2.03 | 2.12 | .93 | .90 | .23 | .24 | .06 | .06 | | |
| 8 | .02 | .07 | 12.6 | 12.8 | 13.0 | .96 | .93 | 5.74 | 5.72 | 3.34 | 3.54 | 6.01 | 6.12 | 3.16 | 3.29 | .96 | .94 | .24 | .24 | .06 | .06 | | |
| 9 | .05 | .42 | 3.4 | 3.4 | 3.2 | .59 | .59 | .91 | .89 | .85 | .84 | 1.54 | 1.50 | 2.90 | 2.73 | .59 | .61 | .15 | .15 | .04 | .04 | | |
| 10 | .01 | .39 | 5.6 | 5.6 | 5.1 | .71 | .71 | 1.85 | 1.82 | 1.56 | 1.54 | 2.59 | 2.57 | 4.02 | 4.07 | .71 | .73 | .18 | .18 | .04 | .04 | | |
| 11 | .03 | .26 | 10.0 | 9.6 | 9.1 | .80 | .84 | 3.66 | 3.79 | 2.93 | 2.80 | 4.56 | 4.55 | 5.70 | 5.53 | .80 | .82 | .20 | .21 | .05 | .05 | | |
| 12 | .02 | .22 | 14.2 | 13.9 | 13.0 | .85 | .86 | 5.51 | 5.65 | 4.26 | 4.10 | 6.52 | 6.49 | 7.14 | 7.24 | .85 | .86 | .21 | .21 | .05 | .06 | | |
| 13 | .02 | .03 | 3.2 | 3.1 | 3.2 | .63 | .65 | .89 | .92 | .79 | .78 | 1.41 | 1.43 | 1.29 | 1.30 | .63 | .63 | .16 | .17 | .04 | .04 | | |
| 14 | .02 | .04 | 5.0 | 4.9 | 5.1 | .80 | .82 | 1.83 | 1.90 | 1.33 | 1.31 | 2.30 | 2.34 | 1.86 | 1.86 | .80 | .80 | .20 | .21 | .05 | .05 | | |
| 15 | .02 | .03 | 8.9 | 8.8 | 9.1 | .90 | .91 | 3.74 | 3.88 | 2.42 | 2.37 | 4.16 | 4.23 | 2.99 | 2.92 | .90 | .90 | .22 | .23 | .06 | .06 | | |
| 16 | .03 | .02 | 12.8 | 12.8 | 13.0 | .93 | .94 | 5.66 | 5.99 | 3.54 | 3.62 | 6.06 | 6.42 | 4.13 | 4.29 | .93 | .92 | .23 | .23 | .06 | .06 | | |
| 17 | .01 | .47 | 3.4 | 3.3 | 3.2 | .59 | .60 | .91 | .91 | .84 | .83 | 1.53 | 1.51 | 2.89 | 2.86 | .59 | .61 | .15 | .15 | .04 | .04 | | |
| 18 | .03 | .43 | 5.6 | 5.3 | 5.1 | .72 | .75 | 1.83 | 1.88 | 1.54 | 1.50 | 2.54 | 2.57 | 4.01 | 4.15 | .72 | .73 | .18 | .19 | .05 | .05 | | |
| 19 | .04 | .24 | 9.9 | 9.4 | 9.1 | .81 | .85 | 3.61 | 3.75 | 2.90 | 2.73 | 4.45 | 4.44 | 5.67 | 5.31 | .81 | .84 | .20 | .21 | .05 | .05 | | |
| 20 | .04 | .20 | 14.1 | 13.4 | 13.0 | .85 | .89 | 5.43 | 5.92 | 4.21 | 3.99 | 6.36 | 6.68 | 7.10 | 6.89 | .85 | .87 | .21 | .23 | .05 | .06 | | |
| 21 | .01 | .01 | 5.6 | 5.5 | 5.6 | .36 | .36 | .42 | .43 | .61 | .62 | 1.18 | 1.20 | 1.15 | 1.18 | .18 | .18 | .18 | .17 | .18 | .19 | .18 | .17 |
| | | | | | | | | | | | | | | | | .18 | .18 | .18 | .18 | .18 | .18 | .18 | .19 |
| 22 | .01 | .01 | 6.8 | 6.7 | 6.8 | .59 | .60 | .96 | .99 | 1.01 | 1.02 | 1.63 | 1.63 | 1.51 | 1.49 | .29 | .30 | .29 | .29 | .29 | .32 | .29 | .30 |
| | | | | | | | | | | | | | | | | .29 | .31 | .29 | .30 | .29 | .30 | .29 | .31 |
| 23 | .03 | .04 | 9.4 | 9.1 | 9.6 | .85 | .88 | 2.53 | 2.79 | 1.87 | 1.89 | 2.98 | 3.19 | 2.39 | 2.41 | .42 | .44 | .42 | .42 | .42 | .45 | .42 | .43 |
| | | | | | | | | | | | | | | | | .42 | .44 | .42 | .44 | .42 | .44 | .42 | .43 |
| 24 | .05 | .04 | 12.5 | 12.4 | 12.7 | .96 | .97 | 4.98 | 5.58 | 2.71 | 2.70 | 5.19 | 5.74 | 3.43 | 3.47 | .48 | .48 | .48 | .49 | .48 | .49 | .48 | .49 |
| | | | | | | | | | | | | | | | | .48 | .48 | .48 | .49 | .48 | .49 | .48 | .47 |
| 25 | .12 | 1.03 | 6.4 | 6.2 | 6.0 | .31 | .32 | .47 | .49 | .75 | .76 | 1.49 | 1.59 | 6.84 | 7.58 | .31 | .30 | .31 | .32 | .31 | .31 | .31 | .31 |
| 26 | .07 | .99 | 9.0 | 8.7 | 8.0 | .44 | .46 | 1.07 | 1.14 | 1.50 | 1.53 | 2.43 | 2.57 | 9.48 | 10.12 | .44 | .42 | .44 | .44 | .44 | .44 | .44 | .43 |
| 27 | .18 | .58 | 14.3 | 14.5 | 12.0 | .56 | .55 | 2.36 | 2.12 | 3.05 | 2.84 | 4.21 | 3.56 | 13.16 | 10.58 | .56 | .58 | .56 | .59 | .56 | .59 | .56 | .56 |
| 28 | .11 | .58 | 19.3 | 19.1 | 16.0 | .62 | .63 | 3.69 | 3.45 | 4.60 | 4.30 | 5.94 | 5.27 | 15.96 | 13.81 | .62 | .63 | .62 | .66 | .62 | .64 | .62 | .64 |
| 29 | .17 | 1.21 | 2.6 | 2.7 | 2.6 | .72 | .73 | 1.17 | 1.22 | .84 | .84 | 1.53 | 1.77 | 4.17 | 4.62 | .18 | .16 | .18 | .18 | .18 | .16 | .18 | .18 |
| 30 | .04 | .87 | 4.4 | 4.4 | 4.2 | .90 | .90 | 2.74 | 2.69 | 1.39 | 1.39 | 3.03 | 2.84 | 5.90 | 5.85 | .23 | .23 | .23 | .23 | .23 | .25 | .23 | .25 |
| 31 | .04 | .64 | 8.1 | 8.1 | 8.0 | .99 | .99 | 6.39 | 6.39 | 1.98 | 1.95 | 6.46 | 6.42 | 8.44 | 8.09 | .25 | .25 | .25 | .24 | .25 | .25 | .25 | .25 |
| 32 | .07 | .50 | 12.0 | 12.0 | 12.0 | 1.00 | 1.00 | 10.33 | 10.42 | 2.14 | 1.92 | 10.34 | 10.12 | 10.38 | 9.57 | .25 | .24 | .25 | .25 | .25 | .27 | .25 | .25 |
| 33 | .00 | .00 | 10.9 | 10.9 | 10.9 | .18 | .18 | .20 | .20 | .45 | .44 | 1.10 | 1.08 | 1.09 | 1.08 | .37 | .38 | .37 | .38 | .37 | .39 | .37 | .36 |
| 34 | .01 | .01 | 14.7 | 14.3 | 14.7 | .27 | .28 | .34 | .35 | .62 | .62 | 1.25 | 1.23 | 1.24 | 1.21 | .54 | .56 | .54 | .56 | .54 | .57 | .54 | .58 |
| 35 | .01 | .02 | 22.5 | 21.6 | 22.5 | .36 | .37 | .51 | .53 | .83 | .81 | 1.44 | 1.41 | 1.42 | 1.37 | .71 | .74 | .71 | .74 | .71 | .75 | .71 | .74 |
| 36 | .01 | .01 | 30.4 | 29.5 | 30.4 | .40 | .41 | .61 | .62 | .94 | .91 | 1.54 | 1.51 | 1.53 | 1.48 | .79 | .82 | .79 | .82 | .79 | .81 | .79 | .82 |
| 37 | .04 | .36 | 6.4 | 7.3 | 6.0 | .31 | .27 | .47 | .39 | .75 | .69 | 1.49 | 1.41 | 4.04 | 3.80 | .31 | .22 | .31 | .25 | .31 | .30 | .31 | .34 |
| 38 | .07 | .36 | 9.0 | 10.6 | 8.0 | .44 | .37 | 1.09 | .83 | 1.47 | 1.30 | 2.45 | 2.22 | 5.54 | 5.31 | .44 | .36 | .44 | .38 | .44 | .38 | .44 | .36 |
| 39 | .10 | .30 | 14.1 | 17.0 | 12.0 | .57 | .47 | 2.48 | 1.72 | 2.93 | 2.50 | 4.36 | 3.71 | 7.74 | 7.20 | .57 | .44 | .57 | .48 | .57 | .49 | .57 | .43 |
| 40 | .11 | .24 | 18.9 | 22.4 | 16.0 | .63 | .54 | 3.98 | 2.63 | 4.35 | 3.62 | 6.28 | 4.85 | 9.49 | 8.26 | .63 | .51 | .63 | .54 | .63 | .48 | .63 | .55 |
| 41 | .03 | .09 | 5.9 | 5.5 | 6.0 | .34 | .36 | .39 | .40 | .58 | .56 | 1.14 | 1.12 | .56 | .66 | .34 | .31 | .34 | .34 | .34 | .35 | .34 | .41 |
| 42 | .04 | .09 | 7.8 | 7.2 | 8.0 | .51 | .55 | .73 | .75 | .83 | .80 | 1.42 | 1.36 | .61 | .85 | .51 | .56 | .51 | .55 | .51 | .56 | .51 | .55 |
| 43 | .03 | .08 | 6.0 | 6.5 | 6.0 | .33 | .31 | .40 | .35 | .61 | .56 | 1.20 | 1.13 | .59 | .68 | .33 | .34 | .33 | .33 | .33 | .32 | .33 | .30 |
| 44 | .03 | .07 | 8.0 | 8.5 | 8.0 | .50 | .47 | .81 | .67 | .98 | .84 | 1.63 | 1.40 | .80 | .90 | .50 | .47 | .50 | .51 | .50 | .47 | .50 | .48 |
| 45 | .02 | .45 | 5.3 | 6.6 | 6.0 | .32 | .30 | .46 | .44 | .72 | .72 | 1.43 | 1.50 | 3.97 | 4.12 | .32 | .36 | .32 | .27 | .32 | .29 | .32 | .30 |
| 46 | .04 | .38 | 8.8 | 9.1 | 8.0 | .46 | .44 | 1.03 | .95 | 1.39 | 1.33 | 2.25 | 2.10 | 5.37 | 4.99 | .46 | .44 | .46 | .44 | .46 | .46 | .46 | .45 |
| 47 | .04 | .33 | 13.5 | 14.3 | 12.0 | .59 | .56 | 2.25 | 1.95 | 2.73 | 2.54 | 3.81 | 3.42 | 7.41 | 6.93 | .59 | .57 | .59 | .56 | .59 | .58 | .59 | .55 |
| 48 | .04 | .32 | 18.0 | 18.8 | 16.0 | .66 | .64 | 3.54 | 3.05 | 4.03 | 3.71 | 5.32 | 4.88 | 9.03 | 8.88 | .66 | .64 | .66 | .61 | .66 | .64 | .66 | .63 |
| 49 | .03 | .19 | 11.1 | 11.8 | 10.9 | .18 | .17 | .23 | .22 | .54 | .51 | 1.30 | 1.21 | 3.75 | 3.35 | .36 | .32 | .36 | .40 | .36 | .34 | .36 | .37 |
| 50 | .02 | .20 | 15.3 | 15.8 | 14.7 | .26 | .25 | .48 | .44 | .97 | .92 | 1.82 | 1.72 | 4.82 | 4.50 | .52 | .53 | .52 | .53 | .52 | .55 | .52 | .52 |
| 51 | .01 | .21 | 23.4 | 22.8 | 22.5 | .34 | .35 | .90 | .94 | 1.76 | 1.79 | 2.64 | 2.71 | 6.21 | 6.35 | .68 | .69 | .68 | .70 | .68 | .69 | .68 | .67 |
| 52 | .02 | .16 | 31.3 | 31.8 | 30.4 | .38 | .38 | 1.25 | 1.14 | 2.42 | 2.20 | 3.27 | 2.97 | 7.15 | 6.56 | .77 | .77 | .77 | .79 | .77 | .76 | .77 | .76 |
| 53 | .05 | .04 | 2.6 | 2.5 | 2.6 | .77 | .78 | 1.14 | 1.19 | .76 | .77 | 1.48 | 1.60 | 1.32 | 1.38 | .19 | .16 | .19 | .21 | .19 | .19 | .19 | .18 |
| 54 | .03 | .03 | 4.1 | 4.1 | 4.2 | .97 | .97 | 2.78 | 2.88 | 1.10 | 1.05 | 2.88 | 3.02 | 1.96 | 1.87 | .24 | .24 | .24 | .22 | .24 | .24 | .24 | .25 |
| 55 | .02 | .02 | 8.0 | 8.0 | 8.0 | 1.00 | 1.00 | 6.70 | 6.62 | 1.21 | 1.36 | 6.70 | 6.53 | 2.86 | 2.92 | .25 | .24 | .25 | .26 | .25 | .27 | .25 | .27 |
| 56 | .02 | .02 | 12.0 | 12.0 | 12.0 | 1.00 | 1.00 | 10.71 | 10.77 | 1.19 | 1.21 | 10.71 | 10.95 | 3.48 | 3.67 | .25 | .27 | .25 | .24 | .25 | .23 | .25 | .24 |

TABLE 3 - FCFS MULTI-CLASS MODEL DESCRIPTIONS

| MODEL | CLASS | NO. CUST. | CPU | | I/O 1 | | I/O 2 | | I/O 3 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | MEAN | C.V. | PROB | MEAN | PROB | MEAN | PROB | MEAN |
| 1 | 1 | 1 | 5.000 | 1.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .500 | 1.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .500 | 1.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .250 | 1.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 2 | 1 | 1 | .250 | 1.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | 5.000 | 1.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .500 | 1.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .500 | 1.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 3 | 1 | 1 | .500 | 1.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .250 | 1.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | 5.000 | 1.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .500 | 1.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 4 | 1 | 1 | .500 | 1.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .500 | 1.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .250 | 1.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | 5.000 | 1.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 5 | 1 | 1 | 5.000 | 2.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .500 | 2.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .500 | 2.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .250 | 2.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 6 | 1 | 1 | .250 | 2.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | 5.000 | 2.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .500 | 2.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .500 | 2.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 7 | 1 | 1 | .500 | 2.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .250 | 2.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | 5.000 | 2.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | .500 | 2.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |
| 8 | 1 | 1 | .500 | 2.000 | .600 | 2.000 | .200 | 1.600 | .200 | 2.667 |
| | 2 | 1 | .500 | 2.000 | .400 | 2.000 | .400 | 1.600 | .200 | 2.667 |
| | 3 | 1 | .250 | 2.000 | .200 | 2.000 | .400 | 1.600 | .400 | 2.667 |
| | 4 | 1 | 5.000 | 2.000 | .200 | 2.000 | .200 | 1.600 | .600 | 2.667 |

TABLE 4 - FCFS MULTI-CLASS MODEL RESULTS

| MODEL | CLASS | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | S.D.W.T. | | UTIL IO 1 | | IO 2 | | IO 3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | APP | SIM | Ps | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 1 | | 6.30 | 7.37 | 5.14 | .79 | .80 | 1.67 | 2.16 | | | 2.64 | 3.98 | | | .42 | .37 | .32 | .27 | .60 | .51 |
| | 1 | 8.80 | 8.13 | 10.75 | .57 | .63 | .61 | .67 | .49 | .47 | 5.33 | 5.42 | 4.94 | 5.11 | .14 | .14 | .04 | .04 | .06 | .06 |
| | 2 | 5.29 | 6.87 | 4.19 | .09 | .07 | .37 | .52 | .48 | .50 | 1.97 | 3.54 | 3.95 | 4.62 | .15 | .11 | .12 | .10 | .10 | .08 |
| | 3 | 5.90 | 7.17 | 4.62 | .08 | .07 | .35 | .50 | .48 | .50 | 2.09 | 3.60 | 4.09 | 4.71 | .07 | .06 | .11 | .10 | .18 | .15 |
| | 4 | 6.13 | 7.44 | 4.34 | .04 | .03 | .34 | .48 | .47 | .50 | 2.09 | 3.53 | 4.19 | 4.76 | .07 | .05 | .05 | .04 | .26 | .21 |
| 2 | | 6.27 | 7.26 | 5.10 | .79 | .80 | 1.67 | 2.15 | | | 2.62 | 3.90 | | | .45 | .39 | .30 | .25 | .60 | .50 |
| | 1 | 5.27 | 6.64 | 3.72 | .05 | .04 | .36 | .50 | .48 | .50 | 1.89 | 3.30 | 3.99 | 4.62 | .23 | .18 | .06 | .05 | .10 | .07 |
| | 2 | 8.71 | 7.94 | 10.60 | .57 | .62 | .61 | .66 | .49 | .47 | 5.32 | 5.26 | 4.94 | 5.06 | .09 | .10 | .07 | .08 | .06 | .07 |
| | 3 | 5.78 | 7.17 | 4.57 | .09 | .07 | .36 | .50 | .48 | .50 | 2.05 | 3.59 | 4.08 | 4.67 | .07 | .06 | .11 | .09 | .18 | .15 |
| | 4 | 6.23 | 7.41 | 4.94 | .08 | .07 | .35 | .49 | .48 | .50 | 2.16 | 3.64 | 4.21 | 4.71 | .06 | .05 | .05 | .04 | .26 | .21 |
| 3 | | 6.12 | 7.28 | 5.00 | .78 | .79 | 1.65 | 2.13 | | | 2.52 | 3.87 | | | .48 | .40 | .31 | .25 | .59 | .50 |
| | 1 | 5.47 | 6.80 | 4.36 | .09 | .07 | .36 | .51 | .48 | .50 | 1.97 | 3.45 | 3.95 | 4.63 | .22 | .18 | .06 | .05 | .10 | .08 |
| | 2 | 5.19 | 6.57 | 3.68 | .05 | .04 | .35 | .49 | .48 | .50 | 1.82 | 3.22 | 3.87 | 4.58 | .15 | .12 | .12 | .09 | .10 | .08 |
| | 3 | 9.05 | 8.35 | 11.08 | .55 | .61 | .59 | .65 | .49 | .48 | 5.32 | 5.44 | 4.94 | 5.13 | .04 | .05 | .07 | .08 | .12 | .13 |
| | 4 | 5.96 | 7.68 | 4.79 | .08 | .06 | .35 | .48 | .48 | .50 | 2.08 | 3.67 | 4.07 | 4.70 | .07 | .05 | .05 | .04 | .27 | .21 |
| 4 | | 6.02 | 7.12 | 4.94 | .76 | .78 | 1.62 | 2.13 | | | 2.45 | 3.79 | | | .49 | .40 | .33 | .28 | .57 | .48 |
| | 1 | 5.41 | 6.72 | 4.33 | .09 | .08 | .36 | .51 | .48 | .50 | 1.94 | 3.40 | 3.87 | 4.64 | .22 | .18 | .06 | .04 | .10 | .08 |
| | 2 | 5.31 | 6.62 | 4.25 | .09 | .08 | .36 | .51 | .48 | .50 | 1.91 | 3.37 | 3.83 | 4.63 | .15 | .12 | .12 | .10 | .10 | .08 |
| | 3 | 5.39 | 6.88 | 3.88 | .05 | .04 | .34 | .48 | .47 | .50 | 1.85 | 3.30 | 3.86 | 4.67 | .07 | .05 | .12 | .09 | .20 | .15 |
| | 4 | 9.46 | 8.57 | 11.58 | .53 | .60 | .56 | .63 | .50 | .48 | 5.33 | 5.43 | 4.94 | 5.21 | .04 | .05 | .03 | .04 | .17 | .18 |
| 5 | | 6.60 | 7.59 | 5.14 | .76 | .76 | 1.73 | 2.14 | | | 2.85 | 4.07 | | | .40 | .36 | .30 | .26 | .57 | .50 |
| | 1 | 9.01 | 8.22 | 10.75 | .56 | .59 | .61 | .64 | .49 | .48 | 5.51 | 5.28 | 5.08 | 9.48 | .13 | .14 | .04 | .04 | .06 | .06 |
| | 2 | 5.52 | 7.13 | 4.19 | .09 | .07 | .39 | .52 | .49 | .50 | 2.13 | 3.69 | 6.90 | 8.67 | .14 | .12 | .12 | .09 | .10 | .08 |
| | 3 | 6.21 | 7.43 | 4.62 | .08 | .07 | .37 | .50 | .48 | .50 | 2.30 | 3.75 | 7.25 | 8.83 | .06 | .05 | .10 | .09 | .17 | .15 |
| | 4 | 6.53 | 7.68 | 4.34 | .04 | .03 | .36 | .48 | .48 | .50 | 2.37 | 3.68 | 7.56 | 8.89 | .06 | .05 | .05 | .04 | .24 | .21 |
| 6 | | 6.56 | 7.78 | 5.10 | .76 | .76 | 1.73 | 2.19 | | | 2.83 | 4.27 | | | .43 | .37 | .28 | .24 | .58 | .47 |
| | 1 | 5.52 | 7.19 | 3.72 | .05 | .03 | .37 | .51 | .48 | .50 | 2.06 | 3.66 | 7.02 | 9.45 | .22 | .17 | .06 | .04 | .10 | .07 |
| | 2 | 8.92 | 8.26 | 10.60 | .56 | .61 | .62 | .66 | .49 | .47 | 5.48 | 5.44 | 5.08 | 10.27 | .09 | .10 | .07 | .07 | .06 | .07 |
| | 3 | 6.07 | 7.77 | 4.57 | .08 | .06 | .37 | .52 | .48 | .50 | 2.25 | 4.03 | 7.21 | 9.73 | .07 | .05 | .11 | .08 | .18 | .14 |
| | 4 | 6.58 | 7.99 | 4.94 | .08 | .06 | .37 | .51 | .48 | .50 | 2.41 | 4.05 | 7.51 | 9.84 | .06 | .05 | .05 | .04 | .24 | .20 |
| 7 | | 6.40 | 7.67 | 5.00 | .75 | .75 | 1.70 | 2.14 | | | 2.72 | 4.11 | | | .46 | .38 | .29 | .25 | .56 | .48 |
| | 1 | 5.73 | 7.15 | 4.36 | .09 | .07 | .38 | .51 | .48 | .50 | 2.15 | 3.65 | 6.94 | 9.31 | .21 | .17 | .06 | .04 | .09 | .08 |
| | 2 | 5.44 | 7.03 | 3.68 | .05 | .04 | .37 | .50 | .48 | .50 | 1.99 | 3.50 | 6.82 | 9.25 | .15 | .11 | .12 | .09 | .10 | .08 |
| | 3 | 9.27 | 8.63 | 11.08 | .54 | .59 | .59 | .64 | .49 | .48 | 5.50 | 5.48 | 5.08 | 10.43 | .04 | .05 | .07 | .08 | .12 | .13 |
| | 4 | 6.29 | 8.09 | 4.79 | .08 | .06 | .37 | .50 | .48 | .50 | 2.31 | 4.04 | 7.26 | 9.83 | .06 | .05 | .05 | .04 | .25 | .20 |
| 8 | | 6.30 | 7.42 | 4.94 | .74 | .74 | 1.68 | 2.09 | | | 2.65 | 3.87 | | | .47 | .38 | .32 | .26 | .54 | .49 |
| | 1 | 5.67 | 6.93 | 4.33 | .09 | .07 | .37 | .50 | .48 | .50 | 2.13 | 3.47 | 6.82 | 8.66 | .21 | .17 | .06 | .05 | .09 | .08 |
| | 2 | 5.56 | 6.85 | 4.25 | .09 | .07 | .37 | .50 | .48 | .50 | 2.08 | 3.44 | 6.73 | 8.58 | .14 | .12 | .12 | .09 | .10 | .08 |
| | 3 | 5.69 | 7.37 | 3.88 | .04 | .03 | .36 | .47 | .48 | .50 | 2.05 | 3.50 | 6.87 | 8.82 | .07 | .05 | .11 | .09 | .19 | .15 |
| | 4 | 9.69 | 8.83 | 11.58 | .52 | .56 | .57 | .61 | .50 | .49 | 5.52 | 5.41 | 5.09 | 9.88 | .04 | .04 | .03 | .04 | .17 | .18 |

12 preemptive models and all non-preemptive models. See Tables 5.5, 5.6 and 5.7. For the 36 models, the computation per model was approximately 400 milliseconds.

In addition to providing reasonable accuracy for models not in local balance, these programs give exact results for models in local balance where class coalescing is not necessary. Though the coalescing techniques do not necessarily give exact results for locally balanced models, the results are very close. In the above validation process, for all FCFS models requiring coalescing, the coalescing process was applied to a locally balanced model similar to the non-locally balanced model being studied. Individual class throughputs and utilizations were compared for the locally balanced model with and without coalescing. The differences were never more than 1% and usually less than that.

These programs are more than an order of magnitude faster than existing implementations of other techniques.

TABLE 5 - PRIORITY MODEL DESCRIPTIONS

| MODEL | CLASS | NO. CUST. | CPU MEAN | I/O 1 PROB | I/O 1 MEAN | I/O 2 PROB | I/O 2 MEAN | I/O 3 PROB | I/O 3 MEAN | I/O 4 PROB | I/O 4 MEAN |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3.333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 2 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 3 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 4 | 1 | .167 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 5 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 6 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| 2 | 1 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 2 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 3 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 4 | 1 | 3.333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 5 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 6 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| 3 | 1 | 1 | 1.000 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 2 | 1 | 2.000 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 3 | 1 | .250 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 4 | 1 | .500 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 5 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 6 | 1 | .250 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| 4 | 1 | 1 | .250 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 2 | 1 | .333 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 3 | 1 | .500 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 4 | 1 | 2.000 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 5 | 1 | 1.000 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| | 6 | 1 | .250 | .333 | 2.000 | .333 | 1.600 | .333 | 2.667 | | |
| 5 | 1 | 1 | 1.000 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 2 | 1 | .100 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 3 | 1 | .100 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 4 | 1 | .100 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 5 | 1 | .100 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 6 | 1 | .100 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| 6 | 1 | 1 | 1.000 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 2 | 1 | .667 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 3 | 1 | .083 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 4 | 1 | .083 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 5 | 1 | .083 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| | 6 | 1 | .067 | .250 | 2.000 | .250 | 1.000 | .250 | .500 | .250 | 1.000 |
| 7 | 1 | 2 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 1 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 1 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| 8 | 1 | 1 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 2 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 1 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| 9 | 1 | 1 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 1 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 2 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| 10 | 1 | 1 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 2 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 3 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| 11 | 1 | 3 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 1 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 2 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| 12 | 1 | 2 | .500 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 2 | 3 | .167 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |
| | 3 | 1 | .250 | .125 | 1.000 | .125 | 1.000 | .500 | 1.000 | .250 | 2.000 |

TABLE 6 - PREEMPTIVE MODEL RESULTS

| MODEL | CLASS | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | UTIL IO 1 | | IO 2 | | IO 3 | | IO 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | APP | SIM | PS | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 1 | | 7.04 | 7.56 | 6.16 | .62 | .69 | 1.68 | 2.23 | | | 1.97 | 2.81 | .58 | .54 | .43 | .40 | .78 | .71 | | |
| | 1 | 8.41 | 6.86 | 9.62 | .40 | .49 | .40 | .49 | .49 | .50 | 3.33 | 3.39 | .14 | .17 | .04 | .05 | .06 | .08 | | |
| | 2 | 6.03 | 6.14 | 5.12 | .06 | .05 | .29 | .34 | .45 | .47 | 1.76 | 2.06 | .13 | .13 | .11 | .10 | .09 | .09 | | |
| | 3 | 6.70 | 7.14 | 5.90 | .05 | .05 | .25 | .34 | .44 | .47 | 1.70 | 2.44 | .06 | .06 | .10 | .09 | .16 | .15 | | |
| | 4 | 7.27 | 8.12 | 6.20 | .02 | .02 | .21 | .31 | .41 | .46 | 1.53 | 2.54 | .06 | .05 | .04 | .04 | .22 | .20 | | |
| | 5 | 6.99 | 8.62 | 5.82 | .05 | .04 | .25 | .37 | .44 | .48 | 1.78 | 3.16 | .10 | .08 | .08 | .06 | .13 | .10 | | |
| | 6 | 7.26 | 9.42 | 5.82 | .05 | .03 | .27 | .38 | .45 | .49 | 1.99 | 3.59 | .09 | .07 | .07 | .06 | .12 | .09 | | |
| 2 | | 6.22 | 6.14 | 6.03 | .60 | .62 | 1.16 | 1.34 | | | 1.21 | 1.37 | .70 | .73 | .50 | .51 | .82 | .80 | | |
| | 1 | 5.27 | 4.49 | 5.33 | .06 | .07 | .06 | .07 | .24 | .26 | .33 | .33 | .23 | .27 | .06 | .08 | .10 | .11 | | |
| | 2 | 4.81 | 4.58 | 5.17 | .07 | .07 | .08 | .09 | .27 | .28 | .39 | .40 | .17 | .18 | .13 | .14 | .11 | .11 | | |
| | 3 | 5.39 | 5.13 | 5.69 | .06 | .06 | .08 | .09 | .28 | .29 | .46 | .46 | .07 | .08 | .12 | .13 | .20 | .21 | | |
| | 4 | 10.68 | 9.92 | 11.21 | .31 | .33 | .42 | .44 | .49 | .50 | 4.46 | 4.36 | .04 | .04 | .03 | .03 | .15 | .16 | | |
| | 5 | 6.83 | 8.06 | 5.73 | .05 | .04 | .25 | .31 | .43 | .46 | 1.69 | 2.51 | .10 | .09 | .08 | .07 | .13 | .11 | | |
| | 6 | 7.10 | 8.66 | 5.73 | .05 | .04 | .27 | .33 | .44 | .47 | 1.90 | 2.90 | .09 | .08 | .08 | .06 | .13 | .10 | | |
| 3 | | 7.00 | 6.92 | 6.27 | .62 | .68 | 1.67 | 1.92 | | | 1.95 | 2.22 | .61 | .64 | .43 | .41 | .77 | .75 | | |
| | 1 | 5.91 | 4.60 | 6.54 | .17 | .21 | .17 | .21 | .37 | .41 | 1.00 | .99 | .20 | .26 | .05 | .06 | .09 | .12 | | |
| | 2 | 7.31 | 6.71 | 7.92 | .27 | .30 | .39 | .42 | .49 | .49 | 2.84 | 2.80 | .11 | .11 | .09 | .09 | .07 | .08 | | |
| | 3 | 6.49 | 6.50 | 5.58 | .04 | .04 | .25 | .27 | .43 | .44 | 1.64 | 1.76 | .06 | .06 | .10 | .10 | .16 | .16 | | |
| | 4 | 8.04 | 8.45 | 6.90 | .06 | .06 | .28 | .32 | .45 | .47 | 2.24 | 2.74 | .05 | .05 | .04 | .04 | .20 | .19 | | |
| | 5 | 7.31 | 8.51 | 5.75 | .05 | .04 | .29 | .34 | .45 | .47 | 2.14 | 2.91 | .09 | .08 | .07 | .06 | .12 | .10 | | |
| | 6 | 7.35 | 9.02 | 5.54 | .03 | .03 | .29 | .35 | .45 | .48 | 2.13 | 3.19 | .09 | .08 | .07 | .06 | .12 | .09 | | |
| 4 | | 6.21 | 6.18 | 6.09 | .60 | .60 | 1.13 | 1.22 | | | 1.17 | 1.26 | .70 | .71 | .49 | .49 | .83 | .84 | | |
| | 1 | 5.17 | 4.52 | 5.11 | .05 | .06 | .05 | .06 | .21 | .23 | .25 | .26 | .23 | .27 | .06 | .07 | .10 | .12 | | |
| | 2 | 4.78 | 4.61 | 5.14 | .07 | .07 | .08 | .08 | .27 | .27 | .37 | .37 | .17 | .18 | .13 | .14 | .11 | .13 | | |
| | 3 | 5.61 | 5.45 | 6.05 | .09 | .09 | .11 | .12 | .32 | .32 | .63 | .65 | .07 | .07 | .11 | .12 | .19 | .19 | | |
| | 4 | 8.78 | 8.76 | 9.43 | .23 | .23 | .31 | .32 | .46 | .47 | 2.76 | 2.85 | .05 | .04 | .04 | .04 | .18 | .19 | | |
| | 5 | 7.76 | 8.62 | 7.04 | .13 | .12 | .32 | .35 | .47 | .48 | 2.48 | 3.01 | .09 | .08 | .07 | .07 | .11 | .10 | | |
| | 6 | 7.02 | 8.37 | 5.51 | .04 | .03 | .26 | .29 | .44 | .45 | 1.83 | 2.44 | .10 | .07 | .08 | .06 | .13 | .10 | | |
| 5 | | 2.71 | 2.79 | 2.62 | .40 | .43 | .76 | .99 | | | .34 | .46 | .88 | .87 | .57 | .55 | .29 | .28 | .63 | .60 |
| | 1 | 4.93 | 4.33 | 5.28 | .20 | .24 | .20 | .24 | .40 | .43 | 1.00 | 1.02 | .25 | .29 | .03 | .03 | .01 | .02 | .03 | .03 |
| | 2 | 2.27 | 2.20 | 2.16 | .04 | .05 | .13 | .14 | .34 | .35 | .30 | .32 | .09 | .09 | .13 | .14 | .07 | .07 | .13 | .14 |
| | 3 | 2.04 | 2.15 | 1.96 | .05 | .05 | .12 | .16 | .32 | .37 | .24 | .34 | .10 | .10 | .20 | .18 | .10 | .09 | .05 | .05 |
| | 4 | 2.47 | 2.56 | 2.37 | .04 | .04 | .10 | .15 | .31 | .36 | .26 | .39 | .10 | .11 | .05 | .04 | .03 | .03 | .25 | .24 |
| | 5 | 2.95 | 3.24 | 2.83 | .03 | .03 | .10 | .14 | .30 | .35 | .28 | .47 | .17 | .14 | .08 | .08 | .04 | .04 | .08 | .08 |
| | 6 | 2.99 | 3.33 | 2.83 | .03 | .03 | .10 | .15 | .31 | .36 | .31 | .51 | .17 | .14 | .08 | .08 | .04 | .04 | .08 | .08 |
| 6 | | 3.05 | 3.15 | 2.70 | .54 | .59 | 1.39 | 1.68 | | | .71 | .88 | .82 | .83 | .50 | .49 | .25 | .24 | .55 | .52 |
| | 1 | 4.91 | 4.02 | 5.43 | .20 | .25 | .20 | .25 | .40 | .43 | 1.00 | 1.02 | .25 | .32 | .03 | .03 | $.0_1$ | $.0_2$ | .03 | .03 |
| | 2 | 3.06 | 2.81 | 2.99 | .22 | .23 | .35 | .39 | .48 | .49 | 1.10 | 1.09 | .07 | .06 | .10 | .12 | .05 | .05 | .10 | .11 |
| | 3 | 2.35 | 2.35 | 1.89 | .04 | .04 | .26 | .27 | .44 | .44 | .62 | .64 | .09 | .08 | .17 | .17 | .09 | .08 | .04 | .04 |
| | 4 | 2.67 | 2.98 | 2.28 | .03 | .03 | .21 | .26 | .41 | .44 | .56 | .78 | .09 | .09 | .05 | .04 | .02 | .02 | .23 | .21 |
| | 5 | 3.14 | 3.63 | 2.72 | .03 | .02 | .18 | .25 | .39 | .43 | .57 | .92 | .16 | .13 | .08 | .06 | .04 | .03 | .08 | .07 |
| | 6 | 3.15 | 3.79 | 2.68 | .02 | .02 | .18 | .25 | .38 | .43 | .55 | .95 | .16 | .14 | .08 | .07 | .04 | .03 | .08 | .06 |

TABLE 6 - PREEMPTIVE MODEL RESULTS

| MODEL | CLASS | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | UTIL IO 1 | | IO 2 | | IO 3 | | IO 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | APP | SIM | PS | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 7 | | 2.71 | 2.69 | 2.61 | .52 | .54 | .95 | .99 | | | .65 | .67 | .46 | .46 | .33 | .32 | .32 | .31 | .74 | .74 |
| | 1 | 2.72 | 2.60 | 2.92 | .37 | .39 | .46 | .49 | .65 | .67 | .62 | .64 | .37 | .37 | .09 | .10 | .09 | .10 | .37 | .38 |
| | 2 | 2.57 | 2.62 | 2.28 | .06 | .06 | .21 | .22 | .41 | .41 | .55 | .58 | .05 | .05 | .19 | .18 | .05 | .04 | .19 | .19 |
| | 3 | 2.86 | 3.00 | 2.43 | .09 | .08 | .28 | .28 | .45 | .45 | .81 | .84 | .04 | .04 | .04 | .04 | .17 | .17 | .17 | .17 |
| 8 | | 2.57 | 2.54 | 2.51 | .42 | .43 | .68 | .69 | | | .43 | .44 | .34 | .35 | .49 | .48 | .34 | .34 | .78 | .78 |
| | 1 | 2.59 | 2.45 | 2.80 | .19 | .21 | .19 | .21 | .39 | .40 | .50 | .51 | .19 | .20 | .05 | .06 | .05 | .06 | .19 | .21 |
| | 2 | 2.51 | 2.51 | 2.42 | .13 | .13 | .27 | .28 | .54 | .55 | .34 | .35 | .10 | .10 | .40 | .38 | .10 | .10 | .40 | .40 |
| | 3 | 2.66 | 2.72 | 2.45 | .09 | .09 | .21 | .21 | .41 | .41 | .57 | .56 | .05 | .05 | .05 | .04 | .19 | .19 | .19 | .17 |
| 9 | | 2.59 | 2.57 | 2.54 | .45 | .46 | .74 | .76 | | | .48 | .49 | .34 | .34 | .35 | .38 | .47 | .46 | .77 | .77 |
| | 1 | 2.57 | 2.46 | 2.81 | .19 | .21 | .19 | .21 | .40 | .41 | .50 | .51 | .19 | .21 | .05 | .05 | .05 | .05 | .19 | .20 |
| | 2 | 2.35 | 2.33 | 2.30 | .07 | .07 | .13 | .14 | .34 | .35 | .31 | .33 | .05 | .05 | .21 | .23 | .05 | .05 | .21 | .20 |
| | 3 | 2.73 | 2.77 | 2.55 | .18 | .18 | .41 | .41 | .64 | .65 | .56 | .57 | .09 | .09 | .09 | .09 | .37 | .36 | .37 | .37 |
| 10 | | 3.34 | 3.34 | 3.31 | .47 | .48 | .87 | .87 | | | .48 | .49 | .34 | .34 | .47 | .46 | .54 | .54 | .90 | .90 |
| | 1 | 3.24 | 3.03 | 3.57 | .15 | .16 | .15 | .16 | .36 | .37 | .50 | .49 | .15 | .15 | .04 | .04 | .04 | .04 | .15 | .16 |
| | 2 | 3.10 | 3.12 | 3.10 | .11 | .11 | .20 | .20 | .46 | .46 | .31 | .31 | .08 | .08 | .32 | .31 | .08 | .09 | .32 | .34 |
| | 3 | 3.57 | 3.64 | 3.37 | .21 | .21 | .52 | .51 | .80 | .79 | .61 | .62 | .11 | .11 | .11 | .11 | .42 | .41 | .42 | .41 |
| 11 | | 3.57 | 3.54 | 3.42 | .61 | .62 | 1.47 | 1.48 | | | .88 | .87 | .54 | .56 | .32 | .32 | .40 | .40 | .84 | .83 |
| | 1 | 3.44 | 3.28 | 3.81 | .44 | .45 | .63 | .65 | .84 | .83 | .73 | .71 | .44 | .46 | .11 | .11 | .11 | .12 | .44 | .45 |
| | 2 | 3.37 | 3.40 | 2.92 | .05 | .05 | .23 | .23 | .42 | .42 | .77 | .78 | .04 | .04 | .15 | .14 | .04 | .04 | .15 | .14 |
| | 3 | 3.93 | 4.11 | 3.21 | .13 | .12 | .61 | .60 | .77 | .77 | 1.19 | 1.24 | .06 | .06 | .06 | .06 | .25 | .25 | .25 | .23 |
| 12 | | 3.43 | 3.44 | 3.33 | .51 | .52 | 1.08 | 1.08 | | | .62 | .62 | .44 | .44 | .55 | .55 | .32 | .31 | .87 | .88 |
| | 1 | 3.34 | 3.19 | 3.66 | .30 | .31 | .36 | .37 | .59 | .59 | .60 | .59 | .30 | .30 | .07 | .08 | .07 | .07 | .30 | .32 |
| | 2 | 3.44 | 3.50 | 3.21 | .15 | .15 | .48 | .48 | .81 | .80 | .55 | .56 | .11 | .11 | .44 | .44 | .11 | .11 | .44 | .43 |
| | 3 | 3.61 | 3.85 | 3.15 | .07 | .07 | .24 | .23 | .43 | .42 | .88 | .90 | .03 | .03 | .03 | .04 | .14 | .13 | .14 | .13 |

TABLE 7 - NON-PREEMPTIVE MODEL RESULTS

| MODEL | CLASS | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | UTIL IO 1 | | IO 2 | | IO 3 | | IO 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | APP | SIM | PS | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 1 | | 6.81 | 7.21 | 6.16 | .64 | .68 | 1.59 | 2.05 | | | 1.80 | 2.46 | .60 | .57 | .45 | .42 | .81 | .74 | | |
| | 1 | 8.16 | 7.06 | 9.62 | .41 | .47 | .42 | .49 | .49 | .50 | 3.46 | 3.45 | .15 | .16 | .04 | .05 | .07 | .07 | | |
| | 2 | 5.91 | 5.98 | 5.12 | .06 | .06 | .28 | .31 | .45 | .46 | 1.44 | 1.83 | .14 | .13 | .11 | .11 | .09 | .09 | | |
| | 3 | 6.49 | 6.71 | 5.90 | .05 | .05 | .23 | .31 | .42 | .46 | 1.38 | 2.05 | .06 | .05 | .10 | .10 | .16 | .16 | | |
| | 4 | 7.21 | 7.81 | 6.20 | .02 | .02 | .20 | .29 | .40 | .46 | 1.36 | 2.30 | .06 | .05 | .04 | .04 | .22 | .20 | | |
| | 5 | 6.66 | 7.88 | 5.82 | .05 | .04 | .22 | .32 | .42 | .47 | 1.43 | 2.52 | .10 | .09 | .08 | .07 | .13 | .11 | | |
| | 6 | 6.81 | 8.38 | 5.82 | .05 | .04 | .23 | .33 | .42 | .47 | 1.56 | 2.80 | .10 | .08 | .08 | .06 | .13 | .11 | | |
| 2 | | 6.26 | 7.07 | 6.03 | .64 | .65 | 1.27 | 1.91 | | | 1.32 | 2.26 | .69 | .61 | .49 | .43 | .82 | .74 | | |
| | 1 | 5.31 | 5.83 | 5.33 | .06 | .06 | .14 | .27 | .34 | .45 | .72 | 1.59 | .23 | .20 | .06 | .06 | .10 | .09 | | |
| | 2 | 5.25 | 6.05 | 5.17 | .06 | .06 | .15 | .28 | .36 | .45 | .87 | 1.70 | .15 | .14 | .12 | .11 | .10 | .09 | | |
| | 3 | 5.89 | 6.82 | 5.69 | .06 | .05 | .16 | .27 | .37 | .45 | .94 | 1.87 | .07 | .06 | .11 | .10 | .18 | .16 | | |
| | 4 | 9.46 | 8.42 | 11.21 | .35 | .40 | .38 | .44 | .49 | .50 | 3.70 | 3.67 | .04 | .05 | .03 | .04 | .17 | .20 | | |
| | 5 | 6.50 | 7.81 | 5.73 | .05 | .04 | .22 | .32 | .41 | .47 | 1.37 | 2.49 | .10 | .08 | .08 | .07 | .14 | .11 | | |
| | 6 | 6.66 | 8.48 | 5.73 | .05 | .04 | .23 | .33 | .42 | .47 | 1.48 | 2.82 | .10 | .08 | .08 | .06 | .13 | .11 | | |
| 3 | | 6.69 | 7.05 | 6.27 | .66 | .66 | 1.57 | 1.82 | | | 1.75 | 2.14 | .63 | .63 | .45 | .43 | .81 | .77 | | |
| | 1 | 5.85 | 6.08 | 6.54 | .17 | .16 | .23 | .30 | .42 | .46 | 1.33 | 1.83 | .21 | .21 | .05 | .05 | .09 | .09 | | |
| | 2 | 6.71 | 6.30 | 7.92 | .30 | .32 | .36 | .39 | .48 | .49 | 2.83 | 2.45 | .12 | .13 | .10 | .11 | .08 | .08 | | |
| | 3 | 6.44 | 6.71 | 5.58 | .04 | .04 | .24 | .26 | .43 | .44 | 1.40 | 1.74 | .06 | .06 | .10 | .10 | .17 | .17 | | |
| | 4 | 7.51 | 8.01 | 6.90 | .07 | .06 | .24 | .27 | .42 | .44 | 1.68 | 2.13 | .05 | .06 | .04 | .04 | .21 | .21 | | |
| | 5 | 6.89 | 7.70 | 5.75 | .05 | .04 | .25 | .31 | .44 | .46 | 1.67 | 2.35 | .10 | .09 | .08 | .07 | .13 | .11 | | |
| | 6 | 6.97 | 8.04 | 5.54 | .04 | .03 | .26 | .30 | .44 | .46 | 1.73 | 2.43 | .10 | .08 | .08 | .06 | .13 | .11 | | |
| 4 | | 6.22 | 6.48 | 6.09 | .62 | .63 | 1.23 | 1.54 | | | 1.28 | 1.66 | .69 | .65 | .49 | .47 | .84 | .81 | | |
| | 1 | 5.23 | 5.34 | 5.11 | .05 | .05 | .13 | .18 | .33 | .39 | .66 | .98 | .23 | .22 | .06 | .06 | .10 | .10 | | |
| | 2 | 5.25 | 5.48 | 5.14 | .06 | .06 | .16 | .21 | .36 | .40 | .87 | 1.13 | .15 | .14 | .12 | .12 | .10 | .10 | | |
| | 3 | 6.06 | 6.34 | 6.05 | .08 | .08 | .18 | .22 | .38 | .42 | 1.07 | 1.42 | .07 | .06 | .11 | .10 | .18 | .17 | | |
| | 4 | 8.17 | 7.56 | 9.43 | .24 | .27 | .29 | .33 | .45 | .47 | 2.42 | 2.53 | .05 | .05 | .04 | .04 | .20 | .20 | | |
| | 5 | 6.88 | 7.21 | 7.04 | .15 | .14 | .26 | .31 | .44 | .46 | 1.76 | 2.23 | .10 | .10 | .08 | .08 | .13 | .12 | | |
| | 6 | 6.67 | 7.86 | 5.51 | .04 | .03 | .23 | .28 | .42 | .45 | 1.49 | 2.22 | .10 | .08 | .08 | .07 | .13 | .12 | | |
| 5 | | 2.68 | 2.77 | 2.62 | .41 | .42 | .72 | .91 | | | .32 | .42 | .89 | .87 | .58 | .56 | .29 | .29 | .63 | .62 |
| | 1 | 4.85 | 4.32 | 5.28 | .21 | .23 | .21 | .23 | .41 | .42 | 1.03 | 1.01 | .26 | .29 | .03 | .03 | .01 | .01 | .03 | .03 |
| | 2 | 2.28 | 2.26 | 2.16 | .04 | .04 | .13 | .13 | .34 | .34 | .31 | .30 | .09 | .08 | .13 | .14 | .07 | .07 | .13 | .13 |
| | 3 | 2.02 | 2.11 | 1.96 | .05 | .05 | .11 | .14 | .32 | .35 | .23 | .30 | .10 | .09 | .20 | .19 | .10 | .09 | .05 | .05 |
| | 4 | 2.44 | 2.59 | 2.37 | .04 | .04 | .09 | .13 | .29 | .34 | .23 | .34 | .10 | .10 | .05 | .04 | .03 | .03 | .26 | .24 |
| | 5 | 2.90 | 3.10 | 2.83 | .03 | .03 | .08 | .13 | .28 | .33 | .24 | .40 | .17 | .15 | .09 | .08 | .04 | .05 | .09 | .08 |
| | 6 | 2.92 | 3.20 | 2.83 | .03 | .03 | .09 | .13 | .28 | .34 | .25 | .43 | .17 | .15 | .09 | .08 | .04 | .04 | .09 | .08 |
| 6 | | 2.95 | 3.08 | 2.70 | .56 | .60 | 1.31 | 1.67 | | | .64 | .86 | .84 | .81 | .52 | .51 | .26 | .25 | .58 | .54 |
| | 1 | 4.75 | 4.20 | 5.43 | .21 | .24 | .23 | .29 | .42 | .46 | 1.08 | 1.24 | .26 | .30 | .03 | .03 | .01 | .02 | .03 | .03 |
| | 2 | 2.83 | 2.63 | 2.99 | .24 | .25 | .33 | .36 | .47 | .48 | .96 | .95 | .07 | .07 | .11 | .11 | .05 | .06 | .11 | .11 |
| | 3 | 2.30 | 2.39 | 1.89 | .04 | .03 | .25 | .27 | .43 | .44 | .51 | .64 | .09 | .08 | .17 | .18 | .09 | .08 | .04 | .04 |
| | 4 | 2.61 | 2.83 | 2.28 | .03 | .03 | .19 | .26 | .39 | .44 | .48 | .74 | .10 | .08 | .05 | .05 | .02 | .02 | .24 | .22 |
| | 5 | 3.06 | 3.50 | 2.72 | .03 | .02 | .16 | .24 | .37 | .43 | .49 | .86 | .16 | .15 | .08 | .07 | .04 | .04 | .08 | .07 |
| | 6 | 3.07 | 3.63 | 2.68 | .02 | .02 | .16 | .25 | .36 | .43 | .48 | .90 | .16 | .13 | .08 | .07 | .04 | .04 | .08 | .07 |

TABLE 7 - NON-PREEMPTIVE MODEL RESULTS

| MODEL | CLASS | CYCLE TIME | | | CPU UTIL | | CPU M.Q.L. | | S.D.Q.L. | | CPU M.W.T. | | UTIL IO 1 | | IO 2 | | IO 3 | | IO 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | APP | SIM | PS | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM | APP | SIM |
| 7 | | 2.68 | 2.64 | 2.61 | .52 | .53 | .92 | .95 | | | .61 | .63 | .46 | .47 | .34 | .33 | .33 | .33 | .75 | .73 |
| | 1 | 2.78 | 2.68 | 2.92 | .36 | .37 | .48 | .50 | .66 | .67 | .67 | .68 | .36 | .38 | .09 | .10 | .09 | .09 | .36 | .36 |
| | 2 | 2.52 | 2.56 | 2.28 | .07 | .06 | .20 | .20 | .40 | .40 | .50 | .52 | .05 | .05 | .20 | .19 | .05 | .05 | .20 | .19 |
| | 3 | 2.65 | 2.66 | 2.43 | .09 | .10 | .23 | .24 | .42 | .43 | .62 | .64 | .05 | .04 | .05 | .04 | .19 | .19 | .19 | .18 |
| 8 | | 2.55 | 2.55 | 2.51 | .42 | .42 | .66 | .66 | | | .42 | .42 | .34 | .34 | .50 | .47 | .34 | .34 | .78 | .79 |
| | 1 | 2.67 | 2.56 | 2.80 | .19 | .20 | .21 | .22 | .41 | .42 | .56 | .57 | .19 | .19 | .05 | .05 | .05 | .05 | .19 | .19 |
| | 2 | 2.51 | 2.56 | 2.42 | .13 | .13 | .27 | .27 | .53 | .53 | .38 | .35 | .10 | .10 | .40 | .39 | .10 | .10 | .40 | .40 |
| | 3 | 2.52 | 2.54 | 2.45 | .10 | .09 | .18 | .17 | .38 | .38 | .51 | .43 | .05 | .05 | .05 | .04 | .20 | .19 | .20 | .20 |
| 9 | | 2.57 | 2.53 | 2.54 | .45 | .46 | .72 | .74 | | | .46 | .47 | .34 | .36 | .35 | .35 | .48 | .48 | .78 | .76 |
| | 1 | 2.66 | 2.50 | 2.81 | .19 | .20 | .22 | .23 | .41 | .42 | .58 | .59 | .19 | .22 | .05 | .05 | .05 | .05 | .19 | .18 |
| | 2 | 2.39 | 2.33 | 2.30 | .07 | .07 | .14 | .15 | .35 | .36 | .40 | .35 | .05 | .05 | .21 | .21 | .05 | .06 | .21 | .20 |
| | 3 | 2.63 | 2.66 | 2.55 | .19 | .18 | .35 | .36 | .59 | .60 | .52 | .47 | .10 | .09 | .10 | .09 | .38 | .38 | .38 | .38 |
| 10 | | 3.34 | 3.28 | 3.31 | .47 | .48 | .85 | .87 | | | .47 | .48 | .34 | .35 | .46 | .47 | .55 | .56 | .90 | .90 |
| | 1 | 3.34 | 3.12 | 3.57 | .15 | .16 | .18 | .19 | .38 | .39 | .59 | .59 | .15 | .16 | .04 | .04 | .04 | .04 | .15 | .15 |
| | 2 | 3.16 | 3.12 | 3.10 | .11 | .11 | .23 | .23 | .49 | .49 | .40 | .36 | .08 | .09 | .32 | .32 | .08 | .08 | .32 | .32 |
| | 3 | 3.46 | 3.46 | 3.37 | .22 | .22 | .45 | .45 | .73 | .73 | .57 | .52 | .11 | .11 | .11 | .11 | .43 | .44 | .43 | .43 |
| 11 | | 3.54 | 3.53 | 3.42 | .61 | .62 | 1.41 | 1.44 | | | .83 | .85 | .53 | .54 | .32 | .33 | .42 | .42 | .85 | .84 |
| | 1 | 3.53 | 3.41 | 3.81 | .43 | .44 | .67 | .69 | .85 | .85 | .79 | .79 | .43 | .43 | .11 | .11 | .11 | .11 | .43 | .44 |
| | 2 | 3.33 | 3.36 | 2.92 | .05 | .05 | .22 | .22 | .41 | .42 | .71 | .75 | .04 | .04 | .15 | .15 | .04 | .04 | .15 | .14 |
| | 3 | 3.66 | 3.83 | 3.21 | .14 | .13 | .52 | .52 | .72 | .73 | .93 | 1.00 | .07 | .07 | .07 | .07 | .27 | .27 | .27 | .26 |
| 12 | | 3.41 | 3.42 | 3.33 | .51 | .52 | 1.04 | 1.05 | | | .59 | .60 | .44 | .45 | .55 | .55 | .33 | .32 | .88 | .89 |
| | 1 | 3.42 | 3.33 | 3.66 | .29 | .31 | .38 | .40 | .60 | .61 | .66 | .67 | .29 | .30 | .07 | .08 | .07 | .07 | .29 | .30 |
| | 2 | 3.40 | 3.44 | 3.21 | .15 | .14 | .45 | .45 | .77 | .76 | .52 | .52 | .11 | .11 | .44 | .44 | .11 | .11 | .44 | .45 |
| | 3 | 3.38 | 3.52 | 3.15 | .07 | .07 | .20 | .20 | .40 | .40 | .67 | .69 | .04 | .04 | .04 | .03 | .15 | .14 | .15 | .14 |

# CHAPTER VI

# SIMULATION OF GENERALIZED QUEUEING NETWORKS

## 6.1 Introduction

Queueing network models can be used to analyze characteristics of computing systems such as scheduling disciplines with priority and/or preemption, non-exponential service time distributions, customer dependent behavior and contention for memory, channels and other resources (B2,B4,F1, K1, Chapter V). Though much progress has been made in using algebraic or numerical techniques to find solutions or approximate solutions for these complex models (B2,B4,C2,K1,Chapter V), simulation is more general than other solution techniques. Confidence intervals for simulation results are very important. Confidence intervals for simulation results for a very large class of queueing network models can be determined using the techniques of Crane and Iglehart (C6,C7). We have developed a versatile simulator incorporating confidence interval analysis; this simulator and the extensions proposed here provide the computer system designer/analyst with powerful new tools.

In section 6.2 we review the techniques of Crane and Iglehart, in Section 6.3 we present a description of the simulator we have implemented, and in section 6.4 we discuss the extension of the existing simulator to the general models described above. In section 6.5 we present a language, QUASCI (QUeueing Analysis by Simulation with Confidence Intervals), based on the language QAL (F1,M2). Only models for which the confidence interval techniques are valid may be expressed in QUASCI; the language is designed to prevent incorrect application of the confidence interval techniques. QUASCI is also

appropriate for model description for solution packages using nonsimulation techniques. This is significant in the application of queueing network models to computer system analysis.

## 6.2 Confidence Intervals -- The Crane-Iglehart Technique

Confidence intervals (M3) may be used to indicate the accuracy of simulation results. We can say with a certain level of confidence, say 90%, that the result of a simulation will lie within an interval, say (a,b). In other words, if we run many simulations, the results of 90% of the simulations will be in the interval (a,b).

Crane and Iglehart have developed confidence interval techniques for simulations of Markovian models with a single chain. We will assume for now that the state space of the model is finite or countably infinite; these techniques may also be applied to other models (C7,L1). The techniques are based on many replications of "tours", a tour being defined as the period between two successive returns to a designated state. The simulation need not simulate the Markov process directly, but it must be able to determine when the system returns to the designated state. Crane and Iglehart show that the expected length of the confidence intervals, given a fixed simulation run length, is independent of the state chosen to define the tours. However, if the state chosen is such that the tours are very long relative to the total simulation run length, then few tours will be replicated and the confidence interval analysis will not be valid. So we should attempt to choose a frequently entered state to define the tours. We should also choose states that are simply defined so that overhead of testing for the state is not too great.

We illustrate application of the techniques in determining confidence intervals for server utilization. Application to other model statistics is similar. Server utilization can be determined by dividing the simulated time during which the server is busy by the total simulated time. Equivalently, we can determine utilization as $\overline{B}/\overline{T}$, where $B_i$ is the time the server is busy during tour i, and $T_i$ is the length of tour i. For g in the interval (0,1) we can determine an approximate 100(1 - g) percent confidence interval for utilization (C6) as: $\left( \frac{\overline{BT} - ks_{12} - \sqrt{D}}{\overline{T}^2 - ks_{22}} , \frac{\overline{BT} - ks_{12} + \sqrt{D}}{\overline{T}^2 - ks_{22}} \right)$, where k, $s_{12}$, $s_{22}$ and D are defined as follows:

Let $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-n^2/2}\, dn$ , $z_x = \Phi^{-1}(x)$ and let n be the number of tours, then $k = \frac{z_1^2 - g/2}{n}$ . $s_{12} = \frac{n}{n-1} (E[BT] - \overline{BT})$. $s_{22} = \frac{n}{n-1} (E[T^2] - \overline{T}^2)$.

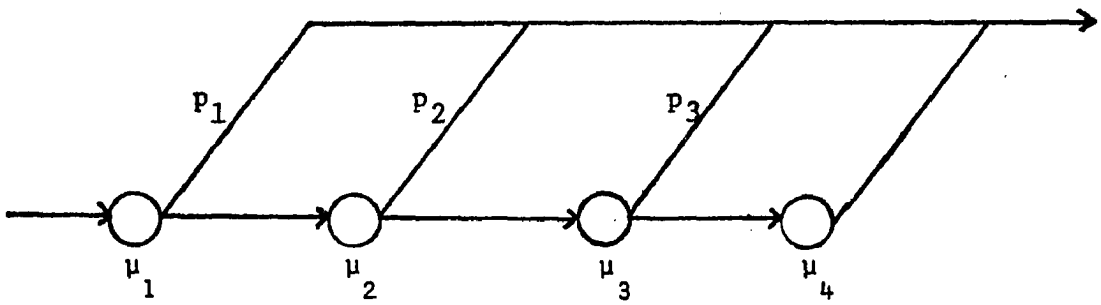
Let $s_{11} = \frac{n}{n-1} (E[B^2] - \overline{B}^2)$, then $D = (\overline{BT} - ks_{12})^2 - (\overline{B}^2 - ks_{11})(\overline{T}^2 - ks_{22})$.

## 6.3 APLOMB - A Simulator for Closed Queueing Networks

We have constructed a queueing network simulator employing the Crane-Iglehart techniques. This simulator exists as a set of Fortran subroutines. The user provides the routines with a definition of the model, criteria for acceptable confidence intervals, and a short routine which is called to determine whether the simulated system is in the tour defining state.

The networks simulated by APLOMB may have several different classes of customers. Each queue may have one of a variety of queueing disciplines, including FCFS and priority disciplines. The existing simulator assumes a single server at each queue, but may be easily extended to allow multiple

identical servers. Service distributions of generalized Erlang form (Figure 6.1) proposed by Cox (C5, Chapter IV, Chapter V) are assumed; the service times may be class dependent. Customers leaving a queue may be routed




$\mu_i$ - rate of exponential stage i
$p_i$ - probability of bypassing stages after stage i

Figure 6.1

to any queue in the network according to fixed probabilities. These probabilities may be dependent on the customer class and the queue being left. The state of the system is determined by the number of customers of each class in each queue, the ordering of customers in each queue, and the current distribution stage for each customer. This system will have a finite state space.

The simulator structure is driven by an event list. An event occurs each time a customer completes a stage of its service distribution. After each event the user supplied routine is called to determine whether the system is in the tour defining state or not. If the system is not in the tour defining state, the simulation continues. If the system is in the tour defining state, the accumulators used in the confidence interval analysis are updated. If a sufficient number of tours have been replicated, confidence

intervals are determined. If the confidence intervals are satisfactory, the simulation is terminated, otherwise additional replications are made until satisfactory intervals are obtained.

This simulator has been used to determine results for over 125 computer system models (Chapter V). The results of these simulations are in agreement with those obtained by analytic approximation techniques.

## 6.4 Extension to Open Networks, Mixed Networks and Passive Servers

The existing simulator may be easily extended to include open and mixed networks. Open networks have sources which emit customers and sinks which absorb customers leaving the network. Mixed networks are open for some classes of customers and closed for other classes. If we represent the time between arrivals from a source by a distribution of the Cox form (Figure 6.1) and make other restrictions as with closed networks, then the system will have a countably infinite state space. The state of the system is determined by the distribution stage of each source, and the same conditions which determine the state of the closed network. In addition to events occurring after customers complete service stages, events must occur when a source distribution stage is completed.

Passive servers are a construction which has been included in queueing network models to consider the effects of blocking in computer systems for resources such as memory, channels and peripheral processors. Customers must acquire units of the server before they may traverse certain parts of the network. If the units are not available, the customer must wait in a queue. When a customer leaves the restricted portion of the network, all units of the passive server are released by the possessing customer and a

queued customer may acquire these units. We assume that the number of units required by a customer is described by a finite probability mass function, and dependent only on the server and the customer class. The state of a system will be determined by the above mentioned conditions, by the allocations of passive servers to each customer, the number of customers of each class in each queue for a passive server, and the ordering of customers in these queues. The state space will be at most countably infinite. Since passive servers will only be affected when customers leave a source or an active server, the same event definitions described above can be used.

## 6.5 QUASCI

QUASCI is a high level language very similar to the language QAL (F1,M2). We have restricted the features of QAL to allow only models which are compatible with the confidence interval techniques, and have added new features to facilitate use of the confidence interval techniques. First we present an example illustrating some of the features of the language, then informally present the syntax and semantics of the language.

### 6.5.1 An Example

Figure 6.2 illustrates a simple model of a computing system. Customers arriving at the system must wait in a queue until allocated space in memory. After receiving memory, the customers alternately request service from the central processing unit (CPU) and from an input/output (I/O) device. After several cycles of CPU and I/O services, the customers release their portion of memory and leave the system. Figure 6.3 gives a QUASCI description of this model.

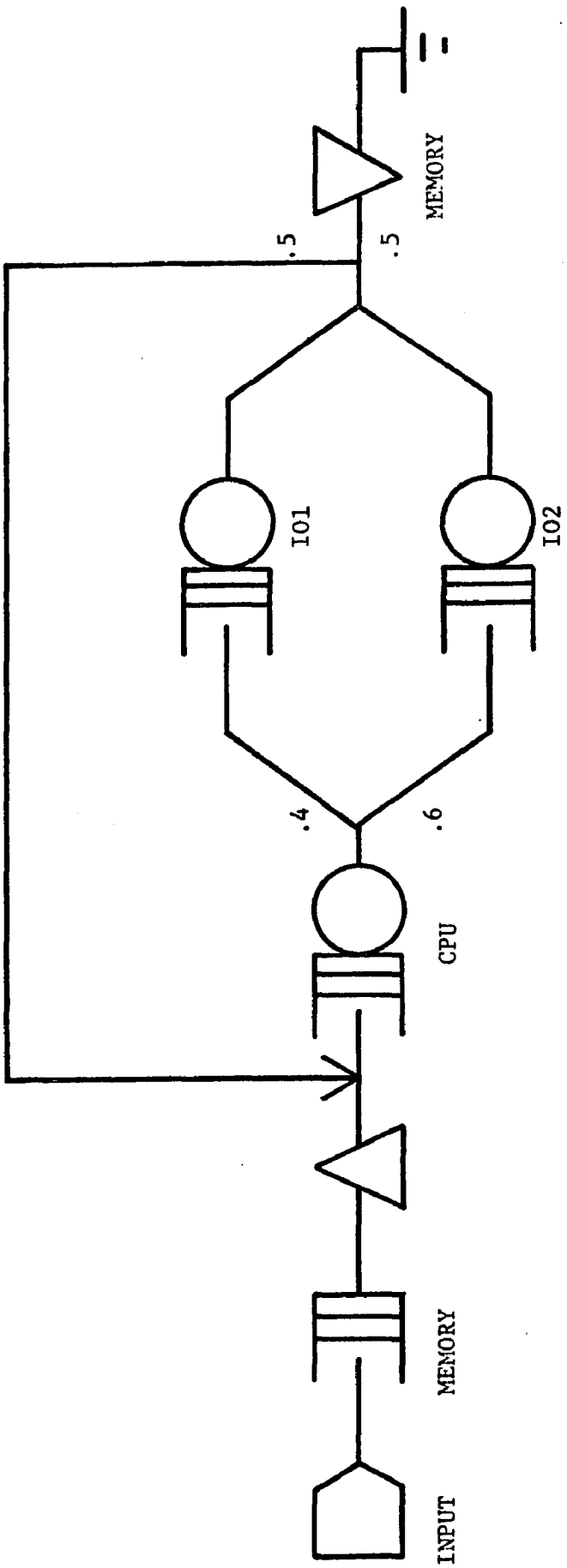



Figure 6.2

```
NETWORK
       1 (1.) (ALLOCATE, MEMORY)
         2    (1.) (CPU)
              3    (.4) (IO1)
              3    (.6) (IO2)
                   4    (.5) (GO,2)
                   4    (.5) (RELEASE, MEMORY)
                        5    (1.) (SINK)
END

SERVERS
       MEMORY, PASSIVE = 4, REQUEST = 1 $
       CPU, DISTRIBUTION = STANDARD(1.,5.), DISCIPLINE = PS,
            TOLERANCE = (1.,.05,.2,.2)  $
       IO1, DISTRIBUTION = STANDARD(2.,.5) $
       IO2, DISTRIBUTION = STANDARD(3.,.5) $
END

SOURCES
       INPUT, DISTRIBUTION = STANDARD(10.,1.), ENTRY POINT = MEMORY $
END

CUSTOMERS
       CPU = (2) $
END

TOURS
       MEMORY, LENGTH = 0 $ CPU, LENGTH = (2) $
       IO1,    LENGTH = 0 $ IO2, LENGTH =  0  $
END

SIMULATE
       BATCH, CONFIDENCE = 95
END
```

Figure 6.3

The NETWORK statement describes the interconnections of the model elements. This statement consists of several "levels". Each level consists of a level number, a traversal probability, and either a network element or a "GO" element. Customers leaving a network element proceed to a level with the next higher level number, unless directed to a specific level or element by a "GO". When there are several levels to choose from, the choice is determined by the associated traversal probabilities.

The SERVERS statement gives the characteristics of each of the servers in the model and their associated queues. Several characteristics are not specified; default values are assumed. MEMORY is a passive server with a total of four units available; customers always request 1 unit. The queueing discipline is FCFS by default, and no criteria are specified for the confidence intervals for the statistics associated with MEMORY. The CPU is an active server (as opposed to a passive server -- the CPU is a server in the traditional sense). The distribution of service requests is a standardized form (Chapter V) with mean of 1 and coefficient of variation 5. The queueing discipline is Processor Sharing (PS), the limiting case of a no overhead round-robin discipline as the quantum approaches zero. TOLERANCE specifies maximum lengths for the confidence intervals for throughput, utilization, queue length and wait time, respectively, at the CPU.

The SOURCES statement specifies the name of the source, INPUT, the distribution for interarrival times from the source, and the place where customers enter the network. Customers arriving from the source are of class 0, by default.

The CUSTOMERS statement specifies that there are to be two class 0 customers at the CPU when simulation begins.

The TOURS statement specifies the conditions which determine the tour defining state. In this example, all queues are empty in the tour defining state, except for the CPU queue, which must have 2 customers. Both customers at the CPU must be in the first stage of their service distribution, by default. Also by default, the source must be in the first stage of its distribution. Notice that the system is in the tour defining state initially.

The SIMULATE statement specifies that simulation is to be initiated, and the confidence level is to be 95%.

### 6.5.2 Syntax and Semantics of QUASCI

QUASCI, like QAL, is intended to be embedded in a high level algorithmic language such as FORTRAN, PASCAL or PL/1. QUASCI programs may take advantage of host language facilities for communicating with the operating system, for iteration, for computing values of variables, etc. The interaction of QUASCI with the host language is more restricted than that of QAL. The primary difference is that host language expressions within QUASCI statements are not evaluated repeatedly during the simulation, but rather evaluated when the QUASCI statement is executed. This is necessary; otherwise, we could not easily guarantee that the Markov process for the simulated system is properly defined. This restriction also removes one of the primary implementation difficulties of QAL.

There are seven statements in the language, SIMULATE, NETWORK, SERVERS, TOURS, CUSTOMERS, SOURCES and SINKS. All programs must include the first three and either CUSTOMERS or SOURCES or both. TOURS must be included unless the system is an open network. We will now discuss these statements in the above order, but first describe notation. Braces { } enclose required items, and brackets [ ] enclose optional items. When there are several lines within braces or brackets, any one line may be used for the required or optional item. Underlined values are used as default values where no item is specified. Capitalized words denote keywords. Where several orderings of keyword items are possible, all orderings are equivalent. An ellipsis (...) represents repetition of the preceeding form.

Figure 6.4 describes the SIMULATE statement. The <name> is used to identify the entire simulation, and must be a valid identifier in the host

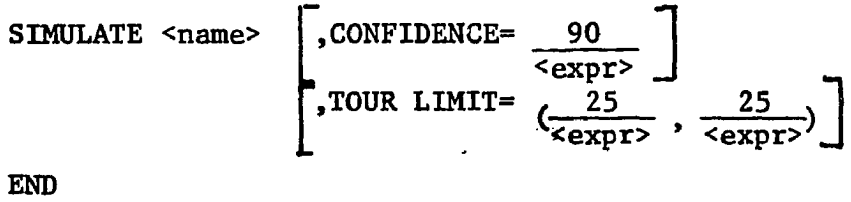



Figure 6.4

language. CONFIDENCE sets the level of confidence for the simulation statistics. The item <expr> denotes a scalar expression in the host language. The first value in parentheses for the TOUR LIMIT sets the minimum number of tours considered necessary for valid confidence intervals. The second parenthesized value sets the number of tours to be replicated before rechecking the confidence intervals.

Figure 6.5 describes the NETWORK statement. The item <level #>

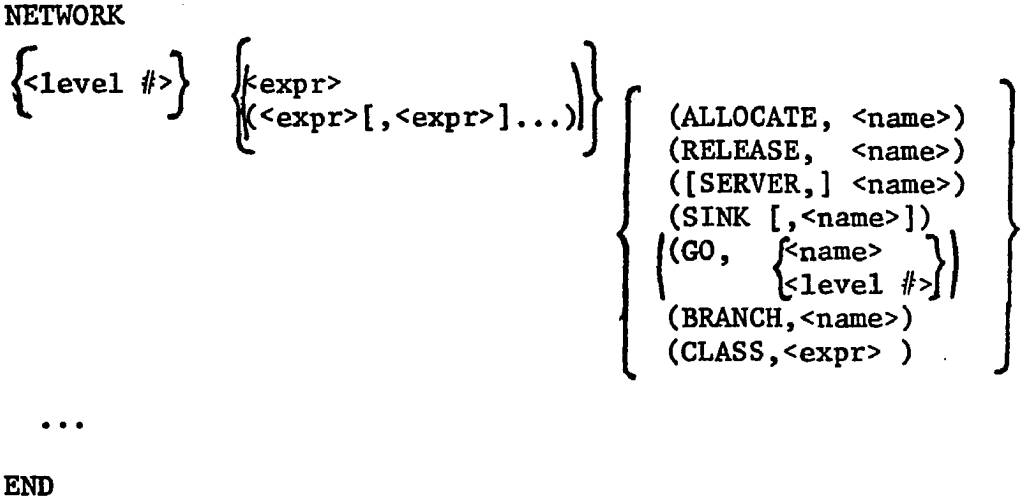



Figure 6.5

must be an unsigned integer. The parenthesized expressions in the second

part of the level give traversal probabilities. The singly nested form gives class independent probabilities, while the doubly nested form gives probabilities for each class from class 0 through the highest class number occurring in this model. (QUASCI allows only single digit class numbers for clarity in the TOURS statement. From experience with APLOMB, it is doubtful that simulation analysis of models with more than ten classes would be tractable.) The items allowed for the third part of the level are generally self explanatory. The BRANCH item is intended as a labeled "dummy" node for convenience in describing complex routings. The CLASS item changes the class of a customer to the value given in the expression.

Syntax of the SERVERS statement is shown in Figure 6.6.

```
SERVERS
<name>
{ [[(<expr>)]] [ ,ACTIVE= <expr> ] , DISTRIBUTION= { <dist>                    }
                                                   { (<dist> [,<dist>] ...)    }
    ,PASSIVE=<expr> ,REQUEST= { <fpmf>                    }
                              { (<fpmf>[,<fpmf>]...)      }                    }
  [ ,DISCIPLINE= FCFS                                     ]
                 PRIORITY(<expr>)
                 PS
                 LCFSPR
                 FF                                       ]
  [ ,TOLERANCE= (<expr>,<expr>,<expr>,<expr>)             ]
  [ ,STATISTICS= GENERAL                                  ]
                 NONE
                 FULL                                     ]
  $
  ...
END
```

Figure 6.6

This figure shows the description possibilities for one server or set of identical servers. Several descriptions may be included in a single statement, with each description terminated by a dollar sign. The first line in the major braces is for active servers, the second for passive servers.

For active servers, the expression in parentheses gives the number of identical servers. The expression after ACTIVE= gives the rate of each server. In the distribution description for the server, we specify whether the distribution is class independent or class dependent. The form with a single expression is for class independent distributions. The parenthesized form is for class dependent distributions. There are several options for <dist>. The primary two are a standardized version of the Cox form (Chapter V), STANDARD(<expr>,<expr>), where the first expression is the mean and the second is the coefficient of variation, and the general Cox form, COX(<expr>, <rates>,<prob>), where the expression gives the number of exponential stages, <rates> is a vector of rates for the stages, and <prob> is a vector of bypassing probabilities for the stages. We may allow other distributions consisting of networks of exponential stages. However, this complicates implementation and adds little generality (Chapter IV). Including distributions which are not representable by a finite number of exponential stages, for example the uniform distribution, does add considerable generality, but also requires restrictions in the tour definitions. If we have such distributions, then the state space will not be countable. In choosing states for tour definition, we are restricted to those states which have tour lengths with finite first and second moments (C7,L1).

With passive servers, the expression after PASSIVE= gives the total number of units available. We specify the number of units for each request

after PASSIVE=. This may be class independent or dependent as with active server distributions. The item <fpmf> may either be an expression, in which case the number of units requested is a constant, or a finite probability mass function expressed as PMF((<expr>,<expr>),(<expr>,<expr>),...), where the first expression of the pair is the probability of the value of the second expression.

Several options are shown for the queueing discipline. The default discipline, first come first served (FCFS) is appropriate for either active or passive servers. The expression after the PRIORITY discipline gives the preemption distance (H2). This discipline is appropriate for passive servers only if it is non-preemptive. PS is appropriate only for active servers as is Last Come First Served Preemptive Resume (LCFSPR). First Fit (FF) is appropriate only for passive servers. FF is similar to FCFS, but when the first customer in the queue requests more units than are available, other customers with smaller requests may be allocated units. Other disciplines may also be added to the language.

As previously mentioned, the expressions in the TOLERANCE description are maximum lengths for the confidence intervals for throughput, utilization, mean queue length and mean wait time, respectively. These lengths are for class independent statistics. Three options are allowed for the statistics gathered; class independent statistics (GENERAL), no statistics (NONE) and class dependent as well as class independent statistics (FULL).

Figure 6.7 shows the form of the TOURS statement. Tour descriptions are

```
TOURS
      <name> [,LENGTH= 0
                       (<expr>[,<expr>]...)]
             [,QUEUE= {[1
                       [(<digit>[<digit>]...)]<digit> [(STAGE= 1
                                                          <expr>)]
                                                      [(<name>= 0
                                                          <expr>)]
             [,STAGE= 1
                      <expr>]                                  ...}...]
      $ ...
END
```

Figure 6.7

appropriate only to servers and sources. The LENGTH and QUEUE keywords are appropriate only to servers, while the STAGE keyword outside of the QUEUE section is appropriate only to sources. Where the total queue length is non-zero, lengths for each class must be specified. If more than one class of customers is present in the queue, and the discipline is not PS or PRIORITY with preemption distance 1, then the ordering of customers in the queue must be specified with the QUEUE section. The customers are specified in order from first to last. The number in parentheses is the number of customers of the class specified by the digit following the parentheses. The STAGE keyword in parentheses is for the current distribution stage of those customers. The <name> in parentheses is the name of a passive server, and the expression gives the number of units of the server that are held. The STAGE keyword for sources is for the current distribution stage for that source. Note that the syntax does not prevent the user from defining a model with multiple chains or using a transient state for tour definition. These are

considered semantic errors and an implementation of the language should attempt to check for such errors before simulation is initiated. For example, if the network is such that customers may initially be in a queue to which they cannot return, if deadlocks may occur, or if customers may change from one class to another but not reverse the class change, then transient states can exist.

The CUSTOMERS, SOURCES and SINKS statement are described in Figures 6.8, 6.9, and 6.10. The CUSTOMERS statement may be used to place

```
CUSTOMERS
        <name> = (<expr>[,<expr>]...)  $  ...
END
```

Figure 6.8

```
SOURCES
      <name>  {,DISTRIBUTION= <dist>}
              [{,ENTRY POINT= <name>   }]
              [{             <level #> }]
              [,CLASS= 0               ]
              [        <expr>          ]
              [,STATISTICS= GENERAL    ]
              [             NONE       ]
       $  ...
END
```

Figure 6.9

```
SINKS
    <name>  [,STATISTICS= GENERAL ]
            [             NONE    ]
            [             FULL    ]
     $  ...
END
```

Figure 6.10

customers initially at different servers in the network. This is necessary for closed networks and useful in open networks. The distribution options for sources are similar to those for servers, but are restricted to forms with a finite number of exponential stages.

Finally, we summarize the differences between QUASCI and QAL. We ignore minor differences, such as substitution of one keyword for another.

First, the language QUASCI is more restricted than QAL, both in terms of features allowed and in terms of semantics. The restrictions are generally necessary to guarantee that the confidence interval techniques may be applied. Expressions in QUASCI statements may not change value during simulation, as may those in QAL. QUASCI cannot allow source distributions which are not representable by finite networks of exponential stages. Control of routing in the network is limited to fixed probabilities in QUASCI, while QAL allows very general predicates to control routing and allows the predicates to change during simulation. QUASCI does not allow customers to create subtasks, as does QAL. QUASCI does not separate queues from servers as does QAL, nor does QUASCI allow the flexibility of server definition that QAL allows. QUASCI does not allow passive servers to be consumed or created. QAL permits user definition of queueing disciplines and other simulation constructs, but QUASCI does not.

Second, QUASCI includes features not found in QAL, such as tour definition, which enable convenient use of confidence interval analysis. As another extension, QUASCI distinguishes between customer classes by digits instead of names and includes class distinctions as an integral part of the

syntax of the language; QAL requires more user effort in the specification of class dependent behavior. Liu (L4) has extended the syntax of QAL in a similar manner.

# CHAPTER VII

## SUMMARY AND RECOMMENDATIONS

We have presented an approach to configuration design of computing systems. We have also presented techniques useful in the implementation of this approach.

We have shown that efficient optimization procedures may be applied to a large class of open queueing networks with different classes of customers. In many situations these open queueing networks may be used as models of computing systems, communication networks and computer networks. We recommend that an existing computer program, such as Hogarth's (H3), be extended to include this class of networks. Extension of our results to closed networks would be very useful, though this appears to be a difficult problem.

Our algorithms for numerical solution of closed queueing networks enable inexpensive parametric analysis of realistic models of computing systems. These algorithms are also valuable in the approximate analysis of more complex models; our approximate analysis techniques for central server models are economical and suitable for analysis of large parameter spaces of configurations. We thoroughly validated our approximations with simulation results for over 125 models. These numerical and approximation techniques are compatible with the techniques for models with passive resources such as memory (B4,K1); computer programs combining our techniques with these previous techniques would be very valuable to the computer system designer.

The characteristics of the models we analyze correspond to developments in computer systems and results of measurement studies. Measurement

has shown that service time distributions are often non-exponential; our models allow a general class of service time distributions. Our models allow multiple identical processors; there is a trend toward architectures with multiprocessing and our models are appropriate for analysis of these models. Empirical studies show that different programs have different service characteristics. Our models represent program dependent behvaior by using different classes of programs. Priority scheduling is widely used in computer systems and computer networks; our models allow preemptive and non-preemptive priorities.

Simulation studies continue to be important in computer system evaluation. Simulation techniques have been used casually in the past and simulation results have been viewed with skepticism for this reason. Our work has helped to formalize simulation technique and provide tools and theory which allow confidence in simulation results.

We have shown that the confidence interval simulation techniques of Crane and Iglehart (C6,C7) may be applied to a very large class of general models of computing systems. We have presented a language, QUASCI, designed to facilitate correct application of these simulation techniques. A simulation implementation of this language would also be extremely valuable to computer system designers.

QUASCI may also be used to represent models soluble by non-simulation techniques. We suggest that this language be used as a general modeling language. It should be possible to implement this language so that the user need not specify the solution technique to be used. In such an implementation, the user would specify the model and the results required;

the implementation itself would then determine which solution technique would be appropriate to these user specifications.

Since the computer system designer will usually be interested in a parameter space of models, we would recommend research to extend and implement our language to allow the user to specify a parameter space of models and criteria for selection of optimal models. This extended language implementation would be responsible for searching the parameter space and reporting to the user the optimal model or models.

## BIBLIOGRAPHY


B1 Baskett, F. Mathematical Models of Multiprogrammed Computer Systems. TSN-17, Computation Center, The University of Texas at Austin, (1971).

B2 Baskett, F., Chandy, K. M., Muntz, R. R., and Palacios-Gomez, F. "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", to appear JACM (1975).

B3 Bell, C. G. and Newell, A. Computer Structures: Readings and Examples. McGraw-Hill Book Company, 1971.

B4 Brown, R. M. An Analytic Model of a Large Scale Interactive System Including the Effects of Finite Main Memory. M. A. Thesis, Department of Computer Sciences, University of Texas at Austin, (1974).

B5 Browne, J. C., Chandy, K. M., Brown, R. M., Keller, T. K., Towsley, D. and Dissly, C. W. "Hierarchical Techniques for Development of Realistic Models of Complex Computer Systems", to appear IEEE Transactions on Computers, (1975).

B6 Buzen, J. Queueing Network Models of Multiprogramming. Ph.D. Dissertation, Division of Engineering and Applied Physics, Harvard University, (1971).

C1 Chandy, K. M. "The Analysis and Solutions for General Queueing Networks", Proceedings Sixth Annual Princeton Conference on Information Sciences, Princeton University, (1972).

C2 Chandy, K. M., Herzog, U., and Woo, L. "Approximate Analysis of General Queueing Networks", IBM Journal of Research and Development, (January, 1975).

C3 Chandy, K. M., Herzog, U., and Woo, L. "Parametric Analysis of Queueing Network Models", IBM Journal of Research and Development, (January 1975).

C4 Chandy, K. M., Howard, J. H. and Towsley, D. F. "Station Balance", submitted, JACM (1975).

C5 Cox, D. R. "A Use of Complex Probabilities in the Theory of Stochastic Processes", Proceedings Cambridge Philosophical Society 51 (1955), pp. 313-319.

C6 Crane, M. A. and Iglehart, D. I. "Simulation of Stable Stochastic Systems I: General Multiserver Queues", JACM 21, 1 (1974) pp. 103-113.

C7 Crane, M. A. and Iglehart, D. I. "Simulation of Stable Stochastic Systems II: Markov Chains", JACM 21, 1 (1974), pp. 114-123.

C8 Courtois, P. J. and Georges, J. "On a Single Server Finite Queueing Model with State Dependent Arrival and Service Processes", Operations Research, (1971) pp. 424-434.

D1 Drake, A. W. Fundamentals of Applied Probability Theory. McGraw-Hill Book Company, 1967.

F1 Foster, D. V., McGehearty, P. F., Sauer, C. H. and Waggoner, C. N. "A Language for Analysis of Queueing Models", Proceedings Fifth Annual Pittsburgh Modeling and Simulation Conference, University of Pittsburgh, (1974).

F2 Foster, D. V. File Assignment in Memory Hierarchies, Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin, (1975).

G1 Gaver, D. P. "Probability Models for Multiprogrammed Computer Systems", JACM 14, 3 (1967) pp. 423-438.

G2 Gaver, D. P. and Shedler, G. S. "Approximate Models for Processor Utilization in Multiprogrammed Computer Systems", SIAM Journal of Computing 2, 3 (1973) pp. 183-192.

G3 Gordon, W. J. and Newell, G. F. "Closed Queueing Systems with Exponential Servers", Operations Research 15, (1967) pp. 254-265.

H1 Herzog, U., Woo, L. and Chandy, K. M. "Solution of Queueing Problems by a Recursive Technique", to appear IBM Journal of Research and Development.

H2 Herzog, U. "Efficient Priority Strategies for Switching Centers in Communication Networks", Proceedings Second Texas Conference on Computing Systems, University of Texas at Austin, (1973).

H3 Hogarth, J. Optimization and Analysis of Queueing Networks. Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin, (1975).

H4 Hu, T. C. Integer Programming and Network Flows. Addison-Wesley, 1969.

I1 Irani, K. B. and Wallace, V. L. "On Network Linguistics and the Conversational Design of Queueing Networks", JACM 18, 4 (1971) pp. 616-629.

J1 Jackson, J. R. "Jobshop-like Queueing Systems", Management Science 10, 1 (1963) pp. 131-142.

J2 Johnson, D. S. A Process-Oriented Model of Resource Demands in Large, Multiprocessing Computer Utilities. TSN-29, Computation Center, The University of Texas at Austin, (1972).

K1 Keller, T. W. and Chandy, K. M. "Computer Models with Constrained Parallel Processors", Proceedings 1974 Sagamore Conference on Parallel Processing.

K2 Kleinrock, L. Communication Nets. McGraw-Hill Book Company, 1964.

K3 Kobayashi, H. "Applications of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions", JACM 21, 2 (1974) pp. 316-328.

K4 Kobayashi, H. "Applications of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling", JACM 21, 3 (1974) pp. 459-469.

L1 Lavenberg, S. S. Efficient Estimation Via Simulation of Work-Rates in Closed Queueing Networks. RJ 1390, IBM Research Laboratory, San Jose, California, (1974).

L2 Lee, C. C. Queueing Models of Device Utilization in Multiprogrammed Computer Systems, TR-7, Department of Computer Sciences, The University of Texas at Austin, (December 1972).

L3 Little, J. D. C. "A Proof for the Queueing Formula L = λω", Operations Research 9 (1966) pp. 383-387.

L4 Liu, C. A. forthcoming M.A. Thesis, Cepartment of Computer Sciences, University of Texas at Austin.

M1 Martin, J. Design of Real-Time Computer Systems. Prentice-Hall, 1967.

M2 McGehearty, P. F. QSIM, An Implementation of a Language for Analysis of Queueing Models. M. A. Thesis, Department of Computer Sciences, University of Texas at Austin, (1974).

M3 Mood, A. M. and Graybill, F. A. Introduction to the Theory of Statistics. McGraw-Hill Book Company, 1963.

M4 Morse, P. M. Queues, Inventories and Maintenance. John Wiley and Sons, 1958.

M5 Muntz, R. R. Poisson Departure Processes and Queueing Networks, IBM Research Report RC-4145 (1972).

P1 Peebles, R. W. "A Homogeneous Network of Computers for Data Sharing", Lecture presented Department of Computer Sciences, University of Texas at Austin, October 16, 1974.

R1 Reiser, M. and Kobayashi, H. "Queueing Networks with Several Closed Subchains: Theory and Computational Algorithms", to appear IBM Journal of Research and Development.

R2 Rockafellar, R. T. Convex Analysis. Princeton University Press, 1970.

S1 Sauer, C. H. and Chandy, K. M. "Approximate Analysis of Central Server Models", to appear IBM Journal of Research and Development, (1975).

S2 Sauer, C. H. "Simulation Analysis of Generalized Queueing Networks", to appear Proceedings Summer Computer Simulation Conference, (1975).

S3 Shedler, G. S. A Cyclic Queue Model of A Paging Machine. IBM Research Report, RC 2814, Yorktown Heights, New York. (March 1970).

S4 Smith, J. L. "An Analysis of Time Sharing Computer Systems Using Markov Models", Proceedings Spring Joint Computer Conference, (1966).

T1 Towsley, D. F. Queueing Networks with State Dependent Branching Probabilities. Forthcoming Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin.

W1 Wallace, V. L. and Rosenberg, R. S. "Markovian Models and Numerical Analysis of Computer System Behavior", Proceedings Spring Joint Computer Conference, (1966).

W2 Williams, A. C. and Bhandiwad, R. Private communication.

W3 Wolfe, P. "Methods of Nonlinear Programming", in Abadie, J., editor, Nonlinear Programming. John Wiley and Sons, 1967.

W4 Wyszewianski, R. J. Feedback Queues in the Modeling of Computer Systems: A Survey. TR 74-1, Department of Industrial and Operations Engineering, University of Michigan, (1974).

# VITA

Charles Herbert Sauer was born in Betheseda, Maryland on August 14, 1947, the son of Doris Johnson Sauer and Herbert Irvin Sauer. After graduation from Hickman High School, Columbia, Missouri, in 1965, he entered Carleton College, Northfield, Minnesota. He continued his studies at the University of Missouri, Columbia, Missouri, in 1966 and at San Francisco State College, San Francisco, California, in 1967-68. After an interruption to pursue musical interests, he continued studies at the University of Texas at Austin, Austin, Texas, in 1969, receiving the degree of Bachelor of Arts with a major in mathematics from the University of Texas at Austin in December 1970. In September 1971 he entered the Graduate School of the University of Texas at Austin.

Permanent address: 1635 Highridge Circle
Columbia, Missouri 65201

This dissertation was typed by Ann M. Patterson.