Approximate Algorithms for Solving Queueing Network Models of Large Distributed Systems

Hamid Reza Bahadori

Abstract

A major challenge for the immediate future in the field of distributed computing systems is to realize their potential benefits without incurring intolerable penalties in performance, complexity, and cost. There is therefore a substantial interest in establishing sound theoretical foundations for the modeling, design, construction, and reliable maintenance of these systems. Analytic modeling for performance evaluation of distributed systems typically requires the solution of large multichain queueing network models. The size of these large models precludes any use of exact solution techniques. Thus, it is important to develop accurate and cost effective algorithms for the approximate solution of large multichain queueing networks. To meet the challenges of future complex distributed computing systems, approximation techniques will have to be employed.

The work presented in this dissertation was motivated by our interest in analyzing and solving queueing network models of large distributed systems. This dissertation focuses on the development of accurate and cost effective algorithms for the approximate solution of large multichain product form queueing networks, and in particular those that represent the models of large distributed computing systems. The results obtained in this dissertation are applied to a variety of configuration design issues and other distributed systems problems. The work presented here can be used as a platform upon which future queueing networks modeling tools can be constructed.

Acknowledgements

I wish to express my utmost gratitude and appreciation to Professor Domenico Ferrari for his continuous guidance, encouragement, and support during the course of my studies at U.C. Berkeley. I would like to thank Professor Chittor Ramamoorthy and Professor Kjell A. Doksum for reading my dissertation and serving on my dissertation committee. I would like to acknowledge Professor Jean Walrand for his numerous suggestions. I especially want to thank Professor Luis Felipe Cabrera for many valuable discussions.

I have benefited from very pleasant interactions with members of the Progres group. I especially wish to thank, Shin-Yuan Tzou, P. Venkat Rangan, Stuart Sechrest, Mark Sullivan, and Peter Danzig for their technical advice and many fruitful discussions. I have benefited greatly from the technical advice and numerous discussions with my friend Harry I. Rubin.

This work was supported in part by NSF under grant DMC-8503575.

I have greatly enjoyed the company of my friends Homayoon Ansari, Farhad Ayrom, and Hormoz Yaghutiel. Finally, I would like to thank my parents and Lisa for their care, patience, and the warmest feelings during my work at Berkeley and the writing of this dissertation.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The widespread use and popularity of distributed systems is fundamentally due to the price-performance evolution in the area of microelectronics and the development of fast and cost effective communication networks. The continual improvement in computing price/performance ratios has reduced the need to adopt large centralized systems in order to realize economies of scale. In recent years we have witnessed a proliferation of local area network-based distributed systems. Very large distributed systems (VLDS) based on fast wide-area networks are already in the design stages in some research organizations. In these systems , resources such as processing power, databases, and a multitude of software services and products are shared among the users and jobs located at different sites. A good example of a VLDS operating system project is the DASH Project currently under way at Berkeley [And87].

Extensibility and reliability are among the most important potential advantages of distributed systems. Extensibility is the ability to adapt easily to both short and long term changes without significant disruption of service on the part of the system. Distributed computing systems are potentially reliable, because the redundancy and autonomy present in them permit partial failures to be masked or localized, thereby allowing the rest of the system to continue functioning. Additionally, services and software resources in a distributed environment are able to migrate to other machines in the case of a failure of the host on which they reside.

A major challenge in distributed computing systems research is to realize these potential benefits without incurring intolerable penalties in complexity, cost, and performance. Consequently, there is currently a substantial interest in establishing sound theoretical foundations for modeling, design, construction, and reliable maintenance of VLDS. We will not achieve the full potential of distributed computing systems until effective solutions to these fundamental problems are found. The growing complexity of distributed systems and the rapidly increasing number and diversity of the tasks that they are being used to perform have made it increasingly important that quantitative tools be devised to aid in the modeling, configuration, planning, design, and performance evaluation of these systems. The intuition of even the very best system designers is no longer sufficient to predict reliably how changes in the workload or configuration of a distributed system can affect the overall system performance. Among the most effective tools for modeling and performance evaluation of computing systems are discrete

event simulations and analytical modeling. Simulations are popular since they have the capability of representing more details of the systems; however they are costly and difficult to construct and implement (particularly for large distributed systems). Analytical modeling, on the other hand, is several orders of magnitude cheaper than simulation. Analytical modeling for performance evaluation of large distributed systems typically requires the solution of large queueing networks with large populations and large numbers of classes for which any exact solution is practically impossible.

The growth in the size of the workload can be exemplified by large transaction-oriented business systems such as banking systems, airline reservation systems, sales and inventory systems, telecommunications networks, and other large distributed systems that are currently being designed. In each of these cases hundreds or thousands of terminals might be active at any given time. Each active terminal in these systems is represented by a customer in the queueing network model. Therefore, the queueing network models representing these systems may contain large populations. As a result of this increase in the populations and the growing complexity of the services provided by these systems, we naturally expect that the number of customer classes required to represent them accurately in the queueing network model will also increase. Users in different geographical locations use different sets of resources (processors, printers, services, and communication lines) in different ways. Hence, if one is interested in accurately modeling the load on the system, customer classes should be used to reflect these different patterns of usage.

Thus, it is important to develop accurate and cost effective approximate algorithms for the solution of large multichain queueing networks. The successful application of queueing network models to the analysis of computer systems has been demonstrated by a number of case studies (examples can be found in [Bar78], [Gra78], [Den78], and [Kie79]). All the studies have proven the utility of these models to predict accurately the performance of the corresponding computing systems. Most of these studies, however, have been applied to centralized systems. These systems generally require models with only a few service centers, customers, and job classes. The recent proliferation of distributed systems has created a growing interest in the application of multiple-class queueing network models to these systems. For example, Goldberg et al. [Gol83] constructed and validated a model of the LOCUS distributed system and reported to be working on some tools for the design of distributed system configurations. de Souza e Silva and Gerla [Sil84] recently applied multichain queueing network models to the problem of optimal load sharing in a LOCUS distributed system environment comprised of a set of heterogeneous processors. Bester et al. [Bes84] studied supporting replicated file systems and scalability aspects of this important problem using a dual priority multichain approximate MVA (mean value analysis) model. They applied and validated their models to different configurations of a 17 site LOCUS distributed system, and reported to be working on models of the buffer contention problems in LOCUS.

These examples are clear evidence that the process of modeling and performance evaluation is becoming one of the principal elements that must be taken into consideration in the design, configuration, development, and tuning of computer systems. In this respect, queueing networks have become an indispensable part of this process. The key challenge in computer performance evaluation is the development of new quantitative methods and tools, which can keep pace with the emerging new generations of systems brought on by the advances in technology.

The work presented in this dissertation was motivated by our interest in analyzing and solving queueing network models of large distributed systems. This dissertation focuses on the development of accurate and cost effective algorithms for the approximate solution of large multichain product form queueing network models, and in particular those that represent the models of large distributed computing systems. All the algorithms proposed in this dissertation have been implemented and used throughout the course of this research. These algorithms can be easily integrated as interactive performance modeling tools for very large distributed systems.

1.2 Organization of the Thesis

This thesis is organized as follows. In chapter 2, we study the general properties of closed multiple-class product form queueing networks. Exact solution techniques for these networks and their computational complexities are discussed first. These results are then extended to the class of mixed multichain networks. The MVA equations for mixed multichain networks and the steady state probability distribution for open multichain networks are obtained. The computational complexities of solving these networks, using the corresponding solution techniques, are then derived.

In chapter 3 we give a comprehensive overview of the existing approximate algorithms for the solution of product form queueing networks. The advantages and shortcomings of these algorithms are discussed and compared.

In chapter 4 we propose a new approximate technique, based on the MVA equation for the solution of large multichain queueing network models with fixed rate single server and infinite server service centers. Such properties of this approximate algorithm as existence, uniqueness, convergence, and computational complexities are studied in detail. The accuracy of the algorithm is empirically studied, and its possible extensions to a multilevel approximate algorithm are also discussed. The accuracy and the performance of this approximation are then compared with those of some existing algorithms.

In chapter 5 we propose a hierarchical network transformation based approximate technique for the solution of large multichain queueing network models with fixed rate single server and infinite server service centers. This technique is based on a hierarchy of network transformations. The resulting iterative algorithm for this technique is analyzed. The computational complexities of the algorithm are derived in detail. The algorithm is asymptotically correct and tends to the exact solution as the population increases. Experimental results concerning the accuracy of the approximation are presented.

We have identified several applications of the results obtained in chapters 2, 4, and 5. In chapter 6 we investigate some of these applications. A decoupling algorithm for the modeling and configuration design of large distributed systems is proposed. This algorithm has also applications in the hybrid simulation of models of large distributed systems. Experimental results show the algorithm to be efficient and cost effective. Another potential application, that to bottleneck detection in large queueing network models, is then studied. We introduce a general algorithm to identify the bottlenecks in multiple-class queueing network models using the approximate MVA-based algorithms developed in chapter 4.

In the final chapter, chapter 7, the main points of this dissertation are summarized and future research directions are suggested.

The notation we have adopted and the definitions of the parameters used in this dissertation can be found in Appendix A, where the reader can find the meanings of the symbols that are not defined in the text.

Chapter 2

Product Form Queueing Networks and Their Properties

2. Introduction

In this chapter we will look at the steady state solution of a subset of the class of queueing network models. The queueing networks in this subset are referred to as multiple-class product form queueing networks (alternatively, multiple-class separable queueing networks). In the first section, we describe the characteristics of these networks and their steady state solution using the convolution algorithm. In the second section, we outline the mean value analysis algorithm for solving such networks. In section 2.3, the computational complexities of exact solution techniques for closed product form queueing networks are studied. We then turn to the properties of mixed multichain queueing networks in sections 2.4 and 2.5. The mean value equations for mixed multichain networks consisting only of *fixed rate* (a fixed rate service center is one with a mean service rate $\mu_s(i)=c$ independent of the number of jobs in the service center) single server and infinite server (IS) service centers are derived, and the computational complexities of the MVA-based solution of mixed networks are evaluated. Finally, in sections 2.6 and 2.7, the solution of multichain open networks is reviewed, and the computational complexity of that solution is derived.

2.1 The Convolution Algorithm

Consider a closed multichain queueing network containing an arbitrary but finite number M of service centers, and an arbitrary but finite number R of different classes of customers (jobs). Customers move through the network and change classes according to transition probabilities. Therefore a customer of class i, after completing service at service center j, will move to service center s while becoming a customer of type r with probability $p_{ij,rs}$. The transition matrix $P = [p_{ij,rs}]$ defines a Markov chain whose states are labeled by the pairs (r,s), $r=1,2,...,R$, $s=1,2,...,M$. The Markov chain is assumed to be decomposable into m ergodic subchains. Two job states belong to the same subchain if there is a non-zero probability that a job will be in both job states during its life in the network.

The service centers can have one the following service disciplines.

(1) **FCFS:** customers are served in the order they arrive; multiple servers are allowed. The service time distribution for all customers is the same, and of exponential type. The service rates of the servers can be dependent on the number of customers at the center.

(2) **PS:** there is a single server at the station, and the discipline is processor sharing. The service times of different job classes can have any distribution with a rational Laplace transform. The service rates of the servers can depend on the number of jobs at the center.

(3) **LCFSPR:** the discipline is last-come-first-served with preemptive resume. The service center can include multiple servers. The service time of different job classes can have any distribution with a rational Laplace transform. The service rate of the servers can depend on the number of jobs at the center.

(4) **IS:** there are as many servers as there are customers. The service times of different job classes can have any distribution with a rational Laplace transform.

We will use $\mathbf{K}$ as an argument of the quantities related to the given queueing network with the population vector $\mathbf{K} = (K_1, K_2, \ldots, K_R)$. We also denote that queueing network by $Q(\mathbf{K})$. The balance equations for $Q(\mathbf{K})$ are given by:

$$\theta_{rs} = p_{o,rs} + \sum_{i=1}^{R}\sum_{j=1}^{M}\theta_{ij}p_{ij,rs} , \quad r = 1,2,\ldots,R , \quad s = 1,2,\ldots,M . \tag{2.1.1}$$

where θ_{rs} is the relative frequency of visits of a class r job to service center s.

Suppose that the queueing network is a *non class-hopping* closed network (i.e., a network in which jobs do not change class when moving from a service center to the next), and that the routing matrix P_r is irreducible over $S(r)$. Then, from standard Markov chain analysis, we can write:

$$\bar{\Theta}_r = \bar{\Theta}_r P_r , \quad r = 1,2,\ldots,R , \tag{2.1.2}$$

where $\bar{\Theta}_r = (\theta_{r1}, \theta_{r2}, \ldots, \theta_{rM})$. To solve equation (2.1.2), we first note that $\sum_{j=1}^{M}\theta_{rj} = 1$. Let E_r represent an $M \times M$ matrix such that:

$$E_r = [\, e_{ij} \,], \quad e_{ij}=1 \text{ for } all\ ij , \quad r = 1,2,\ldots,R . \tag{2.1.3}$$

Thus, we can write:

$$\bar{\Theta}_r \times E_r = 1 , \quad r = 1,2,\ldots,R , \tag{2.1.4}$$

where 1 is the M-dimensional row vector with all elements equal to 1. By adding equations (2.1.2) and (2.1.4) and simplifying the results we can write:

$$\bar{\Theta}_r = 1\left[P_r + E_r - I_M \right]^{-1} , \quad r = 1,2,\ldots,R , \tag{2.1.5}$$

where I_M is the $M \times M$ identity matrix. The values of the visit ratios are obtained by solving equation (2.1.5) for each job class.

We also define $\mathbf{k} = (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M)$ to be the state vector of the network $Q(\mathbf{K})$, where $\mathbf{k}_s = (k_{1s}, k_{2s}, \ldots, k_{Rs})$ is the state vector of service center s. k_{rs} is the population of class r jobs at service center s. If k_s represents the total number of jobs of all classes at service center s, then:

$$k_s = \sum_{j=1}^{R} k_{js} , \quad s = 1,2,\ldots,M ; \tag{2.1.6}$$

from these definitions we can also write:

$$\mathbf{k}_1 + \mathbf{k}_2 + \cdots + \mathbf{k}_M = \mathbf{K} . \tag{2.1.7}$$

Theorem 2.1.1 [Bas75, Rei75]:

The equilibrium probability of the state $\mathbf{k} = (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M)$ is given by:

$$p(\mathbf{k}) = p(\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M) = \frac{1}{G(\mathbf{K})} \prod_{s=1}^{M} f_s(\mathbf{k}_s) , \tag{2.1.8}$$

where:

$$f_s(\mathbf{k}_s) = \frac{k_s!}{\prod_{i=1}^{k_s} \mu_s(i)} \prod_{j=1}^{R} \frac{1}{k_{js}!} (\theta_{js} s_{js})^{k_{js}} , \tag{2.1.9}$$

for the service centers of types 1 (FCFS), 2 (LCFS), or 3 (PS). Observe that in equation (2.1.9) for the type 1 service centers, $s_{js} = s_s$ for all $j \varepsilon R(s)$.

We also have:

$$f_s(\mathbf{k}_s) = \prod_{j=1}^{R} \frac{1}{k_{js}!} (\theta_{js} s_{js})^{k_{js}} , \quad s = 1,2,\ldots,M , \tag{2.1.10}$$

for the service centers of the type 4 (IS). For IS service center s, $\mu_s(i) = i$. Thus, $\prod_{i=1}^{k_s} \mu_s(i) = k_s!$, and (2.1.9) reduces to (2.1.10).

$G(K_1, K_2, \ldots, K_R)$ in (2.1.8) is the normalizing constant chosen so that:

$$\sum_{\mathbf{k} \varepsilon S(\mathbf{K}, M)} p(\mathbf{k}) = 1 . \tag{2.1.11}$$

From equation (2.1.8), and using the relationships in equations (2.1.9) and (2.1.10), a direct relationship between the *normalizing constant* and the parameters of the network can be obtained:

$$G(\mathbf{K}) = \sum_{\mathbf{k} \varepsilon S(\mathbf{K}, M)} \prod_{s=1}^{M} f_s(\mathbf{k}_s) . \tag{2.1.12}$$

There are recursive algorithms to calculate the normalizing constant [Bru78]. Once the normalizing constant is known, the following relationship [Cha80] can be used to calculate the mean throughput rates and then the other performance indices.

$$\lambda_{rs} = \theta_{rs} \frac{G(K_1, K_2, \ldots, K_r - 1, \ldots, K_R)}{G(K_1, K_2, \ldots, K_R)} , \quad r = 1,2,\ldots,R , \quad s = 1,2,\ldots,M. \tag{2.1.13}$$

From the relationship given in (2.1.13), the following important relationship is easily deduced.

$$\frac{\lambda_{rs}}{\lambda_{rs'}} = \frac{\theta_{rs}}{\theta_{rs'}} , \quad r=1,2,\ldots,R , \quad s, s' = 1,2,\ldots,M . \tag{2.1.14}$$

The method for evaluating the normalizing constant and the equilibrium state probability distribution is referred to as the *convolution algorithm*. The convolution algorithm differs from the mean value analysis (MVA) method summarized in section 2.2 in that the convolution algorithm provides the steady state probability distribution, while the MVA provides only the first moment of the performance measures.

2.2 The Mean Value Analysis (MVA) Algorithm

Reiser and Lavenberg [Rei80] have provided another approach to the solution of product form networks. They derived recursive relationships for the calculation of the mean value of the desired performance indices. The MVA algorithm was first developed for product form closed QNs that include fixed rate and queue length dependent rate servers. However, the algorithm has been extended to include a broader range of QNs, including state dependent routing and more general forms of state dependent service rates [Rei81], [Sau83]. Higher moment recursive formulae for a class of multichain product form queueing networks can be found in [Hef82].

The following is the main result of the mean value analysis (MVA) approach due to Reiser and Lavenberg. The steady state mean waiting times $W_{rs}(\mathbf{K})$ at the service centers of $Q(\mathbf{K})$, which are all single server (i.e., M_s=1 for *all* s), are related to other quantities of Q ($\mathbf{K}-e_r$) by:

$$W_{rs}(\mathbf{K}) = s_{rs}\left[1 + N_s(\mathbf{K}-e_r) + \sum_{i=1}^{K} i\left(\frac{1}{\mu_s(i)} - 1\right)p_s(i-1,\mathbf{K}-e_r)\right]. \tag{2.2.1}$$

If service center s is a type 1, 2, or 3 with unit service rate ($\mu_s(i)$=1), then:

$$W_{rs}(\mathbf{K}) = s_{rs}(1 + \sum_{j=1}^{R} N_{js}(\mathbf{K} - e_r)), \tag{2.2.2}$$

and, if the service center is a type 4 (IS service center), then $W_{rs} = s_{rs}$. It can be shown that if the mean service rate of service center s is a constant, i.e., $\mu_s(i) = c$, then (2.2.2) still applies, but with s_{rs} replaced by $\frac{s_{rs}}{c}$.

We now derive the set of recursive relations that are used for the computation of the mean values of the performance indices. We first rewrite the mean value equations (2.2.2) as:

$$W_{rs}(\mathbf{K}) = \begin{cases} s_{rs}(1 + \sum_{j=1}^{R} N_{js}(\mathbf{K} - e_r)) & \text{for } \textit{type } 1, 2, \textit{or } 3 \textit{ centers}, M_s=1, \mu_s(i)=1 \\ s_{rs} & \textit{type } 4 \textit{ (IS) centers}, \end{cases} \tag{2.2.3}$$

From Little's law [Lit61]

$$N_{rs}(\mathbf{K}) = \lambda_{rs}(\mathbf{K})W_{rs}(\mathbf{K}), \quad r = 1,2,...,R, \quad s = 1,2,...,M. \tag{2.2.4}$$

Let D_{rs} be the class r job cycle time at service center s (the cycle time is the average time between successive arrivals of the same class r job at service center s). A class r job visits queue s' an average of $\theta_{rs'}/\theta_{rs}$ times for each visit to service center s. The average time spent in service center s' for each visit to s by the same class r job is therefore $W_{rs'}(\mathbf{K})\theta_{rs'}/\theta_{rs}$. Thus:

$$D_{rs}(\mathbf{K}) = \sum_{s'=1}^{M} W_{rs'}(\mathbf{K})\theta_{rs'}/\theta_{rs}, \quad r = 1,2,...,R, \quad s = 1,2,...,M. \tag{2.2.5}$$

Little's law applied to a closed chain r results in[1]:

$$K_r = \lambda_{rs}(\mathbf{K})D_{rs}(\mathbf{K}). \tag{2.2.6}$$

From equations (2.2.4) - (2.2.6) we can write:

$$N_{rs}(\mathbf{K}) = K_r \frac{\theta_{rs} W_{rs}(\mathbf{K})}{\sum_{s'=1}^{M} \theta_{rs'} W_{rs'}(\mathbf{K})}, \quad r = 1,2,...,R, \quad s = 1,2,...,M. \tag{2.2.7}$$

1 Note that the right hand side of (2.2.6) is independent of s.

The utilization of service center s (non IS service centers) for class r jobs can be determined by:

$$\rho_{rs}(\mathbf{K}) = \lambda_{rs}(\mathbf{K})s_{rs} \ , \quad r = 1,2,...,R \ , \quad s = 1,2,...,M \ . \tag{2.2.8}$$

Recursive applications of equations (2.2.3) - (2.2.8) provide the mean values for the performance indices of all job classes at all the service centers.

2.3 Computational Complexities of Exact Solution Techniques

The existence of computational algorithms is the strong point of closed product form queueing network solutions. All the exact solution techniques for these networks are based on the two major approaches summarized in sections 2.1 and 2.2. These two approaches are alternative ways to exploit the recursive structure of the product form solution. Variations of the two methods are found in [Cha80, Lam83]. The convolution algorithm for product-form queueing networks was first proposed by Buzen [Buz73] for single-class networks, and extended by Reiser and Kobayashi [Rei75] to multiple-class networks. When the class population sizes become large, the normalizing constant $G(K_1,K_2,\ldots,K_R)$ may exceed the floating point range of the processor being used, thereby causing an overflow or underflow condition. A dynamic scaling technique has been introduced that partially alleviates these problems [Lam82].

The MVA algorithm is conceptually simpler than the convolution algorithm, since it avoids the evaluation of the normalizing constant. Although overflow problems are not present in this algorithm, underflow may still occur when solving load dependent networks. For large populations, due to the recursive nature of the MVA algorithm, non-negligible round-off errors can also occur. The critical problem with these two algorithms, however, is their space/time complexity. The computation of visit ratios is an essential requirement for both the MVA and the convolution algorithm.

The evaluation of visit ratios is solely dependent on the queueing network topology and the branching probability matrix. The time complexity involved in the computation of equation (2.1.5) can be derived by first computing the time complexity for each class and multiplying the result by the total number of classes R. We need $M(M-1)$ additions to perform the matrix operations $P_r + E_r - I_M$ (we disregard the additions of zeros resulting from the fact that the diagonal elements of the matrix $E_r - I_M$ are all zero). We need M^3 multiplications/divisions and $M^3 - 2M^2 + M$ additions and subtractions to compute the inverse of $P_r + E_r - I_M$ [Bur85]. Finally, to compute the visit ratios, we need M^2 additions; since we are evaluating the product of the vector 1 by $\left[P_r + E_r - I_M\right]^{-1}$ and all the elements of 1 are unity, we only need M additions per element of the resulting $1 \times M$ vector, for a total of M^2 additions.

Hence, the total time complexity of the visit ratio evaluation for each class is $2M^3$. The time complexity for all classes is therefore $V = 2RM^3$. In the derivation of this complexity, all the chains in the network were assumed to be closed. We will show later that the time complexity for evaluating the visit ratios for the open chains of a QN is slightly higher than that for the closed chains. We then need RM multiplications to evaluate the utilizations for all classes. The derivation of the time complexity to compute the normalizing constant, the mean waiting times, the mean queue lengths, and the mean throughput rates can similarly be carried out, and is discussed in [Bal77, Zah80].

We restrict the storage space required to compute the visit ratios to that needed to retain the final values of the visit ratios. The actual space needed for the matrix additions and inversions is not included in the calculations. Indeed, solving a QN consists of two steps. In the first step the visit ratios are evaluated, in the second the performance measures are computed using one of the solution algorithms. The space needed to solve a multichain QN with large chain populations is many orders of magnitude larger than that needed to compute the visit ratios. Additionally, the visit ratios have to be evaluated before any QN solution can be obtained. Hence, once a solution of a QN is practically possible and the total storage for its solution has been allotted, we can always use (part or all of) this storage in the first step to solve for the visit ratios. When the visit ratios are known, this storage can then be reused for solving the QN. There are trivial cases (not applying to the types of networks considered

here) such as single-class networks and certain special case multiple-class open networks in which the storage space needed to evaluate the visit ratios may be larger than that needed to solve the network. In these cases the maximum space needed should be separately assigned. We will discuss some of these cases in section 2.7. The total space needed to retain the visit ratios is RM. The additional space required to store the utilization values is RM. The storage needed to compute the normalizing constant, the mean waiting times, the mean queue length, and the mean throughput rates can similarly be derived [Bal77, Zah80]. Table 1 shows the space/time complexity of the two approaches for the load independent case. The time complexities for the evaluation of the throughput rates, the mean queue lengths, and the normalizing constant are taken from [Bal77, Zah80].

Table 1. Space/Time Complexity (Load Independent Servers)

Complexity	Mean Value Analysis	Convolution
Time	RM(4X+3)+V	(4RM-2R)X+R(6M-2)+V
Space	(R+M+RM)Y+4RM	2X+5RM

$$X = \prod_{r=1}^{R}(K_r+1) \qquad Y = \prod_{\substack{i=1 \\ i \neq l}}^{R}(K_i+1) \qquad V = 2RM^3$$

Note that, in Table 1, R is the number of classes, M is the number of service centers, K_r, $r = 1,2,...,R$, is the total number of class r customers in the network, and l is the chain visiting the largest number of service centers. Table 2 shows the computational complexities for load dependent product form networks [Bal77, Zah80].

Table 2. Space/Time Complexity (Load Dependent Servers)

Complexity	Mean Value Analysis	Convolution
Time	3RM(XK+2X+4/3)/2+2MK+V	M(2X+2RX+2R+4Z)+4(X-Z)+RM+V
Space	MK(Y+1)+RY(M+1)+4RM	4X+5RM

$$X = \prod_{r=1}^{R}(K_r+1), \qquad Z = \frac{1}{2}\prod_{r=1}^{R}(K_r+1)\,(K_r+2), \qquad K = \sum_{r=1}^{R}K_r \qquad V = 2RM^3.$$

A network is referred to here as *load dependent* if the service rates at all the service centers are load dependent. Therefore, for networks with some load dependent and some load independent service rates, the computational complexities given in Table 2 serve as upper bounds, and those given in Table 1 as lower bounds. Clearly, the time and space complexities of the load dependent networks for both algorithms are substantially higher than those for the load independent ones. The space complexities for mean value analysis in both the load dependent and the load independent cases are upper bounds.

A closer look at Table 1 and Table 2 reveals the enormous space and time complexities that make the use of exact solution techniques virtually impossible even for relatively small numbers of classes and service centers. For instance, to evaluate the mean values of the performance measures for all chains for a load independent queueing network with $R=10$, $K_r=40$, $r=1,2...,10$, and $M=10$ about 3.274×10^{12} Mbytes of memory and 5.369×10^{18} floating point operations are needed. It would take a Cray X-MP/48 processor running at maximum speed 3868.3 years to solve exactly this network (assuming the required memory is available).

2.4 Mixed Multichain Networks

Mixed networks are networks that consist of both open and closed chains. In this section we assume that O represents the set of open chains (classes), and C the set of closed chains. Furthermore, we assume that the sets O and C have respective cardinalities R^o and R^c ($R^o = |O|$, $R^c = |C|$, and $R = R^o + R^c$). Let us further assume, without loss of generality, that the chains in the network are so indexed that $C = \{1,2,...,R^c\}$, and $O = \{R^c+1,R^c+2,...,R\}$. We will use the superscripts o and c to distinguish between the open and the closed chains of a mixed network. The speed of the servers is assumed to be fixed and independent of the service center states, and the total external arrival rate λ to be independent of the network state. λ_{rs}^o is the throughput rate of open chain r at service center s. $\mathbf{K}_c = (K_{R^o+1},K_{R^o+2},\ldots,K_R)$ is the population vector of the closed chains, and $\Lambda_o = (\lambda_{o,1},\lambda_{o,2},\ldots,\lambda_{o,R^o})$ is the external arrival rate vector. $\lambda_{o,rs}$ is the external arrival rate of open chain r at service center s. The following relationship holds for the external arrival rate vector components (note that external arrivals are assumed to be Poisson processes):

$$\lambda_{o,r} = \sum_{s \varepsilon S_o(r)} \lambda_{o,rs} , \quad r \varepsilon O. \tag{2.4.1}$$

Let $p_{o,rs}$ be the probability of an external class r job arrival at service center s. Then for the open chains, the following balance equations hold:

$$\theta_{rs}^o = p_{o,rs} + \sum_{i=1}^{R^o} \sum_{j=1}^{M} \theta_{ij}^o p_{ij,rs} , \quad r \varepsilon O, \quad s=1,2,...,M. \tag{2.4.2}$$

In equation (2.4.2) it is assumed that $\sum_{s \varepsilon S_o(r)} p_{o,rs} = 1$. It can be shown that the set of equations (2.4.2) always have a unique solution. If we assume that open chains constitute a non class-hopping cluster (i.e., that none of the jobs in any of the open chains change class), then this set of equations becomes a set of R^o matrix equations of the form:

$$\overline{\Theta}_r^o = \mathbf{P}_{o,r} + \overline{\Theta}_r^o P_r , \quad r \varepsilon O , \tag{2.4.3}$$

where $\mathbf{P}_{o,r}$ is given by:

$$\mathbf{P}_{o,r} = (p_{o,r1},p_{o,r2},\ldots,p_{o,rM}) , \quad r \varepsilon O. \tag{2.4.4}$$

The set of equations given in (2.4.3) can be solved by solving the following equivalent set of equations:

$$\overline{\Theta}_r^o = \mathbf{P}_{o,r} \left[I_M - P_r \right]^{-1} , \quad r \varepsilon O , \tag{2.4.5}$$

where I_M is the $M \times M$ identity matrix. A close inspection of equations (2.1.5) and (2.4.5) reveals that, for any given r, both sets require the same space and time complexities to solve.

The rates $\lambda_{o,rs}$ and the total external arrival rate λ are related by: $\lambda = \sum_{i \varepsilon R_o(s),s} \lambda_{o,is}$. The parameters $p_{o,rs}$ and $\lambda_{o,rs}$ are related in the following way:

$$p_{o,rs} = \frac{\lambda_{o,rs}}{\sum_{i \varepsilon R_o(s),s} \lambda_{o,is}} , \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{2.4.6}$$

The mean throughput rates for the open chains can then be found using the following set of equations (for a proof of this relationship see [Gel80]):

$$\lambda_{rs}^o = \theta_{rs}^o \sum_{i \varepsilon R_o(s),s} \lambda_{o,is} , \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{2.4.7}$$

The mean value theorem for multichain mixed networks consists of two distinct sets of equations, one for the open chains and a second for the closed chains. In what follows we give a proof of the mean value theorem for the open chains of a mixed networks, based on the normalizing constant. The arrival instant theorem is then used to prove the mean value theorem for the closed chain jobs in a mixed QN.

Theorem 2.4.1:

Consider a mixed multichain separable queueing network consisting of only fixed rate single server service centers, with closed chain population vector $\mathbf{K}_c$ and open chain external arrival rate vector Λ_o. Let $N^o_{rs}(\Lambda_o,\mathbf{K}_c)$ be the mean number of open chain r jobs at service center s of such a network. Then, the following equation holds:

$$N^o_{rs}(\Lambda_o,\mathbf{K}_c)=\frac{\rho^o_{rs}(1+\sum_{j\varepsilon R_c(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c))}{(1-\sum_{i\varepsilon R_o(s)}\rho^o_{is})}\ ,\ r\varepsilon R_o(s)\ ,\ s=1,2,...,M\ . \tag{2.4.8}$$

Proof:

We start with the normalizing constant of mixed networks as given in [Rei75], [Lav80].

$$G(\Lambda_o,\mathbf{K}_c)=\sum_{\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)}\prod_{s=1}^{M}\Big(\sum_{i\varepsilon R_o(s)}\rho^o_{is}\Big)^{k^o_s}\frac{(k^c_s+k^o_s)!}{k^o_s!}\prod_{j\varepsilon R_c(s)}\frac{(\theta^c_{js}s^c_{js})^{k^c_{js}}}{k^c_{js}!}\ . \tag{2.4.9}$$

We prove in Appendix B that the normalizing constant given by (2.4.9) is the same as that given by:

$$G(\Lambda_o,\mathbf{K}_c)=\sum_{\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)}\prod_{s=1}^{M}k^c_s!\prod_{j\varepsilon R_c(s)}\frac{(\theta^c_{js}s^c_{js})^{k^c_{js}}}{k^c_{js}!}\sum_{k^o_s}\frac{(k^c_s+k^o_s)!}{k^c_s!}\prod_{i\varepsilon R_o(s)}\frac{(\rho^o_{is})^{k^o_{is}}}{k^o_{is}!}\ . \tag{2.4.10}$$

The normalizing constant of a mixed network can also be written as the product of $\prod_{s=1}^{M}l^o_s$ and of the normalizing constant of a closed network with population vector $\mathbf{K}_c$, where $\prod_{s=1}^{M}l^o_s$ depends only on the open chains parameters, and the service demands s^c_{rs} of the closed chain jobs are scaled by l^o_s (see Appendix B for a proof). l^o_s is given by:

$$l^o_s=\frac{1}{1-\sum_{i\varepsilon R_o(s)}\rho^o_{is}}\ ,\quad s=1,2,...,M\ . \tag{2.4.11}$$

Therefore, the following also holds:

$$G(\Lambda_o,\mathbf{K}_c)=\left[\prod_{s=1}^{M}\frac{1}{1-\sum_{i\varepsilon R_o(s)}\rho^o_{is}}\right]G_c(\mathbf{K}_c)\ , \tag{2.4.12}$$

where the value of $G_c(\mathbf{K}_c)$ is given by [2]:

$$G_c(\mathbf{K}_c)=\sum_{\mathbf{k}^c\varepsilon S}\prod_{s=1}^{M}k^c_s!\prod_{j\varepsilon R_c(s)}\frac{\left[\theta^c_{js}\dfrac{s^c_{js}}{1-\sum_{i\varepsilon R_o(s)}\rho^o_{is}}\right]^{k^c_{js}}}{k^c_{js}!}\ . \tag{2.4.13}$$

2 We use S as a shorthand notation for $S(\Lambda_o,\mathbf{K}_c,M)$.

We first take the derivative of equation (2.4.12) with respect to the open chain r utilization at service center s. This gives us:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho^o_{rs}} = \left[\frac{G_c(\mathbf{K}_c)}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\sum_{j\varepsilon R_*(s)}N^c_{js}(\mathbf{K}_c)\right]\left(\prod_{s=1}^{M}\frac{1}{(1-\sum_{i\varepsilon R_*(s)}\rho^o_{is})}\right)+$$
$$\left[\frac{1}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\right]^2\left[\prod_{\substack{j=1\\ j\neq s}}^{M}\frac{1}{1-\sum_{i\varepsilon R_*(j)}\rho^o_{ij}}\right]G_c(\mathbf{K}_c)\,, \qquad (2.4.14)$$

where the first term on the right hand side of the above equation is due to the fact that:

$$\frac{\partial G_c(\mathbf{K}_c)}{\partial \rho^o_{rs}} = \sum_{k\varepsilon S}\prod_{s=1}^{M}k^c_s!\sum_{j\varepsilon R_*(s)}\prod_{j\varepsilon R_*(s)}\frac{\left[\theta^c_{js}\frac{s^c_{js}}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\right]^{k^c_{js}-1}}{k^c_{js}!}(k^c_{js})\frac{\theta^c_{js}s^c_{js}}{(1-\sum_{i\varepsilon R_*(s)}\rho^o_{is})^2}, \qquad (2.4.15)$$

which is the same as:

$$\frac{\partial G_c(\mathbf{K}_c)}{\partial \rho^o_{rs}} = \frac{G_c(\mathbf{K}_c)}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\sum_{j\varepsilon R_*(s)}N^c_{js}(\mathbf{K}_c),\ r\varepsilon R_o(s),\ s=1,2,...,M. \qquad (2.4.16)$$

Equation (2.4.14) is then simplified to:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho^o_{rs}} = \left[\frac{G_c(\mathbf{K}_c)}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\sum_{j\varepsilon R_*(s)}N^c_{js}(\mathbf{K}_c)\right]\left[\prod_{s=1}^{M}\frac{1}{(1-\sum_{i\varepsilon R_*(s)}\rho^o_{is})}\right]+\frac{G(\Lambda_o,\mathbf{K}_c)}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}},$$
$$r\varepsilon R_o(s),\ \ s=1,2,...,M,$$

which in turn is reduced to:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho^o_{rs}} = \frac{G(\Lambda_o,\mathbf{K}_c)}{1-\sum_{i\varepsilon R_*(s)}\rho^o_{is}}\left\{1+\sum_{j\varepsilon R_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c)\right\},\ \ r\varepsilon R_o(s),\ \ s=1,2,...,M. \qquad (2.4.17)$$

On the other hand, the derivative of equation (2.4.10) with respect to the chain r utilization of service center s produces the following:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho^o_{rs}} = \frac{G(\Lambda_o,\mathbf{K}_c)}{\rho^o_{rs}}N^o_{rs}(\Lambda_o,\mathbf{K}_c),\quad r\varepsilon R_o(s),\quad s=1,2,...,M. \qquad (2.4.18)$$

By equating (2.4.17) to (2.4.18) we obtain:

$$N^o_{rs}(\Lambda_o,\mathbf{K}_c) = \frac{\rho^o_{rs}(1+\sum_{j\varepsilon R_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c))}{(1-\sum_{i\varepsilon R_*(s)}\rho^o_{is})},\quad r\varepsilon R_o(s),\quad s=1,2,...,M. \qquad (2.4.19)$$

□

We next prove the mean value theorem for the closed chains of mixed networks.

Theorem 2.4.2:

Consider a mixed multichain separable queueing network containing fixed rate single servers. The following relationship holds for the mean number of closed chain r jobs at service center s:

$$N_{rs}^{c}(\Lambda_o,\mathbf{K}_c)=\frac{\lambda_{rs}^{c}s_{rs}^{c}(1+\sum_{j\varepsilon R_o(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r))}{(1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o})},\quad r\varepsilon R_c(s),\ s=1,2,...,M. \tag{2.4.20}$$

Proof:

We can prove this relationship either by using the normalizing constant or by applying the steady state arrival instant distribution theorem [Sev81, Zah81]. A proof based on the arrival instant theorem and given in [Zah81] is presented here. The arrival instant theorem states that a closed chain r customer arriving at server s in steady state sees the service center in equilibrium with itself removed. Hence, we can write:

$$\begin{aligned}
N_{rs}^{c}(\Lambda_o,\mathbf{K}_c)&=\lambda_{rs}^{c}s_{rs}^{c}\left[1+\sum_{i\varepsilon R_o(s)}N_{is}^{o}(\Lambda_o,\mathbf{K}_c-e_r)+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r)\right]\\
&=\lambda_{rs}^{c}s_{rs}^{c}\left[1+\frac{\sum_{i\varepsilon R_o(s)}\rho_{is}^{o}(1+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r))}{(1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o})}+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r)\right]\\
&=\lambda_{rs}^{c}s_{rs}^{c}\left[1+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r)\right](1+\frac{\sum_{i\varepsilon R_o(s)}\rho_{is}^{o}}{1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o}})\\
&=\frac{\lambda_{rs}^{c}s_{rs}^{c}(1+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\Lambda_o,\mathbf{K}_c-e_r))}{(1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o})},
\end{aligned}$$

where we substituted $N_{is}^{o}(\Lambda_o,\mathbf{K}_c-e_r)$ with its expression from equation (2.4.8).

□

It is easy to show that, for IS service centers, equation (2.4.8) becomes:

$$N_{rs}^{o}(\mathbf{K})=\lambda_{rs}^{o}s_{rs}^{o},\quad r\varepsilon R_o(s),\quad s=1,2,...,M. \tag{2.4.21}$$

On the other hand, equation (2.4.20) for closed chain jobs at infinite servers reduces to:

$$N_{rs}^{c}(\mathbf{K}_c)=\lambda_{rs}^{c}s_{rs}^{c},\quad r\varepsilon R_c(s),\quad s=1,2,...,M. \tag{2.4.22}$$

The mean value of the customer waiting times for different chains can be computed by applying Little's theorem to equations (2.4.8) and (2.4.20). The result for open chain customers is:

$$W_{rs}^{o}(\mathbf{K})=\frac{s_{rs}^{o}(1+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\mathbf{K}))}{(1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o})},\quad r\varepsilon R_o(s),\quad s=1,2,...,M, \tag{2.4.23}$$

and that for closed chain jobs:

$$W_{rs}^{c}(\mathbf{K})=\left[\frac{s_{rs}^{c}}{1-\sum_{i\varepsilon R_o(s)}\rho_{is}^{o}}\right](1+\sum_{j\varepsilon R_c(s)}N_{js}^{c}(\mathbf{K}-e_r)),\quad r\varepsilon R_c(s),\quad s=1,2,...,M. \tag{2.4.24}$$

The following relationship holds for the mean number of jobs of all classes at service center s.

$$N_s(\mathbf{K}) = \sum_{r \in R_c(s)} \frac{\lambda_{rs}^c s_{rs}^c (1 + \sum_{j \in R_c(s)} N_{js}^c(\mathbf{K} - e_r))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o)} + \sum_{r \in R_o(s)} \frac{\rho_{rs}^o (1 + \sum_{j \in R_c(s)} N_{js}^c(\mathbf{K}))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o)},$$

$$s = 1,2,...,M. \qquad (2.4.25)$$

Equation (2.4.26) can be further simplified to:

$$N_s(\mathbf{K}) = \sum_{r \in R_c(s)} \frac{\lambda_{rs}^c s_{rs}^c (1 + \sum_{j \in R_c(s)} N_{js}^c(\mathbf{K} - e_r))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o)} + \frac{\rho_s^o}{1 - \rho_s^o}(1 + N_s^c(\mathbf{K})),$$

$$s=1,2,...,M, \qquad (2.4.26)$$

where $\rho_s^o = \sum_{i \in R_o(s)} \rho_{is}^o$ is the utilization of service center s due to the open chain jobs.

2.4.1 Properties of Mixed Multichain Networks

An inspection of equations (2.4.8) and (2.4.20) reveals several interesting properties of mixed multichain networks. The mean value equation for open chains, unlike that for closed chains, does not have any recursive property with respect to the open chains themselves. More interestingly, equation (2.4.20) reveals that the performance measures of the closed chain jobs in a mixed network are exactly the same as those in a closed network consisting of only the closed chain jobs where all the service demands at all the centers are scaled down by the factor $\frac{1}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o)}$. This term is the only contribution of the open chain jobs to the performance metrics of the closed chain jobs. In a forthcoming paper, we will describe the use of mixed networks and their properties to characterize workloads in distributed systems. We will apply those results to the theoretical study of load sharing in distributed systems and validate the conclusions through experimentation.

Stability in a mixed network is defined in the same way as for an open network. A network is said to be stable if its serving capacity can handle the arriving traffic and all the queue lengths in the network remain finite. This is the same as requiring that the balance equations have a nonzero solution. If we let $\rho_s^o = \sum_{i \in R_o(s)} \rho_{is}^o$, then ρ_s^o is the utilization of service center s due to the open chain jobs. Hence, a mixed network remains stable as long as $\rho_s^o < 1$ for all s.

2.5 The Computational Complexities of the MVA Solution of Mixed Multichain Networks

We can determine the computational complexities of the solution of a mixed multichain network by deriving the complexity of the open and closed chain equations and combining the results. The complexity of the closed chain solution can easily be studied by observing that the effect of the open chain jobs on the closed chain performance metrics is limited to a scaling factor. Suppose we have a load independent QN with M fixed rate service centers. Let us assume that there are R^c closed chains, R^o open chains, with $R = R^c + R^o$. We use the following notation:

$T_o(R^o, R^c, M)$, $T_c(R^o, R^c, M)$ = time complexity of the open (o) or closed (c) chain MVA solution of the QN.

$S_o(R^o,R^c,M)$, $S_c(R^o,R^c,M)$ = space complexity of the open (o) or closed (c) chain MVA solution of the QN.

$T(R^o,R^c,M)$ = total time complexity of the MVA solution of the network.

$S(R^o,R^c,M)$ = total space complexity of the MVA solution of the network.

Observe that the following relationships hold:

$T(R^o,R^c,M)=T_o(R^o,R^c,M)+T_c(R^o,R^c,M)$,

$S(R^o,R^c,M)=S_o(R^o,R^c,M)+S_c(R^o,R^c,M)$.

The time/space complexities for evaluating the visit ratios of the open chains can be derived by inspecting equation (2.4.5). To evaluate the open chain time complexity for computing the visit ratios, observe that we need M subtractions to compute $I_M - P_r$. $2M^3-2M^2+M$ operations are needed to evaluate the inverse of $\left[I_M - P_r\right]$. Additional M^2 multiplications and M^2 additions are needed to compute $P_{o,r}\left[I_M - P_r\right]^{-1}$. Therefore, the total time complexity to evaluate the visit ratios for each class is $V=2M^3+2M$. The total time complexity for all classes is $V^o=2R^oM^3+2R^oM$. We have already reported the space/time complexity involved in evaluating the visit ratios for the closed chain jobs in Table 1 and 2. The time complexity of the evaluation of visit ratios for the closed chain jobs is smaller, and $V^o-V^c=2RM$ (for networks of the same size, i.e., $R=R^c=R^o$).

2.5.1 Open Chains Space/Time Complexity

An inspection of the mean value equation (2.4.8) for the open chains of a mixed network reveals that we need R^cM additions to calculate $N_s^c(\Lambda_o,\mathbf{K}_c)=\sum_{j\in R_{s(c)}} N_{js}^c(\Lambda_o,\mathbf{K}_c)$. Another $5R^oM$ operations are needed to calculate $N_{rs}^o(\Lambda_o,\mathbf{K}_c)$ for all the open chains at all the servers. This total corresponds to the sum of R^oM additions, R^oM multiplications in the numerator, $2R^oM$ addition in the denominator, and R^oM divisions. The computation of the mean throughput rates and the mean waiting times of all open chains at all the servers requires $2R^oM$ additional operations. We further need R^oM multiplications to evaluate the utilization values for all the classes. The open chain space complexity is $S_o(R^o,R^c,M)=5R^oM$, one R^oM for each of the following indices: utilization, mean throughput rate, mean queue length, and mean waiting time, plus R^oM units of space to store the visit ratios for all the classes.

It is important to notice that in all of our discussions we have assumed a non class-hopping network. If the original queueing network is a class-hopping one, we have to transform it to a non class-hopping network. This transformation allows us to compute the visit ratios of the QN for all classes independently and simultaneously. Obviously, this transformation and concurrent solution are achieved at the cost of higher space/time requirements. We should point out here that, when dealing with non class-hopping queueing networks, to solve for the visit ratios we need an amount of storage of at least R^oM^2 if the visit ratios of all classes are to be determined simultaneously, or one of at least M^2 if the visit ratios are computed sequentially.

The reason we do not include the storage space required for the transition matrix values, as we discussed in section 2.3, is due to the fact that this space is needed for the first stage of the solution procedure. Since in most cases the total storage needed for the solution of the QN in the second step far exceeds that of the first step, we do not include the storage needed to solve for the visit ratios. We point out, however, that for open networks or certain single class closed networks the storage space needed to evaluate the visit ratios might be larger than that needed to solve the QN in the second stage. We shall discuss these cases in section 2.7.

Therefore, the open chain time and space complexities are given by:

$$T_o(R^o,R^c,M)=8R^oM+R^cM+V^o, \tag{2.5.1}$$

$$S_o(R^o,R^c,M) = 5R^oM. \tag{2.5.2}$$

2.5.2 Closed Chains Space/Time Complexity

The complexity of the closed chain equations in a mixed network is the same as that in a closed network discussed in section 2.3. The only additional term involved is the scaling factor due to the presence of open chains. Taking into account the scaling factor, the time and space complexities for the closed chains become:

$$T_c(R^o,R^c,M) = R^cM(4X+3) + (R^o + 2R^c)M + V^c, \tag{2.5.3}$$

$$S_c(R^o,R^c,M) = (R^c + M + R^cM)Y + 4R^cM. \tag{2.5.4}$$

Table 3 shows the total space and time complexity for the MVA solution of mixed networks. I is the closed chain visiting the largest number of service centers. Similar results for the space and time complexities of a mixed QN's solution using the convolution algorithm can be obtained but will not be reported here.

Table 3 Space/Time Complexity (Load Independent Servers)

Complexity	Mean Value Analysis
$S(R^o,R^c,M)$	$(R^c+M+R^cM)Y + 5R^oM + 4R^cM$
$T(R^o,R^c,M)$	$R^cM(4X+3) + (9R^o + 3R^c)M + V^c + V^o$

$$X = \prod_{r=1}^{R^*}(K_r+1) \qquad Y = \prod_{\substack{i=1 \\ i\neq l}}^{R^*}(K_i+1) \qquad V^o = 2R^oM^3 + 2R^oM \qquad V^c = 2R^cM^3$$

2.6. The Exact Solution of Multichain Open Networks

In this section we discuss the solution of multichain product form open networks. The results presented here can be derived from those obtained for mixed networks in section 2.4. We shall therefore provide only the results. The proofs can be easily deduced from those given for multichain mixed networks.

Consider a multichain product form network containing fixed rate single server and infinite server service centers, and not containing any closed subchains. The total external arrival rate, λ, is assumed to be Poisson and independent of the network state. The balance equations are the same as those given in (2.4.2) and (2.4.3).

The equilibrium state probability distribution of $(k_1,k_2,\ldots,k_M)$, now factorizes completely into a product of steady state distributions for individual service centers:

$$p(k_1,k_2,\ldots,k_M) = \prod_{s=1}^{M} p_s(k_s), \tag{2.6.1}$$

where:

$$p_s(k_s) = (1-\sum_{i\in R(s)}\rho_{is}^o)(\sum_{i\in R(s)}\rho_{is}^o)^{k_s} \quad \text{if } \textit{the server is of type } 1, 2, \textit{or } 3, \tag{2.6.2}$$

$$p_s(k_s) = \frac{e^{-\sum_{i\in R(s)}\rho_{is}^o}(\sum_{i\in R(s)}\rho_{is}^o)^{k_s}}{k_s!} \quad \text{if } \textit{the server is of type } 4. \tag{2.6.3}$$

It should be pointed out here that equations (2.6.2) and (2.6.3) hold only if the network is stable. This means that we should have $\rho_s^o<1$ for all servers of type 1,2, or 3.

The mean queue length of chain r jobs at server s of type 1,2, or 3 is then given by:

$$N_{rs}^o(\Lambda_o) = \frac{\rho_{rs}^o}{1-\sum_{i\varepsilon R(s)}\rho_{is}^o}, \quad r\varepsilon R(s), \quad s=1,2,...,M \ . \tag{2.6.4}$$

The mean waiting time of chain r jobs at server s is given by the following equation:

$$W_{rs}^o(\Lambda_o) = \frac{s_{rs}^o}{1-\sum_{i\varepsilon R(s)}\rho_{is}^o}, \quad r\varepsilon R(s), \quad s=1,2,..,M \ . \tag{2.6.5}$$

The generating function of the stationary queue length distribution for each server of type 1,2, or 3 is derived directly from the distribution given by equation (2.6.2):

$$\phi(t) = \frac{1-\sum_{i\varepsilon R(s)}\rho_{is}^o}{1-t\sum_{i\varepsilon R(s)}\rho_{is}^o} \ . \tag{2.6.6}$$

The mean and the higher moments of the steady state queue lengths at server s can be obtained from equations (2.6.2) and (2.6.3), or from the moment generating function given by equation (2.6.6). The mean queue length of all job classes at any service center of type 1,2, or 3 is given by:

$$N_s^o = \sum_{r\varepsilon R(s)} \frac{\rho_{rs}^o}{1-\sum_{i\varepsilon R(s)}\rho_{is}^o} = \frac{\rho_s^o}{1-\rho_s^o}, \quad s=1,2,...,M \ . \tag{2.6.7}$$

2.7 The Computational Complexities of Open Network Solutions

Since in mixed networks the open and closed chains interact, the complexities of solving open networks are best determined by direct examination of the relationships derived for open networks. The computational complexity of the MVA solution of an open network can be determined by inspecting equations (2.4.5), (2.6.4), and (2.6.5). The storage space needed for the determination of visit ratios is the same as that derived in section 2.5.1. We need an amount of storage space equal to RM^2 to determine the visit ratios of all classes simultaneously and an amount equal to M^2 if the visit ratios are evaluated sequentially. However, the second phase of the solution of open networks does not have the large exponential storage space requirements of the same phase for the closed chains. Thus, the space complexity, $S_o(R^o,M)$, is given by:

$$S_o(R^o,M) = \max(5R^oM, R^oM^2) \ \ or \ \ \max(5R^oM, M^2) \ , \tag{2.7.1}$$

and the time complexity by:

$$T_o(R^o,M) = 8R^oM + V^o \ . \tag{2.7.2}$$

To compare the space and time complexities of the solutions of closed, mixed, and open networks, several cases are presented in Table 4. The first column shows a vector with elements representing the total number of chains, the total number of open chains, the population of each chain (assumed to be the same for all classes), and the number of service centers in the network. The drastic reduction in space and time complexities as the number of open chains increases (while the total number of chains is kept fixed) is evident.

Table 4. Space/Time Complexities for Closed, Mixed, and Open Networks

(R, R^o, K, M)	Space Complexity (Mbytes)	Time Complexity (Mflops)
(10,0,10,10)	2.83×10^{5}	1.03×10^{7}
(10,2,10,10)	1.91×10^{3}	6.85×10^{4}
(10,5,10,10)	0.952	32.2
(10,10,0,10)	5.0×10^{-4}	2.10×10^{-2}

2.8 Conclusions

Exact solution techniques for closed product form queueing networks are very efficient. However, the computational cost of solving closed networks is extremely high and in most cases practically impossible. In mixed multichain networks the dominant factor in the computational complexity of solving them is still contributed by the closed chains.

Chapter 3

An Overview of Approximate Solution Techniques for Multichain Product Form Queueing Networks

3.1 Introduction

We observed in chapter 2 that efficient exact solution techniques for product form queueing networks are available. However, their computational costs become prohibitively expensive, and in many cases practically impossible, for sufficiently large networks. Typical modeling, performance evaluation, and configuration design of present day systems necessitate the solution of large multichain QN models. This need has created an intense interest among scholars in recent years to search for accurate, cost effective, and efficient approximate algorithms and methodologies for the solution of large multiclass QNs. This fervent attention stems from the need to analyze more complex systems, and can be deduced from the wealth of the literature on the subject. In this chapter we present an overview of algorithms for the approximate solution of multichain product form queueing networks.

Most of the techniques proposed for the approximate solution of product form queueing networks fall into one of the following categories:

- MVA-based approximation techniques
- Bounding algorithms
- Asymptotic algorithms
- Other approximate algorithms

3.2 MVA-Based Approximation Algorithms

Most of the attempts in the MVA-based approximate algorithms have been made to accomplish some or all of the following objectives:

(1) To reduce the computational cost drastically.

(2) To achieve a high level of accuracy.

(3) To develop approximate algorithms for which existence, uniqueness, and fast convergence of the solutions can be established.

The first objective has been the driving point for most of the algorithms. This goal is usually realized by finding ways to eliminate the recursive structure of the MVA equations, thereby drastically reducing the computational cost of the solution. The second objective is usually reached through heuristics. There are a few well known MVA-based approximations:

- The Bard-Schweitzer approximation
- The Linearizer approximation
- Other MVA-based algorithms

The characteristics of these methods vary greatly. Below we review the main properties of these algorithms in detail.

3.2.1 The Bard-Schweitzer Approximation

Schweitzer [Sch79] proposed an MVA-based approximate algorithm that applies to product form networks with fixed rate single server and IS service centers. Reports on real-world applications of this algorithm can be found in [Bar80] and [Cha82]. The algorithm is usually referred to as the Bard-Schweitzer algorithm. The objective is to estimate the queue lengths $N_{rs}(\mathbf{K}-e_j)$ from the values of $N_{rs}(\mathbf{K})$, thereby avoiding the recursion for obtaining the values of $N_{rs}(\mathbf{K}-e_j)$ in the MVA equation. This objective is achieved by the following approximation:

$$N_{rs}(\mathbf{K}-e_j)=\begin{cases} N_{rs}(\mathbf{K}) & j\neq r \\ \dfrac{K_r-1}{K_r}N_{rs}(\mathbf{K}) & j=r \end{cases} \tag{3.2.1}$$

The resulting iterative algorithm is the following:

Initially set $N_{rs}^{(0)}(\mathbf{K})=\dfrac{K_r}{M}$, $r=1,2,...,R$, $s=1,2,...,M$.

Do

$$N_{rs}^{(i)}(\mathbf{K}-e_j)=\begin{cases} N_{rs}^{(i-1)}(\mathbf{K}) & j\neq r \\ \dfrac{K_r-1}{K_r}N_{rs}^{(i-1)}(\mathbf{K}) & j=r \end{cases}$$

$$W_{rs}^{(i)}(\mathbf{K})=s_{rs}\left(1+\sum_{j=1}^{R}N_{js}^{(i)}(\mathbf{K}-e_r)\right)$$

$$\lambda_{rs}^{(i)}(\mathbf{K})=\frac{K_r}{\sum_{j=1}^{M}\dfrac{W_{rj}^{(i)}(\mathbf{K})\theta_{rj}}{\theta_{rs}}}$$

$$N_{rs}^{(i)}=\lambda_{rs}^{(i)}(\mathbf{K})W_{rs}^{(i)}(\mathbf{K})$$

Until $|N_{rs}^{(i)}(\mathbf{K})-N_{rs}^{(i-1)}(\mathbf{K})|\leq\varepsilon$.

Bard-Schweitzer's approximation is computationally cheap. The space complexity for the Bard-Schweitzer algorithm is $O(RM)$ for each iteration (under the assumption that the visit ratio values are known). The time complexity depends on the stopping rule bound and on the size and other characteristics of the queueing network. We should point out a major oversimplification made in the literature concerning the time complexity of the Bard-Schweitzer algorithm. In many publications the time complexity is reported to be $O(RM)$. However, the actual time complexity is dependent on the parameters stated above, and indeed networks can be constructed for which the time complexity far exceeds this value. If the algorithm is implemented iteratively as it was described here, then very slowly converging networks can be found whose solution time complexity is larger than $O(RM)$. An approach that removes this problem is to transform the iterative algorithm to a set of nonlinear equations. It is shown in [de-83] that the set of nonlinear equations resulting from the Bard-Schweitzer algorithm always has a solution in the feasible region; however, the uniqueness of the solution is by no means guaranteed. Large errors have been observed in the solution of certain stress networks by [Cha82].

3.2.2 The Linearizer Approximation

Chandy and Neuse have proposed a heuristic [Cha82] called Linearizer, which can be considered as a generalization of the Bard-Schweitzer algorithm. The Linearizer algorithm can solve product form networks consisting of only fixed rate single server and IS service centers. This approximation yields more accurate results than the Bard-Schweitzer algorithm at higher computational costs. The objective is to estimate iteratively the mean queue length with population vectors $\mathbf{K}$, $\mathbf{K}-e_j$ and $\mathbf{K}-e_j-e_l$ in order to eliminate the recursion. The estimator is defined by:

$$N_{rs}(\mathbf{K}-e_j) = (\mathbf{K}-e_j)_r\left(\frac{N_{rs}(\mathbf{K})}{K_r} + D_{rjs}(\mathbf{K})\right), \tag{3.2.2}$$

where D_{rjs} and $(\mathbf{K}-e_j)_r$ are given by:

$$D_{rjs} = \frac{1}{K_r}\left[N_{rs}(\mathbf{K}-e_j) - N_{rs}(\mathbf{K})\right], \tag{3.2.3}$$

$$(\mathbf{K}-e_j)_r = \begin{cases} K_r - 1 & j = r \\ K_r & j \neq r \end{cases} \tag{3.2.4}$$

The Linearizer algorithm assumes that D_{rjs} is a constant, and then tries to estimate its value using an iterative algorithm similar to the Bard-Schweitzer approximation. Observe that, when $D_{rjs} = 0$, Linearizer reduces to the Bard-Schweitzer algorithm. Using the Linearizer algorithm, the exact MVA is reduced to the equivalent of a set of $I + R(I-1)$ simultaneous nonlinear equations, where I is the number of iterations. Since the computational cost of solving a set of R nonlinear equations is (assuming reasonable error bounds for stopping rules) $O(MR^2)$, the computational cost of the Linearizer can become large if there are a large number of classes. The space complexity of Linearizer is $O(MR^2)$ compared with $O(RM)$ for the Bard-Schweitzer algorithm.

One major drawback for the Linearizer algorithm is the absence of any known uniqueness, existence, or convergence proof. Neuse and Chandy [Neu81] extended Linearizer by proposing an algorithm called SCAT to solve load dependent networks. SCAT, however, cannot solve mixed networks and there is no theoretical support for the extension. There is yet another approximation proposed by Zahorjan et al. [Zah86] which is simpler in form than Linearizer, and for which, there is no existence, uniqueness, or convergence proof either. The authors report that the algorithm's accuracy is similar to that provided by Linearizer. This algorithm has time and space complexities of $O(MR^2)$ and $O(MR)$, respectively. However, as we said earlier for Linearizer, the time complexity is a rough estimate. Krzesinski and Greyling [Krz84] have found load dependent networks that when solved by Linearizer, produce large errors; they have proposed an algorithm (with additional heuristics) for solving those load dependent networks.

3.2.3 Other MVA-Based Approximations

As we pointed out, a major drawback of Linearizer and of its improved versions as well is the lack of existence, convergence, and uniqueness proofs. To eliminate these shortcomings, Chow [Cho83] defined a differential error term denoted by:

$$\delta_{rs} = \frac{N_s(\mathbf{K}-e_r) - N_s(\mathbf{K})}{N_s(\mathbf{K})}, \tag{3.2.5}$$

and then proceeded to replace $N_s(\mathbf{K}-e_r)$ in the exact MVA equations by its equivalence from (3.2.5) to produce a set of R nonlinear equations in terms of the unknown throughput rates for each class at some service center. To solve this set of nonlinear equations, the error terms δ_{rs} have to be estimated. As a first approximation, he assumed $\delta_{rs}=0$ (i.e., $N_s(\mathbf{K}-e_r) = N_s(\mathbf{K})$), and proved that the resulting set of nonlinear equations always has a unique solution in the feasible region. The first approximation is very similar and closely related to the Bard-Schweitzer algorithm. This point becomes clear by assuming that the first relationship in Bard-Schweitzer approximation in (3.2.1) holds for all classes and then summing both sides over all classes:

$$\sum_{r=1}^{R} N_{rs}(\mathbf{K}-e_j) = N_s(\mathbf{K}-e_j) \quad j=1,2,...,R \ , \quad s=1,2,...,M \ ,$$

$$\sum_{r=1}^{R} N_{rs}(\mathbf{K}) = N_s(\mathbf{K}) \ , \quad s=1,2,...,M \ , \tag{3.2.6}$$

To improve the approximation, Chow tried to estimate the δ_{rs} terms by the first approximate algorithm. This was done by solving the network approximately at population vectors $\mathbf{K}$ and $\mathbf{K}-e_i$, using $\delta_{ij}=0$. He then estimated δ_{ij} by the following approximation:

$$\delta^*_{rs} = \frac{N^*_s(\mathbf{K}-e_r) - N^*_s(\mathbf{K})}{N^*_s(\mathbf{K})}, \tag{3.2.7}$$

where * superscripts denote the estimated values of the corresponding parameters.

This step produced noticeable improvement only if the estimates obtained using the first approximation were accurate. The error results for both cases were reported to be comparable to those of Linearizer. One major advantage of the first approximation ($\delta_{rs}=0$) is that the existence and uniqueness of the solution in the feasible region is always guaranteed. For the second approximation, only the existence of a solution in the feasible region can be proven. Both algorithms apply to QNs with fixed rate single and IS service centers. We will extend and improve this algorithm in chapter 4.

At least two other MVA-based approximations have been proposed. E de Souza et al. in [Sil84] transformed the Bard-Schweitzer approximation defined by (3.2.1) into a set of nonlinear equations in a similar way as that by Chow, and for which they proved the existence of a solution in the feasible region. Another approach can be found in [Lav83].

3.3 Bounding Algorithms

There are two different approaches in this category:

- Simple single value bounds
- Performance bound hierarchies (PBH)

Simple and single valued upper and lower bounds for performance measures were initially derived for single-class closed queueing networks. These bounds were later extended to include multiple-class networks. In many practical cases, particularly in multichain networks, these bounds are

often so far away from the exact values that their usefulness diminish. The evaluation of the single valued bounds in all cases is computationally cheap and can be carried out using the parameters of the QN and the values of the visit ratios. Thus, computing single value bounds does not demand the solution of the queueing network itself. There are several different categories of single valued bounds.

(1) Asymptotic Bound Analysis Bounds (ABA bounds) [Mun74].

(2) Balanced Job Bounds (BJB) [Zah82].

(3) Composite Bound Method (CBM) [Ker86], [Ker84].

All these single value bounds are on throughput rates. We shall first define the notation used, and then introduce these bounds.

Let:

$L_{rs} = \theta_{rs} s_{rs}$ = the *loading factor* for service center s and class r jobs.

The *maximum loading*, L_{br}, is given by:

$$L_{br} = \max_{j \in S(r)} L_{rj}, \quad r=1,2,...,R; \tag{3.3.1}$$

the *average loading* for chain r jobs, L_{ar}, is defined by:

$$L_{ar} = \frac{\sum_{j \in S(r)} L_{rj}}{M}, \quad r=1,2,...,R; \tag{3..3.2}$$

the *minimum loading* for class r jobs, L_{mr}, is denoted by:

$$L_{mr} = \min_{j \in S(r)} L_{rj}, \quad r=1,2,...,R; \tag{3.3.3}$$

and the minimum response time for class r jobs at some designated center l, R_{rl}, is given by:

$$R_{rl} = \sum_{j \in S(r)} L_{rj}, \quad r=1,2,...,R. \tag{3.3.4}$$

The system throughput rate for class r jobs (at some designated service center l), λ_{rl}, and the throughput rate of class r jobs at service center s, λ_{rs}, are related by $\theta_{rs} = \frac{\lambda_{rs}}{\lambda_{rl}}$ (we assume here $\theta_{rl}=1$, i.e., that the θ_{rs} are normalized). The corresponding values of these parameters for the single-class queueing networks are denoted by θ_s, s_s, L_b, L_a, L_m, R_l, $\lambda_l(K)$, $\lambda_s(K)$. The upper and lower bounds for each class of performance measure bounds are denoted by + and - superscripts respectively.

3.3.1 The ABA Bounds

The ABA bounds are linear asymptotic upper bounds (for single-class and multiple-class networks) of the form:

$$\lambda_l^+(K) = \min\left[K/R_l, 1/L_b\right],$$

$$\lambda_{rl}^+(\mathbf{K}) = \min\left[K_r/R_{rl}, 1/L_{br}\right], \quad r=1,2,...,R. \tag{3.3.5}$$

Clearly the ABA upper bounds for multiple-class QNs are loose bound, since they do not take into consideration the effect of other chains.

3.3.2 The BJB Bounds

The BJB bounds for multiple-class QNs are given by [Zah82]:

$$\lambda_{rl}^+(\mathbf{K}) = \frac{K_r}{(K+M-1)L_{br}}, \quad r=1,2,...,R,$$

$$\lambda_{rl}^-(\mathbf{K}) = \frac{K_r}{(K+M-1)L_{mr}}, \quad r=1,2,...,R. \tag{3.3.6}$$

A tighter set of bounds for single-class networks, also proposed by Zahorjan et al., is of the following form [Zah82].

$$\frac{K}{R_l + (K-1)L_b} \le \lambda_l(K) \le \frac{K}{R_l + (k-1)L_a}, \quad r=1,2,...,R. \tag{3.3.7}$$

The lower bound in equation (3.3.7) has a multiclass counterpart given by:

$$\lambda_{rl}(\mathbf{K}) = \frac{K_r}{R_{rl} + (K-1)L_{br}} \le \lambda_{rl}(\mathbf{K}), \quad r=1,2,...,R. \tag{3.3.8}$$

3.3.3 The CBM Upper Bounds

The CBM bound was introduced by Kerola [Ker86], [Ker84], and is given by:

$$\lambda_{rl}(\mathbf{K}) \le \lambda_{rl}^{+}(\mathbf{K}) = \min_j \left[1 - \sum_{i \ne r} \lambda_{il}^{B^-}(\mathbf{K}) L_{ij} \right] / L_{rj}, \quad r=1,2,...,R, \tag{3.3.9}$$

where $\lambda_{il}^{B^-}$ denotes the lower bound given by (3.3.8). The utilization of service center s due to class r jobs can be determined by $\rho_{rs} = \lambda_{rs} s_{rs}$.

The CBM upper bounds give better estimates when the QN is saturated and has at least one service center (the bottleneck) that is fully utilized. However, if the QN is lightly loaded, good estimates may not be found. For a more in depth analysis and comparisons of CBM bounds with the ABA and the BJB bounds see [Ker85]. All the multiple-class bounds given above have a common limit given by $1/L_{br}$ as $K_r \to \infty$.

We emphasize that these bounds are simple, not accurate, and can readily be found without solving the QNs involved at a minimal computational cost. The only parameters needed for their evaluation are the values of the visit ratios and of the network's parameters. As a result, these bounds could produce gross errors for the throughput rates. Several other bounds for single-class QNs have been proposed. Noteworthy among them are the generalized quick bounds studied by Suri [Sur83].

3.3.4 Performance Bound Hierarchies

A different, hierarchical bounding technique which is based on MVA is the Performance Bound Hierarchies (PBH) algorithm developed in [Eag83] for single-class networks, and in [Eag86] for multiple-class networks. Assuming that $\bar{\lambda}_s^i$ and $\underline{\lambda}_s^i$ denote the upper and lower level i bounds for the mean throughput rates of a single-class network with population K, the PBH algorithm produces a hierarchy of upper and lower bounds for the throughput rates in the following sense:

$$\underline{\lambda}^{i-1}(K) \le \underline{\lambda}^{i}(K) \le \lambda(K) \le \bar{\lambda}^{i}(K) \le \bar{\lambda}^{i-1}(K), \tag{3.3.10}$$

where $\lambda(K)$ is the exact mean throughput rate at service center s. The level i bounds are generated by evaluating the upper and lower bounds on performance measures of a network with population of $K-i$. For the initial conditions at population $K-i$, any of the simple bounds discussed in sections 3.3.1-3.3.3 would suffice. Using these bounds, similar bounds for other performance measures can subsequently be found. The exact solutions are obtained by the level K bounds. The PBH algorithm has been extended to multiple-class networks, and to a special class of load dependent multiple-class QNs [Eag86]. The PBH bounds tend to loosen and become slack as the congestion in the network increases. The time complexity of the PBH algorithm for multichain QNs is given by:

$$6M\,|R|\left[\,|R_i|+i\right], \tag{3.3.11}$$

and the space complexity is given by:

$$M\,|R|\left[\,|R_i|+i\right]. \tag{3.3.12}$$

The computational complexity of the PBH algorithm increases combinatorically as a function of the number of classes, and becomes prohibitive for QNs with many classes of jobs. The class of networks solved using the PBH algorithm is more general than that dealt with by most other approximations. However, as in the case of exact solution techniques, it is expensive and impractical for large QN models. The PBH algorithm, as reported in [Eag86], can be implemented easily. One important observation to be made here is that the bounding algorithms (both the simple and the PBH bounds) for single-class networks are a theoretical curiosity only, since in most practical cases the computational cost of solving single-class networks exactly is not high.

3.4 Asymptotic Algorithms

Asymptotic algorithms provide asymptotic values of performance measures for the queueing network being considered. In these techniques the approximate values of the performance measures, as some parameters of the network becomes large, approach the exact values. A recently proposed asymptotic algorithm, based on a multiple integral representation of the normalizing constant $G(\mathbf{K})$, is the Asymptotic Expansion Algorithm [McK81], [McK82], [McK84], [Ram82]. The Asymptotic Expansion method replaces the factorial terms in equation (2.1.9) in chapter 2 by their Euler's integral representation using:

$$k! = \int_0^\infty e^{-t} t^n dt \tag{3.3.13}$$

The multinomial theorem is then applied to simplify the summation over all possible states with the following identity:

$$(\sum k_{ij})! = \int_0^\infty e^{t_j} \prod t_j^{k_{ij}} dt_j \qquad j=1,2,...,M. \tag{3.3.14}$$

The multiple integral given in (3.3.14) is then expressed as a function of a "large parameter" N, i.e., $G(\mathbf{K}) = I(N)$. The asymptotic expansion of $I(N)$ is then derived and expressed in the following form:

$$I(N,m) \simeq \sum_{k=1}^{m} \frac{A_k}{N^k}, \tag{3.3.15}$$

where:

$$\lim_{m \to \infty} I(N,m) = I(N), \tag{3.3.16}$$

The coefficients A_k are related to the normalizing constant of certain product form networks, called pseudo networks, with much smaller populations. The bounds on the difference between $I(N)$ and its m-term estimates $I(N,m)$ are then evaluated. As the value of m increases, these bounds become tighter, and in the limit converge to the exact values. Obviously the larger the value of m, the more expensive the algorithm becomes. It is reported in [Ram82] that, to obtain satisfactory estimates, no more than four terms are needed to be evaluated in the expansion. The Asymptotic Expansion Algorithm applies to multichain product form queueing networks with single server fixed rate and IS service centers which contain at least one IS service center visited by all the chains. Additionally, for the algorithm to produce accurate estimates of the performance measures, all the service centers in the queueing network model should operate at "normal usage", i.e., should have a utilization not larger than .85 [Ram82]. The Asymptotic Expansion Algorithm yields upper and lower error bounds for the performance measures. The complexity of the underlying algorithm for the asymptotic expansion suggests that the implementation of the algorithm is not trivial. In fact version 1.1 of Panacea, the implementation of the algorithm by the developers, is about 6000 lines of C-code long [Ram82]

Another approach that investigates the asymptotic properties of the same kind of queueing network considered by the Asymptotic Expansion Algorithm has been proposed by Lavenberg [Lav80]. Assume that the throughput rates of class r jobs at the IS service center M visited by all the chains are

denoted by λ_{rM} , $r=1,2,...,R$. Furthermore, suppose that $\lim_{K\to\infty} K_r \mu_{rM}$ exists and is finite. Here, K_r is the population of chain r jobs and $\sum_{i=1}^{R} K_i = K$. Lavenberg showed that, as the population of the network increases, the equilibrium state probability distribution of the network approaches that of an open network with Poisson arrival rates given by λ_{rM}. For more details about this approach see section 5.7.

A closely related technique was proposed by Whitt [Whi84]. He suggested a transformation of the same type of queueing network into an open network, and devised an algorithm for finding the arrival rates of the open network by a hierarchy of upper and lower bounds using an iterative algorithm. We will extend Whitt's algorithm to the case of mixed networks in chapter 5. Lavenberg's approximation will become the asymptotic limit in our algorithm.

3.5 Other Approximate Algorithms

There are two approximate algorithms that have attracted considerable attention in the literature. These are:

- The hierarchical decomposition technique or Norton's theorem
- The diffusion approximation

These techniques have been applied to single-class queueing networks with some degree of success. However, the extension of these two technique to multiple-class queueing networks has not been very successful.

3.5.1 Hierarchical Decomposition and Norton's Theorem

In some queueing network models certain subnetworks can be identified for which the average rate of interactions among the service centers in the subnetwork is orders of magnitude higher than the rate of interaction between the subnetwork and the rest of the network. Such subnetworks are said to be "weakly coupled" to the rest of the network. This term describes the fact that the behavior of the service centers in the subnetwork is much more influenced by that of the other service centers in the subnetwork than by that of the rest of the network. If weakly coupled subnetworks in a queueing network can be identified, then we can obtain separate solutions of these subnetworks and then aggregate the results to obtain a solution for the original network. This technique is based on a methodology proposed by Courtois [Cou75], [Cou77].

Most of the decomposition techniques are based on the Chandy-Herzog-Woo theorem [Cha75], which is also referred to as Norton's theorem. The basic result of this theorem states that in a queuing network a subnetwork may be replaced by a "flow equivalent" load dependent service center. This method is proven to be exact for product form queueing networks. Norton's theorem has been applied to nonproduct form networks, especially for modeling simultaneous resource possession, with some degree of success.

A typical example is shown in figure 3.1. The network given in figure 3.1 can be solved by decomposition in the following manner. First, the subnetwork containing service centers 1 and 2 is solved by "shorting" it from the rest of the system. The shorted subnetwork is then solved for all possible populations K of jobs in the corresponding subnetwork of the original network. The values of throughput rates $\lambda(K)$ for all possible values of K are found. Finally the subnetwork is replaced by a single flow equivalent load dependent service center which has a service rate of $\mu(K) = \lambda(K)$. This method has severe limitations in its application to multiple-class networks. There is in general no reduction in computational complexity when going from a direct exact solution to one based on decomposition. Decomposition has been applied to simultaneous resource possession problems and to parametric analysis of single-class networks.

3.5.2 Diffusion Approximations

The idea of the diffusion approximation in queueing systems was originally proposed by Newell [New71] and Gaver and Shedler [Gav71]. The method was applied to a single G/G/1 queue. It consists of substituting the integer-valued queue length with a continuous random variable that has as probability density function the solution to a diffusion equation, subject to appropriate boundary condition. The continuous random variable, by the central limit theorem, has a normal distribution under heavy traffic. Gelenbe and Mitrani [Gel80], in the most general case, apply the diffusion approximation to multiple-class open networks with FCFS service centers. Efforts to apply the diffusion approximation to closed single or multiple-class QNs have generally not been very successful [Rei74].

3.6 Conclusions

Most of the methods discussed in this chapter apply to load independent product form networks with fixed rate single server service centers and IS service centers. The main drawback in the iterative algorithms such as Linearizer and Bard-Schweitzer is the lack of any formal proof of the convergence, existence, or uniqueness of the solution. Extensions and improvements to the MVA-based approximate algorithms should consider these requirements. Hierarchical approximation techniques seem to be one of the most promising approaches, since they allow for a smooth tradeoff between accuracy and cost. Future work in this area should look for algorithms that are asymptotically correct. There is no generally accepted approximation technique for solving multiple server and/or load dependent QNs satisfactorily. Except for the Asymptotic Expansion Algorithm, no other algorithm discussed provides error bound for the approximate performance measures. Therefore, the accuracy of all the approximations has to be verified experimentally, but there are no generally accepted standard for testing the accuracy of these approximations. Consistent and comprehensive frameworks and standards must be developed to facilitate the validation process of these approximations. In conclusion, there is much urgent demand for improvements, extensions, and formalizations of the algorithms for the approximate solution of product form queueing networks. In light of the expanding roles of distributed computing systems, these algorithms will play a crucial role in the design and evaluation of such systems.

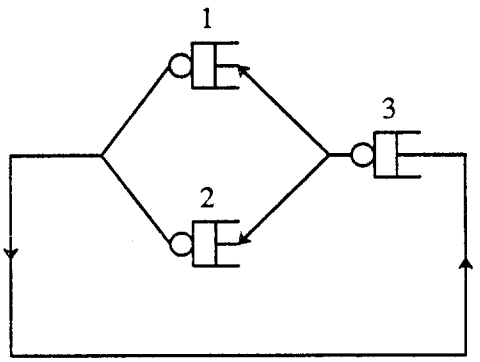

Figure 3.1 Original Network

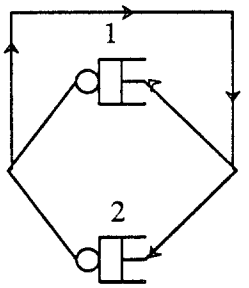

Figure 3.2 Shorted Subnetwork

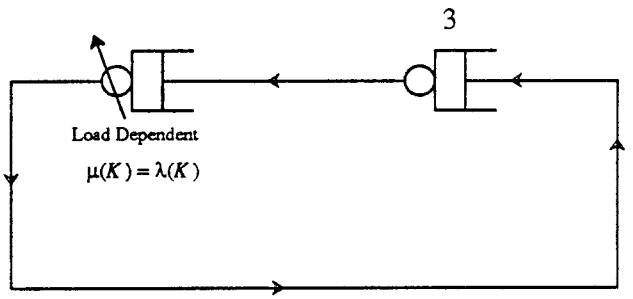

Figure 3.3 Decomposed Network

Chapter 4

An MVA-Based Approximation For Product Form Queueing Networks

4.1 Introduction

The existence of efficient computational algorithms is the strong point of product form queueing networks. The convolution algorithm for single-class product form queueing networks was first proposed by Buzen [Buz73], and then extended by Reiser and Kobayashi [Rei75] to multiple-class networks. A problem associated with this algorithm is that, when the population vector gets too large, the value of the normalizing constant $G(\mathbf{K})$ may exceed the floating point range of the processor being used, thereby causing overflow and/or underflow conditions [Cha80]. A dynamic scaling technique has been introduced that partially alleviates these problems [Lam83]. The exact MVA algorithm, proposed by Reiser and Lavenberg [Rei80], does not suffer from overflow problems, but underflow may still occur when solving load dependent networks [Rei81].

It was demonstrated in chapter 2 that both the exact MVA and the convolution algorithms have computational complexities so large that their applications to the solution of large product form queueing network models are severely limited. The need to solve increasingly larger models has motivated substantial research efforts in recent years to search for approximate algorithms. The MVA-based approximations have been among the most popular approaches considered. A basic motivation behind this popularity is the simplicity of the MVA equations. Most of the MVA-based approximate algorithms have been introduced to accomplish some or all of the following objectives:

- To reduce the computational cost drastically.
- To achieve a high level of accuracy.
- To develop approximate algorithms for which existence, uniqueness, and fast convergence of the solutions can be established.

The first objective has been the driving point for most of the algorithms. This objective is usually realized by searching for ways to eliminate the recursive structure of the MVA equations, thereby

reducing the computational cost of solving product form queueing networks drastically. The second objective is mostly achieved through heuristics and verified by means of experimentation. The third goal, though much desirable, is only attained by few existing approximate algorithms. The Bard-Schweitzer iterative algorithm, as discussed in section 3.2.1, and the Linearizer algorithm, reviewed in section 3.2.2, are heuristics for which no proofs of existence, uniqueness, or convergence of the algorithms are known. A more rigorous and analytically sound approach is the one that reduces the computational complexities to levels similar to those of the Bard-Schweitzer and the Linearizer algorithms, and at the same time transforms the exact MVA equations into a set of nonlinear equations for which the desirable properties of existence, uniqueness, and convergence of the solution in the feasible region can be established. The algorithms proposed by Chow [Cho83] and Lavenberg [Lav83] are among those following this approach.

Two prime motivations are behind the approaches taken in this chapter:

(1) There is much room remaining for improving, generalizing, and extending the MVA-based approximate algorithms. This motivation is justified in the discussion of an existing approximation in section 4.2.

(2) A substantial amount of information provided by the parameters of a given model is discarded when most of the currently existing approximate MVA techniques are employed to solve the model. The body of information that is available can be utilized to build multilevel approximate algorithms, which can achieve a much higher degree of accuracy with a marginal increment in the computational cost with respect to the existing approximate algorithms. We demonstrate the validity of this argument by introducing a multilevel algorithm in section 4.7. This approach has the added advantage of yielding solutions that can achieve all three objectives mentioned earlier.

In this chapter, with no loss of generality (as discussed in section 2.1), we consider non class-hopping product form queueing networks consisting of only fixed rate single server and IS service centers. Most of the parameters used in this chapter are defined in Appendix A. Those parameters that appear exclusively in some sections of this chapter are defined in the course of our discussion.

4.2 An MVA-Based Approximate Algorithm

Before presenting the derivation of our first proposed approximation, we attempt to justify motivation (1) discussed in the previous section by examining an existing approximate algorithm.

Let the population vector $\mathbf{K} = (K_1, K_2, \ldots, K_R)$ be such that $\mathbf{K} = K\gamma$, where the vector γ is defined as $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_R)$, and $K = \sum_{i=1}^{R} K_i$ is the total population of all classes. The following relationships hold:

$$K = \sum_{i=1}^{R} K_i \; ; \quad \gamma_i = \frac{K_i}{K} \; ; \quad \sum_{i=1}^{R} \gamma_i = 1, \tag{4.2.1}$$

γ is the population mix vector of a multiple-class queueing network. Assume, with no loss of generality, that the network consists of M service centers and that the first M_1 service centers (with $M_1 < M$) are of type 1, 2, or 3 (using the notation introduced in chapter 2). Service centers $M_1 + 1$ to M are assumed to be of type 4 (IS) service centers.

We now examine an approximate algorithm proposed by Chow [Cho83]. This algorithm is also closely related to the Bard-Schweitzer algorithm. A general algorithm is then proposed which greatly improves the accuracy over this algorithm and other similar techniques. In the study of MVA-based approximations in section 3.2.3, we stated that Chow [Cho83] defined a differential error term $\delta_{ij} = \dfrac{N_j(\mathbf{K} - e_i) - N_j(\mathbf{K})}{N_j(\mathbf{K})}$. He then proceeded to replace the $N_j(\mathbf{K}-e_i)$ term in the exact MVA equations with its approximate value assuming $\delta_{ij}=0$, to produce a set of R nonlinear equations in terms of

the unknown mean throughput rates for each class at some service center m. The resulting set of non-linear equations always produces a unique solution in the feasible region. The first approximation is very similar and closely related to that used in the Bard-Schweitzer algorithm (see section 3.2.3 for more details).

To improve the accuracy, Chow tried to estimate the δ_{ij} terms by using the first approximation. This was done by solving the network approximately at population vectors $\mathbf{K}$ and $\mathbf{K}-e_i$, with δ_{ij}=0. He then estimated δ_{ij} by the following approximation:

$$\delta_{ij}^{*} = \frac{N_j^{*}(\mathbf{K}-e_i) - N_j^{*}(\mathbf{K})}{N_j^{*}(\mathbf{K})}, \tag{4.2.2}$$

where * superscripts denote the estimated values of the corresponding parameters.

This step produced a noticeable improvement only if the estimates obtained using the first approximation were accurate. The error results for the approximation and the improvement were reported to be similar to those of the Linearizer algorithm.

The approximation obtained by assuming δ_{ij}=0 can be improved substantially if we observe that in fact the one chain i customer removed from the population vector $\mathbf{K}$ (i.e., $\mathbf{K}-e_i$) in a perfectly balanced network would reduce the mean number of customers at service center s by $\frac{1}{|S(i)|}$ (i.e., $N_j(\mathbf{K}) - N_j(\mathbf{K}-e_i) = \frac{1}{|S(i)|}$). Let:

$$\Delta_{ij} = N_j(\mathbf{K}) - N_j(\mathbf{K}-e_i)\,; \tag{4.2.3}$$

then, assuming that $\Delta_{ij} = 0$ can cause large errors when the population vector $\mathbf{K}$ has small components ($\Delta_{ij} = 0$ is equivalent to $\delta_{ij} = 0$). In addition, in any network, the following identity always holds:

$$\sum_{j=1}^{M}\left[N_j(\mathbf{K}) - N_j(\mathbf{K}-e_i)\right] = \sum_{j=1}^{M}\Delta_{ij} = 1\,. \tag{4.2.4}$$

Furthermore, the value of Δ_{ij} could in fact be as large as 1. We will discuss and analyze in detail in chapter 6 the networks that produce such large values for the Δ_{ij} terms. It should be pointed out here that, in multichain queueing networks, there are cases where specific population mix vectors, at values of K ranging from small to very large, do indeed induce values close to 1 for Δ_{ij}. These cases do not cause large errors for networks with large population vectors (large $K\gamma_i$ for all i). However, large errors can occur if one or more service centers can become saturated due to some (or all) chains with small populations visiting those service centers. More interestingly, in some queueing networks, if the number of service centers M is larger than the number of classes R, up to R service centers may become saturated [Bal87]. Similar arguments hold true for the Bard-Schweitzer algorithm. These properties motivated us to define a general correction term to the difference between the $N_j(\mathbf{K})$ and $N_j(\mathbf{K}-e_i)$. This correction term is:

$$\Delta_{ij} = N_j(\mathbf{K}) - N_j(\mathbf{K}-e_i) = \frac{f_j(R,M,\mathbf{K})}{|S(i)|}. \tag{4.2.5}$$

A simple version of this correction term will be used in our first attempt. We will, however, use equation (4.2.5) in the remaining of this section to derive a set of nonlinear equations replacing the exact MVA equations.

Realizing that $\sum_{i=1}^{R} N_{ij}(\mathbf{K}-e_i) = N_j(\mathbf{K}-e_i)$, and applying Little's law to (2.2.3), we can rewrite the exact MVA equations given by (2.2.3) in the following form:

$$N_{ij}(\mathbf{K}) = \begin{cases} \lambda_{ij}(\mathbf{K})s_{ij}\left[1 + N_j(\mathbf{K}-e_i)\right] & j=1,2,\ldots,M_1 \qquad (4.2.6) \\ \lambda_{ij}(\mathbf{K})s_{ij} & j=M_1+1,\ldots,M\,, \qquad (4.2.7) \end{cases}$$

substituting for $N_j(\mathbf{K}-e_i)$ from (4.2.3) yields the following relationship:

$$N_{ij}(\mathbf{K}) = \lambda_{ij}(\mathbf{K})s_{ij}\left[1 + N_j(\mathbf{K}) - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right], \quad j=1,2,...,M_1; \tag{4.2.8}$$

summing (4.2.8) over all possible values of i we get the following:

$$N_j(\mathbf{K}) = \sum_{i=1}^{R}\lambda_{ij}(\mathbf{K})s_{ij}\left[1 + N_j(\mathbf{K}) - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right], \quad j=1,2,...,M_1; \tag{4.2.9}$$

solving for $N_j(\mathbf{K})$ in equation (4.2.9) we get:

$$N_j(\mathbf{K}) = \frac{\sum_{i=1}^{R}\lambda_{ij}(\mathbf{K})s_{ij}\left[1 - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right]}{(1 - \sum_{i=1}^{R}\lambda_{ij}(\mathbf{K})s_{ij})}, \quad j=1,2,...,M_1. \tag{4.2.10}$$

In the denominator of equation (4.2.8), we have

$$\sum_{i=1}^{R}\lambda_{ij}(\mathbf{K})s_{ij} = \sum_{i=1}^{R}\rho_{ij}(\mathbf{K}) = \rho_j(\mathbf{K}), \quad j=1,2,...,M_1, \tag{4.2.11}$$

where $\rho_j(\mathbf{K})$, $j=1,2,...,M_1$ is the utilization of service center j. To make the analysis simpler, assume that $\frac{f_j(R,M,\mathbf{K})}{|S(i)|}$ has the same value for all i. Replacing the values of $N_j(\mathbf{K})$ in equation (4.2.8) by their equivalent from equation (4.2.10) yields:

$$N_{ij}(\mathbf{K}) = \frac{\lambda_{ij}(\mathbf{K})s_{ij}\left[1 - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right]}{(1 - \sum_{k=1}^{R}\lambda_{kj}(\mathbf{K})s_{kj})}, \quad i=1,2,...,R, \quad j=1,2,...,M_1; \tag{4.2.12}$$

knowing that:

$$\sum_{j=1}^{M}N_{ij} = \sum_{j=1}^{M_1}N_{ij}(\mathbf{K}) + \sum_{j=M_1+1}^{M}N_{ij}(\mathbf{K}) = K_i, \quad i=1,2,...,R, \tag{4.2.13}$$

we can write:

$$\sum_{j=1}^{M_1}\frac{\lambda_{ij}(\mathbf{K})s_{ij}\left[1 - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right]}{(1 - \sum_{k=1}^{R}\lambda_{kj}(\mathbf{K})s_{kj})} + \sum_{j=M_1+1}^{M}\lambda_{ij}(\mathbf{K})s_{ij} = K_i, \quad i=1,2,...,R. \tag{4.2.14}$$

The remaining final step is to transform the set of equations in (4.2.14) into a set of R nonlinear equations and R unknowns. Assume that we are interested in obtaining the mean throughput rates of all classes at some designated service center m. The set of unknowns is then denoted by $\{\lambda_{im}(\mathbf{K}) \mid i=1,2,...,R\}$. Let $\Lambda_m(\mathbf{K}) = (\lambda_{1m}(\mathbf{K}),\lambda_{2m}(\mathbf{K}),\ldots,\lambda_{Rm}(\mathbf{K}))$. The following relationships hold for the ratios of the mean throughput rates (see chapter 2):

$$\frac{\lambda_{ij}(\mathbf{K})}{\lambda_{im}(\mathbf{K})} = \frac{\theta_{ij}}{\theta_{im}}, \quad i=1,2,...,R, \quad j=1,2,...,M; \tag{4.2.15}$$

letting ϕ_{ij} be denoted by:

$$\phi_{ij} = \frac{\theta_{ij}s_{ij}}{\theta_{im}}, \quad i=1,2,...,R, \quad j=1,2,...,M, \tag{4.2.16}$$

and substituting for $\lambda_{ij}(\mathbf{K})$ using (4.2.15) and (4.2.16), we get the following set of R nonlinear equations and R unknowns $\lambda_{im}(\mathbf{K})$, $i=1,2,...,R$:

$$F(\Lambda_m(\mathbf{K}),i) = K_i - \lambda_{im}(\mathbf{K}) \sum_{j=M_1+1}^{M} \phi_{ij} - \sum_{j=1}^{M_1} \frac{\phi_{ij}\lambda_{im}(\mathbf{K})\left[1 - \frac{f_j(R,M,\mathbf{K})}{|S(i)|}\right]}{(1 - \sum_{k=1}^{R} \phi_{kj}\lambda_{km}(\mathbf{K}))} = 0,$$

$$i=1,2,...,R. \tag{4.2.17}$$

For the set of nonlinear equations, any feasible solutions must be such that $0 \le \rho_m(\mathbf{K}) < 1$, i.e., the utilization of service center m, $m=1,2,...,M_1$, cannot exceed unity. Hence, the feasible solution should always be contained in the region defined by:

$$B = \left\{ \Lambda_m(\mathbf{K}) \mid 0 \le \rho_m(\mathbf{K}) < 1, \lambda_{im}(\mathbf{K}) \ge 0 \text{ for } all\ i \right\} \tag{4.2.18}$$

To summarize, so far we have proposed a new approximation derived by substituting for $N_j(\mathbf{K} - e_i)$ using the relationship given in (4.2.5) into the exact MVA equations. The approximation has been used to transform the exact MVA equation into a set of nonlinear equations in terms of the mean throughput rates at some arbitrary service center m given by (4.2.14). The feasible region for the solution of equations in (4.2.17) is defined by the set B given in (4.2.18). Throughout the rest of this chapter, we assume that $0 < \frac{f_j(R,M,\mathbf{K})}{|S(i)|} < 1$.

In the next two sections it is proven that the set of equations defined by (4.2.17) always possesses a unique solution in the feasible region B defined by (4.2.18).

4.3 Existence of a Solution

In this section we prove that the set of nonlinear equations given by (4.2.17) always has a solution in the feasible region defined by the set B given in (4.12.18). We first introduce the following notations to simplify (4.2.17) and (4.2.18). Let:

$$x_i = \lambda_{im}(\mathbf{K}), \quad i=1,2,...,R,$$

$$b_i = \sum_{j=M_1+1}^{M} \phi_{ij}, \quad i=1,2,...,R,$$

$$\sigma_j = 1 - \rho_j = 1 - \sum_{k=1}^{R} \phi_{kj} x_k, \quad j=1,2,...,M_1,$$

and $\mathbf{x} = (x_1, x_2, \ldots, x_R)$. Using the notations defined above, the set of nonlinear equations given in (4.2.17) can be written as follows[1]:

$$F(\mathbf{x},i) = K_i - x_i b_i - \sum_{j=1}^{M_1} \phi_{ij} x_i \left(1 - \frac{f_j}{|S(i)|}\right)\left(1 - \sum_{k=1}^{R} \phi_{kj} x_k\right)^{-1} = 0,$$

$$i=1,2,...,R, \tag{4.3.1}$$

and the feasible region given by (4.2.18) becomes:

$$B = \left\{ \mathbf{x} \mid 0 \le \sum_{k=1}^{R} \phi_{kj} x_k < 1, \ \mathbf{x} \ge 0, \text{ for } all\ j \right\}. \tag{4.3.2}$$

1 To simplify the notations, we have replaced $f_j(R,M,\mathbf{K})$ by f_j

We next use Brouwer's fixed point theorem [Ort70] in a similar but not identical way as Chow [Cho83] did to prove that (4.3.1) has a solution in the feasible region defined by (4.3.2). Obviously $\mathbf{x} = 0 = (0,0,...,0)$ cannot be a solution of (4.3.1), since this requires $K_i=0$ for all i.

Brouwer's fixed point theorem states that, if $\mathbf{H}: D \subset R^n \rightarrow R^n$ is a continuous mapping on a compact, convex set D such that $\mathbf{H}(\mathbf{x}) \in D$ for every $\mathbf{x} \in D$, then $\mathbf{H}(\mathbf{x}) = \mathbf{x}$ has a solution in D. We will construct such an $\mathbf{H}$ and D. Let $\mathbf{G}(\mathbf{x}) = (G_1(\mathbf{x}),G_2(\mathbf{x}),...,G_R(\mathbf{x}))$, where:

$$G_i(\mathbf{x}) = K_i \left[\frac{\sum_{j=1}^{M_i} \phi_{ij}(1 - \frac{f_j}{|S(i)|})}{1 - \sum_{k=1}^{R} \phi_{kj} x_k} + b_i \right]^{-1}, \quad i=1,2,...,R. \tag{4.3.3}$$

We define the function $\mathbf{H}(\mathbf{x}) = (H_1(\mathbf{x}),H_2(\mathbf{x}),...,H_R(\mathbf{x}))$ on the set C by:

$$H_i(\mathbf{x}) = \begin{cases} G_i(\mathbf{x}) & \mathbf{x} \in C \\ 0 & \mathbf{x} \notin C \end{cases}, \qquad i=1,2,...,R, \tag{4.3.4}$$

where the set C is defined by:

$$C = \left\{ \mathbf{x} \mid 0 \le x_i \le G_i(0),\ 0 \le \sum_{k=1}^{R} \phi_{kj} x_k < 1, \right\}. \tag{4.3.5}$$

Let us also define a set D as:

$$D = \left\{ \mathbf{x} \mid 0 \le \mathbf{x} \le \mathbf{G}(0) \right\}. \tag{4.3.6}$$

It is easy to show that $G_i(\mathbf{x})$ is a decreasing function of $\mathbf{x}$. This is true since:

$$\frac{K_i \partial (G_i(\mathbf{x}))^{-1}}{\partial x_k} = \sum_{j=1}^{M_i} \phi_{ij} \phi_{kj} x_i (1 - \frac{f_j}{|S(i)|}) \sigma_j^{-2} > 0, \quad k=1,2,...,R, \quad k \neq i\ ; \tag{4.3.7}$$

clearly, for all $x_i > 0$, the above relation implies that the $G_i(\mathbf{x})$ are decreasing functions of $\mathbf{x}$. For the case $k=i$, we can write:

$$\frac{K_i \partial (G_i(\mathbf{x}))^{-1}}{\partial x_i} = \sum_{j=1}^{M_i} (\phi_{ij} \sigma_j^{-1} + \phi_{ij}^2 \sigma_j^{-2} x_i)(1 - \frac{f_j}{|S(i)|}) > 0, \quad i=1,2,...,R. \tag{4.3.8}$$

Therefore, we have

$$G_i(\mathbf{x}) \le G_i(0) < \infty \quad \text{for } \textit{all } i \text{ and } \mathbf{x} \ge 0. \tag{4.3.9}$$

Thus, $G_i(\mathbf{x})$ is continuous on D (note that, if $\rho_j=1$, $H_i(\mathbf{x}) = G_i(\mathbf{x}) = 0$). The set B is closed, and bounded by linear constraints. Therefore B is a compact and convex set. If $\mathbf{x} \in D$, then $0 \le H_i(\mathbf{x}) \le H_i(0)$ for all i. Therefore $\mathbf{H}(\mathbf{x}) \in D$. This by Brouwer's theorem requires $\mathbf{H}(\mathbf{x})=\mathbf{x}$ to have a solution in D. Next we show that any solution in D is also in C. Suppose that there is a solution $\mathbf{x}$ in D that is not in C, and that $\sum_{k=1}^{R} \phi_{kj} x_k \ge 1$. Then this assumption implies that $\mathbf{x}$ is also a solution of (4.3.1), and requires that $\sigma_j = 1 - \sum_{k=1}^{R} \phi_{kj} x_k \le 0$.

This implies that:

$$(\sum_{j=1}^{M_i} \phi_{ij}(1 - \frac{f_j}{|S(i)|}) + b_i) x_i \ge K_i\ . \tag{4.3.10}$$

But we also know that $0 \le x_i \le H_i(0)$. Thus, evaluating $H_i(0)$ from (4.3.3) and substituting in the inequality $x_i \le H_i(0)$ requires the following relationship to hold:

$$(\sum_{j=1}^{M_1}\phi_{ij}(1-\frac{f_j}{|S(i)|})+b_i)x_i \le K_i \ , \tag{4.3.11}$$

this is a contradiction to the inequality obtained in (4.3.10). Hence, every solution of $\mathbf{H}(\mathbf{x})=\mathbf{x}$ in D must also be in C. Since $C \subset B$, any solution of (4.3.1) in C must be confined to B. Therefore, the set of nonlinear equations defined by (4.3.1) has a solution in the feasible region B.

4.4 Uniqueness of the Solution

In this section we prove that the set of equations given by (4.3.1) always has a unique solution in the feasible region defined by B. We give a proof by contradiction similar to that in [Cho83]. First assume $R>1$.

Suppose that (4.3.1) does not have a unique solution, and that there are at least two nonzero solutions $\mathbf{x}$ and $\mathbf{x}'$. Then, assuming $\mathbf{z}=\mathbf{x}'-\mathbf{x}\neq 0$, we can write:

$$z_i^{-1}(F(\mathbf{x},i)-F(\mathbf{x}',i))=\sum_{j=1}^{M_1}\phi_{ij}(1-\frac{f_j}{|S(i)|})(\sigma_j'^{-1}+\frac{x_i}{z_i}\sum_{k=1}^{R}\phi_{kj}z_k\sigma_j^{-1}\sigma_j'^{-1})+b_i=0 \ , \tag{4.4.1}$$

where $\sigma_j'=1-\rho_j'=1-\sum_{k=1}^{R}\phi_{kj}x_k'$. Since $x_i>0$, $b_i\ge 0$, $\sigma_j>0$, $\sigma_j'>0$, and $\phi_{kj}\ge 0$ for all $j=1,2,...,M_1$, the above equation implies that:

$$\sum_{j=1}^{M_1}\frac{\phi_{ij}}{z_i}(1-\frac{f_j}{|S(i)|})\sum_{k=1}^{R}\phi_{kj}z_k\sigma_j^{-1}\sigma_j'^{-1}<0 \ ; \tag{4.4.2}$$

multiplying the inequality by z_i^2 and summing over all i, we get

$$\sum_{j=1}^{M_1}\sum_{i=1}^{R}\phi_{ij}z_i(1-\frac{f_j}{|S(i)|})\sum_{k=1}^{R}\phi_{kj}z_k\sigma_j^{-1}\sigma_j'^{-1}<0 \ , \tag{4.4.3}$$

which is equivalent to:

$$\sum_{j=1}^{M_1}(\sum_{k=1}^{R}\phi_{kj}z_k)^2\sigma_j^{-1}\sigma_j'^{-1}(1-\frac{f_j}{|S(i)|})<0 \ . \tag{4.4.4}$$

Since the left hand side of (4.4.4) is strictly positive, the only way the inequality may be satisfied is to have $z_i=0$ for all i. Thus, $\mathbf{z}=0 \rightarrow \mathbf{x}=\mathbf{x}'$, and the solution of (4.3.1) is unique.

For the case $R=1$, it is easy to show, using the fact that the functions $G_i(\mathbf{x})$ are decreasing, that: $\frac{\partial F(\mathbf{x},i)}{\partial x_i}<0$, $\frac{\partial F(\mathbf{x},i)}{\partial x_k}<0$, $k\neq i$, $\frac{\partial^2 F(\mathbf{x},i)}{\partial x_i^2}<0$.

Therefore, $F(\mathbf{x},i)$ is a monotonically decreasing and convex function, and since $F(0,i)=K_i>0$, and $F(\min_j \phi_{ij}^{-1})=-\infty$, the set of equations (4.3.1) has a unique solution in the feasible region.

4.5 Computational Algorithm

There are two distinct steps to be taken for the implementation of the algorithm we proposed in section 4.2. First, the set of nonlinear equations (4.2.17)-(4.2.18) has to be solved. By solving these equations, we obtain the mean throughput rates at some designated center m. Second, using the values of the mean throughput rates obtained in the first step, all other remaining performance measures can be evaluated. These performance measures include: the remaining per class mean throughput rates at service centers other than m, the per class mean queue lengths, the per class mean waiting times, and the per class utilization at each of the service centers. Any other performance measures (such as cycle times) can be obtained from these metrics.

Assume that the probability transition matrix for each class of jobs, the mean service time demand matrix, and the service center disciplines for a network are given. To solve the network using the approximation given by (4.2.17) in the feasible region denoted by (4.2.18), we perform the following steps:

(1) Evaluate the visit ratios using equation (2.1.5).

(2) Evaluate the parameters ϕ_{ij} using (4.2.16).

(3) Choose the proper values of the correction terms $f_j(R,M,\mathbf{K})$, $i=1,2,...,R$, $j=1,2,...,M_1$.

(4) Solve the set of nonlinear equations given by (4.2.17). The unique solution for $\Lambda_m(\mathbf{K}) = (\lambda_{1m}(\mathbf{K}),\lambda_{2m}(\mathbf{K}), \ldots, \lambda_{Rm}(\mathbf{K}))$ must lie in the feasible region defined by (4.2.18).

(5) Evaluate the throughput rates $\lambda_{ij}(\mathbf{K})$, $i=1,2,...,R$, $j=1,2,...,M$, $j \neq m$, using (4.2.14).

(6) Evaluate the per class utilizations $\rho_{ij}(\mathbf{K})$, $i=1,2,...,R$, $j=1,2,...M_1$, using (4.2.11).

(7) Evaluate the per class mean queue length and the mean queue length at each service center using (4.2.12) and (4.2.10) respectively.

(8) The per class mean waiting times $W_{ij}(\mathbf{K})$, $i=1,2,...,R$, $j=1,2,...,M$ can be computed using Little's law.

A program to solve the set of nonlinear equations in (4.2.17) and another one to evaluate the performance measures were implemented on an IBM 3090/200 in a CMS operating system environment. The solution program makes use of the nonlinear equation solution package Minpack-1 [Mor80].

The subroutine used for the solution of (4.2.17), Hybrid1, is a modified Minpack-1 version of Powell's hybrid method [Pow70]. The method chooses a correction term as a convex combination of the Newton and scaled gradient directions, and updates the Jacobian of the nonlinear equations by the rank-1 method of Broyden [Bro65]. The choice of the correction term, under reasonable conditions, guarantees a very fast rate of convergence for starting points (initial seeds) far from the solution. The Jacobians are approximated by forward difference at the starting points, and they are not used again unless the rank-1 method fails to produce satisfactory progress toward the solution. As we shall see, the execution times for the solution of the set of nonlinear equations for the networks we used in our experiments were very small, suggesting a very fast convergence. Minpack-1's Hybrid1 subroutine has a built-in flag that invokes a warning whenever the number of iterations exceeds $200\times(N+1)$, where N is the number of unknowns (in our case $N=R$). This flag was never invoked in our experiments. This was partially due to the fact that we investigated the proper choice of the initial seeds (starting points). As a first guess we used an upper bound defined by:

$$\lambda_{im}(\mathbf{K}) \leq \frac{K_i}{s_{im}}, \quad i=1,2,...,R. \tag{4.5.1}$$

The proof of the above relation is easy to obtain if we observe that, in the MVA equations given by (4.2.6) and (4.2.7), the maximum value for the $N_{ij}(\mathbf{K})$ values at each service center is K_i, and the minimum value on the right hand side of these equations is $\lambda_{ij}(\mathbf{K})$ for all i, j. But these bounds, as well as the single value bounds introduced in chapter 3, are loose and sometimes fall outside the feasible region defined by (4.2.18). We will use the bounds given by (4.5.1) as the initial conditions for an iterative algorithm we will define in section 5.5. The best initial seeds were found to be small non-negative values. All the operations in Minpack-1 subroutines as well as the computation of the performance measures were carried out using double precision. The error tolerance bound for the stopping rule is automatically set equal to the machine precision by the Minpack-1 function DPMPAR(1). The implementation of the approximate algorithm proposed and used to obtain the numerical results of this chapter consists of approximately 850 lines of code. This include the subroutines for evaluating the visit ratios from the transition probability matrix and the driver subroutine for Minpack-1.

4.5.1 Computational Complexities of the Algorithm

The derivation of space and time complexities of our algorithm can be divided into two parts. The first part is related to the computational complexities of the solution of the set of nonlinear equations. This part is determined by Minpack-1 and by the set of equations defined by (4.2.17). The second part is associated with the evaluation of the visit ratios and of the performance measures of the queueing network. The first part corresponds to steps (2), (3), and (4) of the computational algorithm described in section 4.5, while the second part corresponds to the remaining steps.

Each call to the Hybrid1 subroutine of Minpack-1 requires about $\frac{23}{2}R^2$ [Mor80] operations to solve the set of R nonlinear equations and R unknowns, where R is the number of chains in the network. As with any solution technique for nonlinear equations, the time complexity reported in [Mor80] for Minpack-1 depends on the stopping rule criterion. Therefore, $\frac{23}{2}R^2$ is a rough estimate of the time complexity. To evaluate the set of equations in (4.2.17) for all classes, we need $RM_1(R+5)+3R(M+1)$ operations (M_1, as defined earlier, is the number of non IS service centers). This total includes the $2RM$ operations needed to evaluate the parameters ϕ_{ij}, i=1,2,..,R, j=1,2,...,M. The second part of the computation involves the evaluation of the visit ratios θ_{ij}, and the per class performance measures ρ_{ij}, λ_{ij}, N_{ij}, and W_{ij}. We have already derived the computational complexity of the evaluation of visit ratios in chapter 2 and found it to be $2RM^3$. We need $2R(M-1)$ operations to compute the mean throughput rates λ_{ij} for all i,j, $j \neq m$. The evaluation of the ρ_{ij}'s requires RM_1 operations. To compute the per class mean queue lengths $N_{ij}(\mathbf{K})$, we use (4.2.12) for the first M_1 service centers, and (4.2.7) for service centers M_1+1 through M (IS service centers). The resulting complexity is $7RM_1$ operations for the non IS service centers and $(M-M_1)R$ for the IS service centers. Finally, the evaluation of the per class mean waiting times requires RM operations. Adding all the operations needed, the time complexity $T(R,M,M_1)$ of the algorithm is found to be:

$$T(R,M,M_1) = R^2(M_1+23/2) + RM(2M^2+7) + R(12M_1+1)\,. \tag{4.5.2}$$

The evaluation of the space complexity of the algorithm consists of the storage space required to solve the set of nonlinear equations defined by (4.2.14), using Minpack-1 and the storage space needed to store the per class values of θ_{ij}, ϕ_{ij}, $\lambda_{ij}(\mathbf{K})$, $f_j(R,M,\mathbf{K})$, $S(i)$, $N_{ij}(\mathbf{K})$, $\rho_{ij}(\mathbf{K})$, K_i, and $W_{ij}(\mathbf{K})$. The storage space required for these parameters amounts to $5RM+2R+M+RM_1$ double precision storage locations. Hybrid1 requires $\frac{(3R^2+17R)}{2}$ double precision storage locations (this is in addition to the storage required by the subroutine itself). Therefore the space complexity $S(R,M,M_1)$ of the algorithm is:

$$S(R,M,M_1) = R(5M+M_1+21/2+1.5R)+M\,. \tag{4.5.3}$$

Clearly, the space and time complexities of our algorithm are smaller than those of Linearizer, and of the same order as those of the Bard-Schweitzer algorithm.

4.6 Experimental Results

To evaluate the accuracy of the approximate algorithm defined by (4.2.17) and to gain more insight, a number of experiments were performed. The exact and approximate solutions were obtained for the sample networks under considerations. The approximation is intended for large networks with large numbers of classes, service centers, and customers. Naturally, we were interested in investigating the accuracy of the algorithm for these kinds of networks. However, due to limitations in time and space complexities associated with the exact solution techniques, we chose networks for which the exact solution techniques could be obtained up to a reasonably large population vector and number of classes R. To solve exactly the set of networks in the experiments, RESQ2 [Sau82] was used. Using RESQ2, networks with three classes or less can practically be solved with population vectors up to

$\mathbf{K} = (30,30,30)$. (RESQ2 requires about 12 Mbytes to solve a network with $M=4$, $R=3$, and a population vector of $\mathbf{K} = (30,30,30)$).

Error Criteria

To compare the accuracy of the approximate algorithm, we use three different criteria. A "queue length error tolerance" and a "maximum queue length error tolerance", first introduced by Chandy et al. [Cha75], has traditionally been used for the MVA-based approximations. The queue length error tolerance $E_t(N_{ij})$ is denoted by:

$$E_t(N_{ij}) = \frac{|N_{ij} - N_{ij}^*|}{K_i}, \quad i=1,2,...,R, \quad ,j=1,2,...,M \; ; \tag{4.6.1}$$

the matrix representing the $E_t(N_{ij})$ elements will be denoted by:

$$E_t(R,M) = [E_t(N_{ij})], \quad i=1,2,...,R, \quad j=1,2,...,M. \tag{4.6.2}$$

The maximum queue length error tolerance is defined by:

$$E_{tmax}(N_{ij}) = \max_{i,j} \frac{|N_{ij} - N_{ij}^*|}{K_i} \; . \tag{4.6.3}$$

A third error criterion which we will refer to as the *absolute error criterion* is defined by:

$$E_a(N_{ij}) = \frac{|N_{ij} - N_{ij}^*|}{N_{ij}}, \quad i=1,2,...,R, \quad j=1,2,...,M \; . \tag{4.6.4}$$

The matrix with elements $E_a(N_{ij})$, is given by:

$$E_a(R,M) = [E_a(N_{ij})], \quad i=1,2,...,R, \quad j=1,2,...,M \; ; \tag{4.6.5}$$

similarly, the maximum absolute error criterion is denoted by:

$$E_{amax}(N_{ij}) = \max_{i,j} \frac{|N_{ij} - N_{ij}^*|}{N_{ij}} \; . \tag{4.6.6}$$

In the above equations, N_{ij} and N_{ij}^* represent the exact and the estimated mean queue lengths of chain i jobs at service center j. The $\mathbf{K}$ arguments are omitted from the mean queue length parameters. The elements of the matrices $E_t(R,M)$ and $E_a(R,M)$ are given in percentage form. To compare the relative magnitudes of the maximum queue length error tolerance $E_{tmax}(N_{ij})$ and the maximum absolute error criterion $E_{amax}(N_{ij})$ at the same population vector $\mathbf{K}$, we define their ratio as:

$$R_{ta}(\mathbf{K}) = \max_{ij} \frac{E_t(N_{ij})}{E_a(N_{ij})} \; . \tag{4.6.7}$$

We also define a parameter which is an indication of the stretch in the loading factor is denoted by:

$$V_{L_{ij}} = \frac{\max_{ij} L_{ij}}{\min_{ij} L_{ij}} \; , \tag{4.6.8}$$

The $V_{L_{ij}}$ factor is an indication of the variability of the loadings at different service centers in a queueing network.

Let:

$$\Theta(R,M) = [\theta_{ij}], \quad i=1,2,...,R, \quad j=1,2,...,M \; . \tag{4.6.9}$$

$\Theta(R,M)$ is referred to as the *visit ratios matrix*. Similarly, assume that the *mean service time matrix* is denoted by:

$$S(R,M) = [s_{ij}], \quad i=1,2,...,R, \quad j=1,2,...,M \ . \tag{4.6.10}$$

We now turn our attention to several examples. The purpose of these examples is to introduce the characteristics of our algorithm, to compare it with similar algorithms, and to gain insight into the problems of extending and improving the algorithm. The networks in these examples are defined and characterized by the visit ratios matrix $\Theta(R,M)$, the mean service demand matrix $S(R,M)$, the population mix vector γ, the total population K, and the service disciplines of the service centers in the QN. The queue length is the most sensitive performance measure with respect to variations in queueing network parameters. Therefore, the queue length is generally the preferred performance measure for testing the accuracy of approximate algorithms for the solution of queueing networks. Other performance measures (namely the mean waiting times, the mean throughput rates, and the utilizations) are more robust, producing smaller values of error tolerances and absolute error criteria. We therefore chose to examine the accuracy of our algorithms using the mean queue length as the parameter for testing the accuracy of the algorithms. In each of these examples, the queue length error tolerance and the absolute error criterion matrices will be evaluated and compared. For the remainder of this chapter, we refer to our approximation defined by (4.2.17) as the initial approximation (or algorithm I). We used $f_j(R,M,\mathbf{K}) = \frac{1}{R}$ in all the examples. However, during the course of our experimentation, we realized that $f_j(R,M,\mathbf{K}) = \frac{1}{MR}$ produced smaller absolute errors for more balanced networks. This was expected, since in more balanced networks Δ_{ij}'s are more balanced and small in value. On the other hand, in an imbalanced network Δ_{ij} could be close to 1 for bottleneck centers.

As a first example, consider a network with $M=4$, $R=3$, $\mathbf{K} = (2,3,2)$. Let the visit ratio matrix be given by:

$$\Theta(R,M) = \begin{bmatrix} 1.502 & .502 & .502 & 1.00 \\ 1.262 & .457 & .432 & 1.00 \\ 1.468 & .899 & .665 & 1.00 \end{bmatrix} , \tag{4.6.11}$$

and assume that the mean service demand matrix is given by:

$$S(R,M) = \begin{bmatrix} .666 & 1.992 & 1.992 & 1.00 \\ .792 & 2.188 & 2.315 & 1.00 \\ .681 & 1.112 & 1.504 & 1.00 \end{bmatrix} . \tag{4.6.12}$$

The queue length error tolerance and the absolute error criterion matrices for this queueing network and using our approximate algorithm are given by:

$$E_t(R,M) = \begin{bmatrix} 2.27 & 2.26 & 2.41 & .93 \\ 2.31 & 2.14 & 2.49 & .91 \\ 2.33 & 2.32 & 2.34 & .89 \end{bmatrix} , \quad E_a(R,M) = \begin{bmatrix} 7.63 & 7.58 & 8.13 & 8.58 \\ 7.77 & 7.17 & 8.43 & 8.64 \\ 7.87 & 7.43 & 7.88 & 8.26 \end{bmatrix} , \tag{4.6.13}$$

and for a population vector of $\mathbf{K} = (26,21,29)$ we get the following $E_t(R,M)$ and $E_a(R,M)$ matrices:

$$E_t(R,M) = \begin{bmatrix} 2.19 & 1.62 & 2.78 & .017 \\ 2.23 & 1.49 & 2.87 & .018 \\ 2.91 & 2.20 & 3.41 & .017 \end{bmatrix} , \quad E_a(R,M) = \begin{bmatrix} 6.68 & 4.86 & 2.78 & 1.35 \\ 6.81 & 4.47 & 6.85 & 1.41 \\ 6.95 & 4.76 & 7.20 & 0.96 \end{bmatrix} . \tag{4.6.14}$$

The accuracy of the algorithm is affected by the choice of the stopping rule for the solution of the set of nonlinear equations. The stopping rule is automatically set to the processor's precision by Minpack-1. We therefore did not try to test the effects of different stopping rule criteria on the accuracy and speed of convergence of the algorithm. However, none of the models solved using our

algorithm ever exceeded the limit on the number of iterations reported in section 4.5, and the internal convergence parameter limit of Minpack-1 was never violated.

The network of the first example is a case of a balanced system in the sense that the loading factors are deliberately set equal, i.e., all $L_{ij} = \theta_{ij} s_{ij}$ are the same. Clearly, the absolute errors are all less than 9%. From (4.6.14), it is clear that there is a general decrease in absolute errors for larger populations. However, this decrease is not evident from the queue length error tolerance values. Obviously, the queue length error tolerance values point to a much more optimistic error performance. Most of the accuracy tests on MVA-based approximations have been carried out using the queue length error tolerance. Examples are found in [Cha82] [Zah86]. It is interesting that, while from (4.6.13), we have $E_{tmax}(N_{ij}) = 2.49$ for class 2 jobs at service center 3, $E_{amax}(N_{ij}) = 8.64$ for the same class of jobs but at the IS service center (service center 4). It seems that the absolute error criterion provides a more realistic indication of the accuracy of approximate algorithms. The network in the first example was highly utilized with the population vector $\mathbf{K} = (26,21,29)$. The utilizations of the first three service centers were larger than .973.

As a second example, consider a network with the same dimensions for M and R, and the same visit ratio matrix, but with a mean service time matrix such that the network is lightly loaded. Let the mean service demand matrix be given by:

$$S(R,M) = \begin{bmatrix} .033289 & .099602 & .099602 & 10.00 \\ .039620 & .109409 & .115741 & 10.0 \\ .034060 & .055617 & .075188 & 10.0 \end{bmatrix} ; \qquad (4.6.15)$$

then the $E_t(R,M)$ and $E_a(R,M)$ matrices, using our approximate algorithm, and for a population vector of $\mathbf{K} = (27,30,25)$, are given by:

$$E_t(R,M) = \begin{bmatrix} .0142 & .0156 & .01697 & .106 \\ .0162 & .0123 & .02011 & .1043 \\ .01393 & .01105 & .01189 & .1154 \end{bmatrix} , \qquad (4.6.16)$$

$$E_a(R,M) = \begin{bmatrix} 1.77 & 1.96 & 2.13 & .108 \\ 2.03 & 1.53 & 2.53 & .1069 \\ 1.74 & 1.45 & 1.88 & .1152 \end{bmatrix} . \qquad (4.6.17)$$

In this case, very small queue length error tolerance and absolute error criterion values are observed. These small values are evident from (4.6.16) and (4.6.17).

Stress networks for our algorithm as well as for most other MVA-based algorithms, as was noted in section 4.2, are network with single or multiple bottlenecks. We especially expect the largest errors to occur in networks where some service centers are saturated but the chain populations causing the saturation are small. From the MVA equations in (4.2.6), and knowing that $\rho_{ij}(\mathbf{K}) = \lambda_{ij}(\mathbf{K}) s_{ij}$, it is easy to see that, if $\rho_{ij} = 1$, then we have $\Delta_{ij} = 1$.

As a last example, we consider one such stress network. In section 4.7, we will propose an algorithm that removes this drawback and improves the initial algorithm (algorithm I) drastically. Consider a network with the same visit ratios matrix given by (4.6.11), and assume the same dimensions for M and R as in previous examples, and a population vector of $\mathbf{K} = (5,5,5)$. The mean service time matrix is given by:

$$S(R,M) = \begin{bmatrix} .599 & 1.594 & 2.191 & 1.00 \\ .872 & 2.845 & 4.861 & 2.5 \\ 2.044 & 2.225 & 5.263 & 3.1 \end{bmatrix} . \qquad (4.6.18)$$

Solving the network using our algorithm produces the following error matrices:

$$E_t(R,M) = \begin{bmatrix} 2.46 & 1.65 & 10.82 & .39 \\ 2.61 & 1.94 & 11.44 & .734 \\ 2.48 & 1.64 & 10.41 & .393 \end{bmatrix}, E_a(R,M) = \begin{bmatrix} 14.12 & 13.54 & 16.78 & 6.74 \\ 16.99 & 16.97 & 16.91 & 8.90 \\ 13.48 & 16.25 & 8.90 & 6.78 \end{bmatrix}. \quad (4.6.19)$$

The error matrices using Chow's algorithm (δ_{ij}=0.) are given by:

$$E_t(R,M) = \begin{bmatrix} 2.78 & 1.86 & 5.06 & .44 \\ 2.79 & 2.17 & 5.62 & .768 \\ 2.81 & 1.78 & 5.02 & .430 \end{bmatrix}, E_a(R,M) = \begin{bmatrix} 16.10 & 15.01 & 7.84 & 7.47 \\ 21.99 & 18.11 & 8.31 & 9.32 \\ 15.25 & 17.61 & 9.32 & 7.44 \end{bmatrix}, \quad (4.6.20)$$

Comparing the two algorithm, it is clear that the error due to algorithm I is lower for all entries except at the bottleneck service center (service center 3). The larger error for service center 3 is clearly due to the nature of our approximation that estimates the correction terms for Δ_{ij} values at service center 3 to be much smaller than they really are. A second significant observation is that there is no apparent correlation between the magnitude of the queue length error tolerance and that of the absolute error criterion values. This observation means that small values for queue length error tolerance do not necessarily imply that the corresponding absolute error criteria are also proportionally small. The same conclusion holds true for the for the maximum queue length error tolerance and the maximum absolute error criterion values. In section 4.7, we describe a multilevel algorithm that improves even further the estimates of the performance measures at the bottleneck centers as well as at all other service centers.

As we indicated earlier, the time complexity of our algorithm is very small compared with exact solution techniques. To verify the validity of our claim, we compared the time complexities of the respective approaches to the solution of a network with M=4, and R=3 as a function of the total population K. The results are shown in Figure 4.1. These results show that, as the population increases, the difference between the costs of the exact solution and of algorithm I increases rapidly.

4.7 Possible Extensions

We observed in the previous section that stress networks for our algorithm are networks with single or multiple bottlenecks where the saturation occurs at relatively small values of the chain populations that cause the saturation. Obviously, if the saturation occurs for very large chain populations, the error will be very smaller (for both the absolute error criterion and the queue length error tolerance values). We realized in the last example of section 4.7 that our algorithm introduced larger errors at the bottleneck center than that of Chow [Cho83]. An immediate remedy for this problem is to combine the two algorithms, i.e., to use δ_{ij}=0 for the bottleneck centers and our algorithm for all other service centers. This can be done by keeping track of the per class utilizations ρ_{ij}. Once a service center is saturated with respect to a chain (or chains), both algorithms are invoked, and the two approximations are then employed to evaluate the performance measures. The additional computational cost of solving the network approximately twice is marginal, considering that in most cases the exact solution is impractical. It is important to note that, in multichain networks, as we shall see in chapter 6, there could be multiple bottlenecks for different population vector mixes. However, if a certain ordering (to be discussed in chapter 6) exists among loading factors, then there are cases where there is only one bottleneck in the network. There are asymptotic values for the performance measures of networks with only one bottleneck, but none is known for networks with multiple bottlenecks. A more effective approach is to use a second level of approximation. The improvement in accuracy is in most cases substantial. As was noted in section 4.2, Chow used his first approximation (δ_{ij} = 0) to evaluate an estimate δ_{ij}^* of the parameters δ_{ij} using (4.2.2). He then used the δ_{ij}^* estimates to solve the network for a second time, and concluded that the improvement in accuracy was substantial only if the estimates evaluated using the first approximation were accurate.

To achieve more accurate solutions, we propose a multilevel iterative algorithm with the following steps.

(1) Use the above described combination of algorithm I and Chow's algorithm to solve the given network with populations $\mathbf{K}$ and $\mathbf{K}-e_i$. Evaluate the estimates of $N_{ij}^*(\mathbf{K})$ and $N_{ij}^*(\mathbf{K}-e_i)$ using our algorithm for non-bottleneck service centers and Chow's for the saturated centers.

(2) Use (4.2.3) to evaluate the estimates of the Δ_{ij} parameters. Let us denote the estimates by Δ_{ij}^*. Then:

$$\Delta_{ij}^* = N_{ij}^*(\mathbf{K}) - N_{ij}^*(\mathbf{K}-e_i)\ . \tag{4.7.1}$$

(3) Substitute the Δ_{ij}^* estimates in (4.2.17), and solve the set of equations in (4.2.17) to evaluate the new estimates of the performance measures.

(4) Continue the iterations until a specific stopping rule criterion is satisfied.

This algorithm was applied to the last example of section 4.6, and the error matrices were found to be the following:

$$E_t(R,M) = \begin{bmatrix} 2.12 & 1.41 & 4.65 & .32 \\ 2.24 & 1.82 & 4.55 & .711 \\ 2.10 & 1.52 & 4.68 & .313 \end{bmatrix},\ E_a(R,M) = \begin{bmatrix} 12.15 & 12.21 & 7.21 & 6.20 \\ 15.34 & 15.26 & 8.11 & 8.14 \\ 13.01 & 15.71 & 8.76 & 6.10 \end{bmatrix}\ . \tag{4.7.2}$$

Substantial improvements resulted from a single iteration of the algorithm. This algorithm has to be studied in more detail to investigate its accuracy. Indications are that the algorithm can drastically improve the accuracy over the algorithm I. Although there are additional computational costs involved, the time complexity of algorithm I is very small and does not impose unreasonable computational costs. The execution time for the solution of any of the models using the initial approximation on an IBM 3090/200 was at most .19 seconds.

4.8 Conclusions

In this chapter we have generalized an existing approximate algorithm by proposing a new algorithm that applies to product form queueing networks with fixed rate single server and IS service centers. The algorithm's accuracy is similar to that achieved by Linearizer, but is obtained less expensively in terms of time and space complexities. We have further extended that approximation by proposing a multilevel approximate technique. One significant advantage of algorithm I is that the existence and uniqueness of the network's solution in the feasible region is guaranteed. The solution of the set of nonlinear equations to which algorithm I leads seems to have a very rapid rate of convergence. We then introduced a multilevel algorithm that improves the accuracy of the solution over algorithm I. We have studied the accuracy and the computational cost of the algorithms experimentally. Our initial approximate algorithm's accuracy was examined using two different error criteria. We observed that small values for queue length error tolerance do not necessarily imply that the corresponding absolute error criteria are also proportionally small. Future work include further investigation of the choices of the values for the functions $f_j(R,M,\mathbf{K})$ for different types of networks and at different service centers.

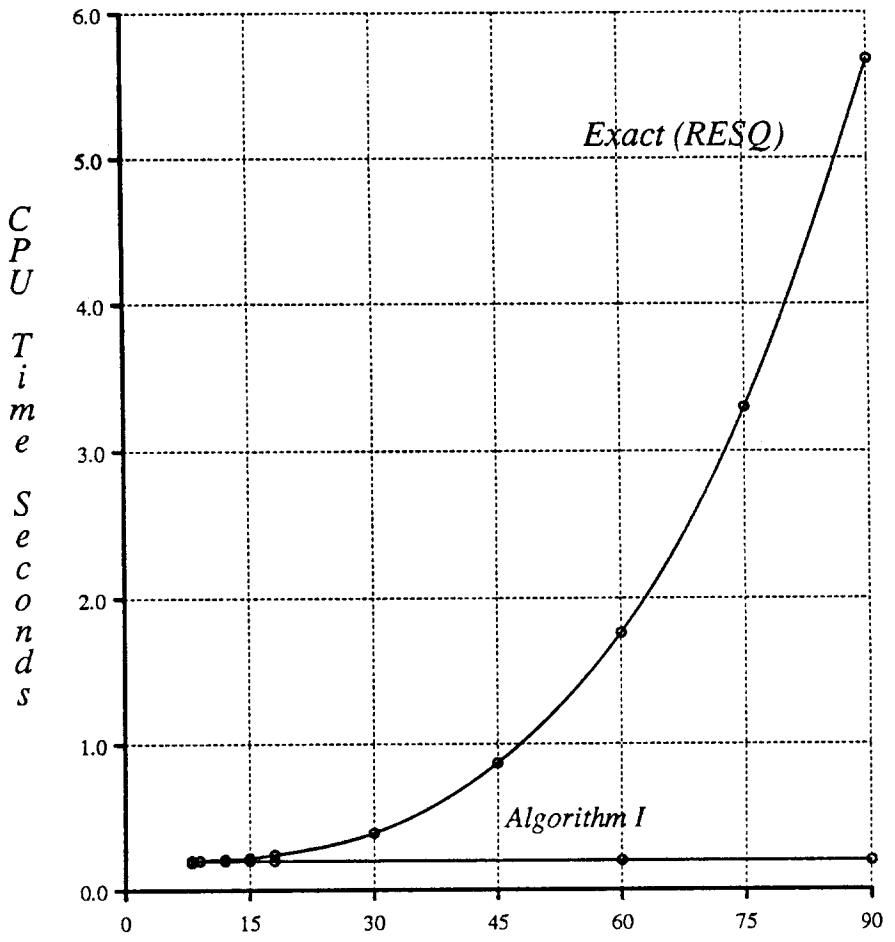
Time Complexity vs. Number of Users
CPU Time Seconds
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0
15
30
45
60
75
90
Exact (RESQ)
Algorithm I
Number of Users (K1+K2+K3)

Figure 4.1

Chapter 5

A Hierarchical Network Transformation Based Approximation for Large Queueing Networks

5.1 Introduction

Hierarchical approximate algorithms are one of the most promising approaches for the approximate solution of large product form queueing networks. The most important advantage of these techniques is the gradual introduction of approximation up to the point where computational costs become acceptable. Most other approximate solution techniques cannot be introduced only partially, hence they do not allow a variable degree of approximation to be used.

In this chapter we propose a hierarchical approximation technique for multiclass separable queueing networks. This technique, which relies on a hierarchy of network transformations, provides us with a smooth tradeoff between cost and accuracy. The key elements of the approach entail transforming in the first step queueing networks containing multiple infinite servers into ones containing a single infinite server model[1]. In the next stage, at least some of the closed chains are transformed into open chains, resulting in a mixed network; this is done on the basis of the desired accuracy and computational cost. If necessary, a completely open network may be obtained. Furthermore, upper and lower bounds of the performance measures can be computed. These bounds are asymptotically correct. Numerical results are presented at the end of this chapter which compare this method with those yielding exact values and with other approximate algorithms. Additionally some of the chains need not visist any of the IS service centers.

Our approach gracefully introduces approximations commensurate with the increasing complexity of the model. A typical example is shown in Figure 5.1, where the nodes of a distributed system are connected by an internetwork. Each of the nodes may in turn consist of several sites connected by a local area network. The typical node and site configurations are shown in Figure 5.2. Assuming that the system can be modeled by a closed multiclass product form network, we then proceed to transform the queueing network (QN) into smaller mixed networks by decoupling the individual building blocks

1 This requirement, as we shall discuss, can further be relaxed to networks with fixed rate single server service centers and multiple IS service centers with *at least* one IS visited by the chains that are opened.

from the whole network.

We start with a basic description of the structure and type of queueing network models that our methodology can be applied to. We consider closed multichain product form queueing networks consisting of only fixed rate single server and infinite server service centers. The model is assumed to contain at least one infinite server service center, and each chain to visit at least one of the infinite servers. As it was pointed out, some of the chains may visit none of the IS service centers in the network. The approximation, however, does not apply to these chains. When modeling distributed systems, each infinite server typically represents a set of terminals. This technique can be extended to include models where some of the chains do not visit any of the IS service centers. Note that the assumption that the original network is closed makes our treatment easier, but is by no means essential. In general, the original QN may be a mixed network.

Our methodology consists of the following steps:

(1) The original model *N1* is transformed into an equivalent network *N2* containing a single infinite server service center. An outline of this transformation is presented in section 5.2. *N1* and *N2* are equivalent in the sense that the equilibrium state probability distribution of *N1* can be obtained from that of *N2*.

(2) Some of the closed chains of the network *N2* at the infinite service center (visited by all chains) are replaced by a hierarchy of constant rate Poisson sources whose rates are dependent on the parameters of the remainder of the QN in a way to be explicitly defined later. The resulting mixed product form QN is referred to as *N3*. This transformation is discussed in sections 5.3 - 5.7. The number of chains to be opened is determined by the desirable accuracy and computational costs of solving the mixed network.

(3) *N3* is then solved using exact solution algorithms.

The choice of bound hierarchies and the number of iterations for the open chains depend on the desired accuracy and the acceptable computational complexity (both space and time) costs. Stopping rules and error criteria are discussed in section 5.8. Computational complexities and experimental results are presented in sections 5.9 and 5.10, respectively.

Without any loss of generality, throughout this study it is assumed that the product form networks we are studying are all of the nonclass-hopping type. It has been shown by Reiser and Kobayashi [Rei75] that any class-hopping network model can be mapped into a multichain model with non class-hopping and with the same steady state solution.

5.2 Transformation into a Network with One Infinite Server

Theorem 5.2.1:

> Consider a multichain product form queueing network consisting of only fixed rate single server and infinite server service centers so that every chain visits at least one infinite server. Then an equivalent queueing network can be constructed with only one aggregate infinite server such that all chains visit this infinite server.

The newly constructed QN is said to be equivalent to the original network if the equilibrium state probability distribution of the original network can be obtained from the steady state distribution of the constructed QN.

A brief outline of the proof is given here. There are two cases to consider:

(1) When each of the chains in the original network visits only one of the infinite servers.

(2) When some or all the chains visit more than one infinite service center in the original network.

Case 1:

Let service centers 1,2,...,I be the IS service centers in the original network QN1, and $R(i)$ be the set of chains visiting the infinite server i. Construct a network QN2 by aggregating all the infinite servers into an infinite server T such that $R(T) = R(1) \cup R(2) \cup \cdots \cup R(I)$. Note that the sets $R(1), R(2), \ldots, R(I)$ are mutually exclusive sets, i.e., $R(i) \cap R(j) = \emptyset$ for all i≠j. It is a matter of algebraic manipulation to establish that the steady state probability distribution of the original network QN1 can be derived from that of the constructed network QN2.

Case 2:

This is the case where the sets $R(1), R(2), \ldots, R(I)$ are no longer mutually exclusive. The task of transformation in this case is performed in two steps. In the first step, we introduce a new class for the chains visiting each additional infinite server. Therefore, if a chain visits m infinite server service centers, we add $m-1$ additional new classes at the aggregate infinite server. The probability of reaching each of the new classes at the aggregate infinite server is the same as the probability of reaching each of the infinite servers in the original network. The result is a class-hopping product form network with a single infinite server.

Once again, that the steady state probability distribution of the original network can be obtained from that of the constructed one can be proved by directly manipulating the distribution of the multiple infinite server QN and arriving at the solution for the QN with only one infinite server service center. Although this step is not necessary, as we mentioned earlier, the new equivalent class-hopping single infinite server QN can further be transformed into a non class-hopping QN using the Reiser and Kobayashi algorithm [Rei75]. Henceforth, without any loss of generality, we assume that we have a product form queueing network consisting of only fixed rate single server and IS service centers with at least one infinite server visited by all the chains. This assumption is a more relaxed version of the results of theorem 5.2.1. since it implies that the network can have multiple IS service centers. It is only required that at least one IS service centers be visited by all chains.

□

5.3 Closed-to-Mixed Network Transformation

In this section we present an algorithm that generates a hierarchical approximation for multichain product form QNs consisting of only fixed rate single server and IS service centers with at least one infinite server that is visited by all the chains. Starting with a closed network, we replace some closed chains with constant independent external Poisson sources. Clearly, as each closed chain is replaced by an open chain, the computational complexity of the solution of the resulting mixed network decreases. At the same time, each replacement of a closed chain introduces an increase in the error affecting the performance measures computed from the resulting mixed network. This process of transformation can continue until an acceptable balance between error and complexity is achieved. In practice, one major limitation is the feasibility of solving the mixed network. This is because, as we discussed, the complexity of solving mixed networks is dominated by the complexity of the closed chains. By considering the allowable space and time complexities, one can keep as many closed chains as is possible to minimize the error. The error is maximum when all the closed chains are transformed into open chains. This limiting case has been studied in different ways. Pittel [Pit79] investigated the asymptotic behavior of multichain separable QNs with capacity constraints. Another approach that investigates the asymptotic properties of the same kind of queueing network has been proposed by Lavenberg [Lav80]. Assume that the throughput rates of class r jobs at the IS service center M visited by all the chains are denoted by λ_{rM} , $r=1,2,\ldots,R$. Furthermore, suppose that $\lim_{K\to\infty} K_r \mu_{rM}$ exists and is finite. Here, K_r is the

population of chain r jobs and $\sum_{i=1}^{R} K_i = K$. Lavenberg showed that, as the population of the network increases, the equilibrium state probability distribution of the network approaches that of an open network with Poisson arrival rates given by λ_{rM}. For more details about this approach see section 5.7.

A closely related technique was proposed by Whitt [Whi84]. He suggested a transformation of the same type of queueing network into an open network, and devised an algorithm for finding the arrival rates of the open network by a hierarchy of upper and lower bounds using an iterative algorithm. We will extend Whitt's algorithm to the case of mixed networks in in this chapter. Lavenberg's approximation will become the asymptotic limit in our algorithm.

5.3.1 Open Chains Arrival Rates Estimation

Assume that the original closed network has been transformed into an equivalent closed network containing at least an infinite server service center, T_1, visited by all the chains. Suppose that there are M other service centers (besides T_1). The remaining M service centers are fixed rate single server or IS service centers. We assume that we have ordered the set R of the chains into two groups. The chains in the first set, O, are to be transformed into open chains. The chains in the second set, C, are to remain closed.

Assume that we can obtain an equivalent mixed network such that the equilibrium state probability of the original network can be derived from that of the constructed mixed network. The equivalent network has R^o open chains and R^c closed chains replacing the original R closed chains. Suppose the constant arrival vector for the open chains is denoted by $\Lambda_o = (\lambda_{o,1}, \lambda_{o,2}, \ldots, \lambda_{o,R^o})$. Let $\mu_{o,r}$ be the service rate of chain r jobs ($r \varepsilon O$) at the infinite server service center T_1. Little's law should apply to both the original and the equivalent networks. In addition, the arrival and departure rates at the IS service center T_1 should be equal (under asymptotic conditions we discuss in section 5.7). Hence, applying these two laws to the mixed network, and using equations (2.4.8) and (2.4.20), we can write:

$$\lambda_{o,r} s_{o,r} = K_r - \sum_{s \varepsilon S_*(r)} \frac{\rho^o_{rs}(\Lambda_o)(1 + \sum_{j \varepsilon R_*(s)} N^c_{js}(\Lambda_o, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_*(s)} \rho^o_{is}(\Lambda_o))}, \qquad r \varepsilon O, \tag{5.3.1}$$

and, substituting for $N^c_{js}(\Lambda_o, \mathbf{K}_c)$ from equation (2.4.20), we get:

$$\lambda_{o,r} s_{o,r} = K_r - \sum_{s \varepsilon S_*(r)} \frac{\rho^o_{rs}(\Lambda_o)((1 - \rho^o_s(\Lambda_o)) + \sum_{j \varepsilon R_*(s)} \lambda^c_{js} s^c_{js}(1 + \sum_{i \varepsilon R_*(s)} N^c_{is}(\Lambda_o, \mathbf{K}_c - e_j)))}{(1 - \sum_{i \varepsilon R_*(s)} \rho^o_{is}(\Lambda_o))^2},$$

$$r \varepsilon O. \tag{5.3.2}$$

Observing that $s_{o,r} = (\mu_{o,r})^{-1}$ and simplifying equation (5.3.2), we obtain:

$$\lambda_{o,r} = K_r \mu_{o,r} - \sum_{s \varepsilon S_*(r)} \frac{\mu_{o,r} \rho^o_{rs}(\Lambda_o)}{(1 - \sum_{i \varepsilon R_*(s)} \rho^o_{is}(\Lambda_o))} + \sum_{s \varepsilon S_*(r)} \sum_{j \varepsilon R_*(s)} \frac{\lambda^c_{js} s^c_{js} \mu_{o,r}}{(1 - \sum_{i \varepsilon R_*(s)} \rho^o_{is}(\Lambda_o))^2} (1 + \sum_{i \varepsilon R_*(s)} N^c_{is}(\Lambda_o, \mathbf{K}_c - e_j)),$$

$$r \varepsilon O. \tag{5.3.3}$$

Equations (5.3.1) and (5.3.2) are also valid for the closed network under certain conditions. In other words, the arrival rates that are obtained by solving the set of nonlinear equations in (5.3.1) will also apply to the original closed network (where the arrival rates become the throughput rates at the IS service center visited by all chains) if certain asymptotic conditions (section 5.7) are satisfied. A

discussion of the asymptotic properties of equations (5.3.1) - (5.3.3) will be presented in section 5.7. We recall that the constraint of stability for the mixed network should always be satisfied. This, as we discussed earlier, requires that $\rho_s^2<1$ for *all* s.

5.4 Basic Definitions and Theorems

In this section we review some basic definitions and theorems relating to the contraction mapping which we will use to solve the set of equations represented by (5.3.1) - (5.3.3) (see Ortega [Ort70]).

Definition 5.4.1:

A map $\mathbf{F}: D_1 \subset R^n \rightarrow R^n$ is said to be *nonexpansive* on a set $D_o \subset D_1$ if:

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\| \leq \|\mathbf{x} - \mathbf{y}\|, \quad \text{for } \textit{all } \mathbf{x},\mathbf{y} \in D_o. \tag{5.4.1}$$

The map is said to be *strictly nonexpansive* on D_o if strict inequality holds in (5.4.1) whenever $\mathbf{x} \neq \mathbf{y}$. Any nonexpansive map on D_o is Lipschitz continuous on D_o. Any solution $\mathbf{x}^{(*)}$ in the domain of $\mathbf{F}$ that satisfies the relationship $\mathbf{x}^{(*)} = \mathbf{F}(\mathbf{x}^{(*)})$ is called a fixed point of map $\mathbf{F}$. It is easy to show that strictly nonexpansive maps have at most one fixed point.

Definition 5.4.2:

A map $\mathbf{F}: D_1 \subset R^n \rightarrow R^n$ is a *contraction mapping* on a set $D_o \subset D$ if there is a $\gamma \in (0,1)$ such that $\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\| \leq \gamma \|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x},\mathbf{y} \in D_o$. Obviously, any contraction mapping is strictly nonexpansive.

Theorem 5.4.1 (The Contraction-Mapping Theorem):

Let $\mathbf{F}: D_1 \subset R^n \rightarrow R^n$ be a contraction mapping on a closed set $D_o \subset D_1$, and assume that $\mathbf{F}(D_o) \subset D_1$. Then $\mathbf{F}$ has a unique fixed point.

Corollary 5.4.1:

If we form the sequence $\mathbf{x}^{(I+1)} = \mathbf{F}(\mathbf{x}^{(I)})$, $I=1,2,....$, then the sequence $\{\mathbf{x}(I),\ I=1,2,...\}$ is a Cauchy sequence and has a limit $\mathbf{x}^{(*)}$ in D_o. In addition, by the continuity of $\mathbf{F}$, we can conclude that $\lim_{I \rightarrow \infty} \mathbf{F}(\mathbf{x}^{(I)}) = \mathbf{F}(\mathbf{x}^{(*)})$. Hence, $\mathbf{x}^{(*)}$ is a fixed point of $\mathbf{F}$ (see Ortega [Ort70]).

Theorem 5.4.2 (The Monotone Bounded Sequence Theorem):

Assume that $D_o \subset D_1$ is a convex set. Let $\mathbf{F}: D_1 \subset R^n \rightarrow R^n$ be a nonexpansive map on D_o. Then $\mathbf{F}$ has a fixed point in D_o if and only if the sequence $\mathbf{x}^{(I+1)} = \mathbf{F}(\mathbf{x}^{(I)})$, $I = 1,2,....$, is bounded for at least one $\mathbf{x}^{(i)} \in D_o$ for some $i \in I$.

For a complete proof of this and the other theorems see [Ort70].

5.5 Approximate Arrival Rates Estimation

We now define two sets of iterative equations similar to those in [Whi84] to be solved for the approximate arrival rates of the chains at the infinite server visited by all the chains.

Let us denote by $\overline{\Lambda}_o^{(I)} = (\overline{\lambda}_{o,1}^{(I)}, \overline{\lambda}_{o,2}^{(I)}, \ldots, \overline{\lambda}_{o,R_o}^{(I)})$ the I^{th} upper-bound recursive estimate of the arrival rate vector of the open chain, and by $\underline{\Lambda}_o^{(I)} = (\underline{\lambda}_{o,1}^{(I)}, \underline{\lambda}_{o,2}^{(I)}, \ldots, \underline{\lambda}_{o,R_o}^{(I)})$ the I^{th} lower-bound recursive estimate of the arrival rate vector of the open chains.

We estimate the upper-bound mean arrival rates for the mixed network obtained by removing R^o of the chains at the infinite server and replacing them with the independent external Poisson sources defined by the following recursive relationship:

$$\overline{\lambda}_{o,r}^{(I+1)} = K_r \mu_{o,r} - \sum_{s \in S_o(r)} \frac{\mu_{o,r} \rho_{rs}^o(\underline{\Lambda}_o^{(I)})(1 + \sum_{j \in R_c(s)} N_{js}^c(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o(\underline{\Lambda}_o^{(I)}))}, \quad r \varepsilon O, \quad I=1,2,3,\ldots, \tag{5.5.1}$$

while the lower-bound estimates for the mean arrival rates are given by:

$$\underline{\lambda}_{o,r}^{(I)} = K_r \mu_{o,r} - \sum_{s \in S_o(r)} \frac{\mu_{o,r} \rho_{rs}^o(\overline{\Lambda}_o^{(I)})(1 + \sum_{j \in R_c(s)} N_{js}^c(\overline{\Lambda}_o^{(I)}, \mathbf{K}_c))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o(\Lambda_o^{(I)}))}, \quad r \varepsilon O, \quad I=1,2,3,\ldots. \tag{5.5.2}$$

Using equation (2.4.20) for the mean number of closed chain jobs in the mixed network and substituting for $N_{rs}^c(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c)$, we get the following contraction mapping for the upper bounds:

$$\overline{\lambda}_{o,r}^{(I+1)} = K_r \mu_{o,r} - \sum_{s \in S_o(r)} \frac{\mu_{o,r} \rho_{rs}^o(\underline{\Lambda}_o^{(I)})((1 - \rho_s^o(\underline{\Lambda}_o^{(I)})) + \sum_{j \in R_c(s)} \lambda_{js}^c s_{js}^c \mu_{o,r}(1 + \sum_{i \in R_c(s)} N_{is}^c(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c - e_j)))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o(\underline{\Lambda}_o^{(I)}))^2},$$

$$r \varepsilon O, \quad I=1,2,3,\ldots, \tag{5.5.3}$$

and the equation given below provides the lower-bound estimates:

$$\underline{\lambda}_{o,r}^{(I)} = K_r \mu_{o,r} - \sum_{s \in S_o(r)} \frac{\mu_{o,r} \rho_{rs}^o(\overline{\Lambda}_o^{(I)})(1 - \rho_s^o(\overline{\Lambda}_o^{(I)})) + \sum_{j \in R_c(s)} \lambda_{js}^c s_{js}^c \mu_{o,r}(1 + \sum_{i \in R_c(s)} N_{is}^c(\overline{\Lambda}_o^{(I)}, \mathbf{K}_c - e_j))}{(1 - \sum_{i \in R_o(s)} \rho_{is}^o(\overline{\Lambda}_o^{(I)}))^2},$$

$$r \varepsilon O, \quad I=1,2,3,\ldots. \tag{5.5.4}$$

Finally, equations (4.3.3) and (4.3.4) can be reduced to:

$$\overline{\lambda}_{o,r}^{(I+1)} = \sum_{s \in S_o(r)} \frac{(K_r(1 - \rho_s^o(\underline{\Lambda}_o^{(I)})) - \rho_{rs}^o(\underline{\Lambda}_o^{(I)}))\mu_{o,r}}{(1 - \rho_s^o(\underline{\Lambda}_o^{(I)}))} + \sum_{\substack{s \in S_o(r) \\ j \in R_c(s)}}^{M} \frac{\lambda_{js}^c s_{js}^c \mu_{o,r}}{(1 - \rho_s^o(\underline{\Lambda}_o^{(I)}))^2}(1 + \sum_{i \in R_c(s)} N_{is}^c(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c - e_j))$$

$$r \varepsilon O, \quad I=1,2,3,\ldots, \tag{5.5.5}$$

and:

$$\underline{\lambda}_{o,r}^{(I)} = \sum_{s \in S_o(r)} \frac{(K_r(1 - \rho_s^o(\overline{\Lambda}_o^{(I)})) - \rho_{rs}^o(\overline{\Lambda}_o^{(I)}))\mu_{o,r}}{(1 - \rho_s^o(\Lambda_o^{(I)}))} + \sum_{\substack{s \in S_o(r) \\ j \in R_c(s)}} \frac{\lambda_{js}^c s_{js}^c \mu_{o,r}}{(1 - \rho_s^o(\Lambda_o^{(I)}))^2}(1 + \sum_{i \in R_c(s)} N_{is}^c(\overline{\Lambda}_o^{(I)}, \overline{\mathbf{K}}_c - e_j))$$

$$r \varepsilon O\ ,\quad I=1,2,3,.... \tag{5.5.6}$$

Though equations (5.5.3) and (5.5.4) can be used to solve for the approximate mean throughput rate, equations (5.5.5) and (5.5.6) suggest alternative ways of doing so to reduce the round off and other computational errors. In the special case where there are no closed chains remaining and the closed network is transformed into an open network, the equations in (5.5.5) and (5.5.6) are reduced to those in [Whi84]. The last two equations also show their sensitivity to ρ_s^o.

5.5.1 Initial Conditions for Open Chains Arrival Rates

The initial conditions for the iterative relationships given by equations (5.5.5) and (5.5.6) is given by:

$$\bar{\lambda}_o^{(1)} = \alpha K_r \mu_{o,r}\ ,\quad r \varepsilon O\ ,\quad \alpha \varepsilon\ (0,1]\ , \tag{5.5.7}$$

where α is to be chosen to satisfy:

$$K_r > \sum_{s=1}^{M} \frac{\rho_{rs}^o(\bar{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon \bar{K}_*(s)} N_{js}^c(\bar{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon \bar{K}_*(s)} \rho_{is}^o(\bar{\Lambda}_o^{(1)}))}\ ,\quad r \varepsilon O\ . \tag{5.5.8}$$

In most cases the choice $\alpha = 1$ is sufficient. However, there are cases where $\alpha = 1$ does not satisfy (5.5.8), and a different value of it must be chosen. The initial conditions in (5.5.7) is the same as those derived in (4.5.1).

5.6 Convergence of the Iterative Algorithm

In this section, we prove that the solutions of the iterative mappings given by equations (5.5.1) and (5.5.2) for the upper and lower bounds of the arrival rates converge to a single fixed point.

Theorem 5.6.1

> Consider a mixed multichain separable queueing network consisting of only fixed rate single server and IS service centers. Then, the mean queue lengths given by the relationship in equation (2.4.8) are strictly increasing functions of the open chain external arrival rates.

Proof:

Combining equations (2.4.1) and (2.4.5), we obtain the following equation:

$$\lambda_{rs}^o = \theta_{rs}^o \sum_{i \varepsilon O} \lambda_{o,i}\ . \tag{5.6.1}$$

If chain r does not visit service center s, then $(\theta_{rs}^o = 0) \rightarrow (\lambda_{rs}^o = 0)$. We also know that:

$$\rho_{rs}^o = \lambda_{rs}^o s_{rs}^o = s_{rs}^o \theta_{rs}^o \sum_{i \varepsilon O} \lambda_{o,i}\ . \tag{5.6.2}$$

Therefore the following relationship also holds:

$$\sum_{i \varepsilon \bar{K}_*(s)} \rho_{is}^o = \sum_{i \varepsilon \bar{K}_*(s)} \sum_{j \varepsilon O} \theta_{is}^o s_{is}^o \lambda_{o,j}\ . \tag{5.6.3}$$

Substituting for the ρ^o_{rs}'s in equation (2.4.8), and using equation (5.6.3), we obtain:

$$N^o_{rs}(\Lambda_o,\mathbf{K}_c)=\frac{\theta^o_{rs}s^o_{rs}\sum_{i\varepsilon O}\lambda_{o,i}(1+\sum_{j\varepsilon R^c_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c))}{(1-\sum_{i\varepsilon R^o_*(s)}\sum_{j\varepsilon O}\theta^o_{is}s^o_{is}\lambda_{o,j})},\qquad r\varepsilon R_o(s),\qquad s=1,2,...,M. \tag{5.6.4}$$

Taking the partial derivative of the function given in equation (5.6.4), we have:

$$\frac{\partial N^o_{rs}(\Lambda_o,\mathbf{K}_c)}{\partial\lambda_{o,r}}=\frac{\theta^o_{rs}s^o_{rs}}{(1-\sum_{i\varepsilon R^o_*(s),j\varepsilon O}\theta^o_{is}s^o_{is}\lambda_{o,j})^2}(1+\sum_{j\varepsilon R^c_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c))+$$

$$\frac{\rho^o_{rs}}{1-\rho^o_s}\frac{\partial}{\partial\lambda_{o,r}}\left[\sum_{j\varepsilon R^c_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c)\right]. \tag{5.6.5}$$

Using equation (2.4.20) we can write:

$$\frac{\partial}{\partial\lambda_{o,r}}\left[\sum_{j\varepsilon R^c_*(s)}N^c_{js}(\Lambda_o,\mathbf{K}_c)\right]=\sum_{j\varepsilon R^c_*(s)}\frac{\partial}{\partial\lambda_{o,r}}\frac{\lambda^c_{js}s^c_{js}(1+\sum_{k\varepsilon R^c_*(s)}N^c_{ks}(\Lambda_o,\mathbf{K}_c-e_j))}{(1-\sum_{i\varepsilon R^o_*(s)}\rho^o_{is})},\qquad r\varepsilon O. \tag{5.6.6}$$

Equation (5.6.6) is a recursive relationship. Under the assumption of a stable mixed network, the first term in equation (5.6.5) is always strictly positive (except for the trivial case where there are no open chains and as a result that term reduces to zero). These derivatives involve terms of the form:

$$\frac{\partial}{\partial\lambda_{o,r}}\left(\frac{1}{(1-\sum_{\substack{i\varepsilon R^o_*(s)\\ j\varepsilon O}}\theta^o_{is}s^o_{is}\lambda_{o,j})^m}\right)=\frac{m\,\theta^o_{rs}s^o_{rs}}{(1-\sum_{\substack{i\varepsilon R^o_*(s)\\ j\varepsilon O}}\theta^o_{is}s^o_{is}\lambda_{o,j})^{m+1}},\quad \text{for } all\ m\varepsilon Z^+. \tag{5.6.7}$$

Observing that for a stable mixed network these derivatives are always positive for all positive integers m, we conclude that $N^o_{rs}(\Lambda_o,\mathbf{K}_c)$ is a strictly increasing and continuous function of the open chain external arrival rates. For the IS service centers, the proof that the mean queue lengths are strictly increasing function of the open chain external arrival rates, is trivial. This is due to the fact that for IS service centers, $N^o_{rs}=\theta^o_{rs}s_{rs}\sum_{i\varepsilon O}\lambda_{o,i}$.

□

Theorem 5.6.2

Consider a closed multiclass product form queueing network consisting of only[2] fixed rate single servers. Suppose an iterative relationship defined by (5.5.1) and a corresponding initial condition denoted by (5.5.7) and (5.5.8) is defined on this network. Then, the sequence given by $\left\{\bar{\lambda}^{(I)}_{o,r}:\ r\varepsilon O,\ I=1,2,...\right\}$ is a strictly decreasing sequence.

Proof:

We prove the theorem for α=1. The extension of the proof to the case where $\alpha\neq1$ is trivial. For I=1, equation (5.5.2) can be written in the following form:

$$\underline{\lambda}^{(1)}_{o,r}=\bar{\lambda}^{(1)}_{o,r}-\sum_{s\varepsilon S_*(r)}\frac{\mu_{o,r}\rho^o_{rs}(\bar{\Lambda}^{(1)}_o)(1+\sum_{j\varepsilon R^c_*(s)}N^c_{js}(\bar{\Lambda}^{(1)}_o,\mathbf{K}_c))}{(1-\sum_{i\varepsilon R^o_*(s)}\rho^o_{is}(\Lambda^{(1)}_o))},\qquad r\varepsilon O. \tag{5.6.8}$$

2 This assumption is to simplify the proof, otherwise, the network may also contain multiple IS service centers.

Since the second term on the right hand side is always strictly positive, the following inequality holds:

$$\underline{\lambda}_{o,r}^{(1)} < \bar{\lambda}_{o,r}^{(1)}, \quad \text{for } \textit{all } r \varepsilon O \ . \tag{5.6.9}$$

From equation (5.5.1) and for $I=1$, it follows that:

$$\bar{\lambda}_{o,r}^{(2)} = \bar{\lambda}_{o,r}^{(1)} - \sum_{s \varepsilon \bar{S}_o(r)} \frac{\mu_{o,r} \rho_{rs}^{o}(\underline{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon \bar{R}_o(s)} N_{js}^{c}(\underline{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon \bar{R}_o(s)} \rho_{is}^{o}(\underline{\Lambda}_o^{(1)}))} \ . \quad r \varepsilon O, \tag{5.6.10}$$

Observing that the second term on the right hand side of equation (5.6.10) is positive, we have:

$$\bar{\lambda}_{o,r}^{(2)} < \bar{\lambda}_{o,r}^{(1)}, \quad \text{for } \textit{all } r \varepsilon O \ . \tag{5.6.11}$$

We can also write:

$$\underline{\lambda}_{o,r}^{(1)} - \underline{\lambda}_{o,r}^{(2)} =$$

$$\sum_{s \varepsilon \bar{S}_o(r)} \frac{\mu_{o,r} \rho_{rs}^{o}(\bar{\Lambda}_o^{(2)})(1 + \sum_{j \varepsilon \bar{R}_o(s)} N_{js}^{c}(\bar{\Lambda}_o^{(2)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon \bar{R}_o(s)} \rho_{is}^{o}(\Lambda_o^{(2)}))} - \sum_{s \varepsilon \bar{S}_o(r)} \frac{\mu_{o,r} \rho_{rs}^{o}(\bar{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon \bar{R}_o(s)} N_{js}^{c}(\bar{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon \bar{R}_o(s)} \rho_{is}^{o}(\Lambda_o^{(1)}))} \ . \tag{5.6.12}$$

By (5.6.11) we know that $\bar{\lambda}_{o,r}^{(2)} < \bar{\lambda}_{o,r}^{(1)}$ for all $r \varepsilon O$. The two terms on the right hand side of equation (5.6.12), by theorem 5.6.1, are increasing functions of external arrival rates. Thus, we conclude that $\underline{\lambda}_{o,r}^{(2)} < \underline{\lambda}_{o,r}^{(1)}$ for all $r \varepsilon O$. From this result we can similarly deduce that:

$$\bar{\lambda}_{o,r}^{(3)} < \bar{\lambda}_{o,r}^{(2)}, \quad \text{for } \textit{all } r \varepsilon O \ .$$

By a similar argument, we can conclude that:

$$\bar{\lambda}_{o,r}^{(I)} < \bar{\lambda}_{o,r}^{(I-1)} < \cdots < \bar{\lambda}_{o,r}^{(2)} < \bar{\lambda}_{o,r}^{(1)} \quad \text{for } \textit{all } r \varepsilon O \ . \tag{5.6.13}$$

Observe that, by the initial condition assumption assumption we also have:

$$\bar{\lambda}_{o,r}^{(1)} = \alpha K_r \mu_{o,r}, \ r \varepsilon O, \text{ and for } \textit{some } \alpha \varepsilon (0,1] \ .$$

□

Corollary 5.6.1

The following relation holds for the lower bound sequence:

$$\bar{\lambda}_{o,r}^{(I+1)} > \underline{\lambda}_{o,r}^{(I+1)} > \underline{\lambda}_{o,r}^{(I)} > \cdots > \underline{\lambda}_{o,r}^{(1)} \quad \text{for } \textit{all } r \varepsilon O. \tag{5.6.14}$$

The proof of the corollary follows directly from the proof of theorem 5.6.2.

□

The iterative algorithm, by the contraction mapping theorem and the monotone bounded sequence theorem given in section 5.4, converges. The sequence of lower and upper bound estimates as computed by equations (5.5.1) and (5.5.2) always converges to a unique fixed point as the number of iterations approaches infinity.

$$\lim_{I \to \infty} \bar{\lambda}_{o,r}^{(I)} = \lim_{I \to \infty} \underline{\lambda}_{o,r}^{(I)} = \lambda_r', \quad r \varepsilon O \ . \tag{5.6.15}$$

Hence, the solution of the set of nonlinear equations (5.3.1) and (5.3.2) are asymptotically correct (that is, they are exact in the limit as K becomes large).

5.7 Asymptotic Properties

Assume that N is a closed multichain product form network that consists of $M+1$ fixed rate single server service centers and IS service centers. Suppose that the $(M+1)^{th}$ service center is the $(M+1)^{th}$ center visited by all the chains. Then let:

$$K_r = \left\lfloor p_r K \right\rfloor \quad , \quad r \varepsilon R(M+1), \tag{5.7.1}$$

where $K \varepsilon Z^+$, and the p_i's are chosen so that:

$$\sum_{i \varepsilon O} p_i = 1. \tag{5.7.2}$$

Furthermore, assume that:

$$\lim_{K \to \infty} K_r \mu_{o,r} \quad \textit{exists} \text{ and } \textit{is finite} \text{ for } \textit{all } r \varepsilon R(M+1). \tag{5.7.3}$$

Suppose, using the notation in equations (2.1.2) and (2.1.3), that $N1$ is the subnetwork of the original closed network N obtained by excluding the $(M+1)^{th}$ service center from the state vector of the original network. The state of subnetwork $N1$ is denoted by $\mathbf{k}^* = (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M)$. Let us also define:

$R(s) =$ set of chains visiting server s;

$m_r = \sum_{i=1}^{M} k_{ri} =$ population of chain r jobs in subnetwork $N1$;

$S^*(\mathbf{K}) = \left\{ \mathbf{k}^* : m_r \le K_r, \; r \varepsilon C \right\} =$ set of states of subnetwork $N1$;

$p_1(\mathbf{k}^*, \mathbf{K})$: equilibrium state probability distribution of subnetwork $N1$.

Furthermore let:

$$\lambda_r = K_r \mu_{rM+1} \, , \quad r \varepsilon R(M+1) \, . \tag{5.7.4}$$

and

$$\rho_{r,s} = s_{rs} \theta_{rs} \sum_{i \varepsilon R(M+1)} \lambda_i \, , \quad r \varepsilon R(s) \, , \quad s = 1,2,\ldots,M \, , \tag{5.7.5}$$

where $\rho_s = \sum_{i \varepsilon R(s)} \rho_{is}$ represents the utilization of server s. Suppose the following holds:

$$\rho_s < 1 \text{ for } \textit{all } s \neq M+1 \, . \tag{5.7.6}$$

Then it can be shown that [Lav80]:

$$\lim_{K \to \infty} p_1(\mathbf{k}^*, \mathbf{K}) = \prod_{s=1}^{M} \frac{k_s!}{(1 - \sum_{i \varepsilon R(s)} \rho_{is})} \prod_{j \varepsilon R(s)} \frac{(\rho_{js})^{k_{jr}}}{k_{js}!} \, . \tag{5.7.7}$$

Moreover, for each $\rho_s < 1$, $s \neq M+1$, we have:

$$\lim_{K \to \infty} \lambda_{r,s} = \theta_{rs} \sum_{i \varepsilon R(M+1)} \lambda_i \, , \quad r \varepsilon R(s) \, , \quad s = 1,2,\ldots,M \, ; \tag{5.7.8}$$

if, for some $s \neq M+1$, it is $\rho_s \ge 1$, then for all $\mathbf{k}^* \ge \mathbf{O}$ ($\mathbf{O}$ is the zero vector):

$$\lim_{K\to\infty} p_1(\mathbf{k}^*,\mathbf{K}) = \mathbf{O}\,. \tag{5.7.9}$$

Thus, for the closed multichain network defined above, if $\rho_s<1$ for *all* $s\neq 1,2,...,M$, the steady state probability distribution of closed subnetwork *N1* converges to that of an open network *N2* consisting of service centers 1,2,...,*M* and obtained from N by replacing the IS service center visited by all chains with constant external Poisson arrival rates as defined by (5.7.4).

Furthermore, let $\mathbf{X}(\mathbf{K})$ be the random vector with probability distribution given by $\{p_1(\mathbf{k}^*,\mathbf{K}) : \mathbf{k}^* \in S^*(\mathbf{K})\}$; i.e., $\mathbf{X}(\mathbf{K})$ is the equilibrium state vector of subnetwork *N1*. Suppose the random vector $\mathbf{X}^o$ represents the steady state vector for the stable open subnetwork *N2*. Performance metrics such as queue length distributions, moments of queue size, and utilizations can be defined as functions of the state vectors for networks *N1* and *N2*. A stronger asymptotic relation exists between the two queueing subnetworks. Using the Lebesgue Monotone Bounded Convergence Theorem [Bil86], it can be shown that for any continuous function f:

$$\lim_{K\to\infty} E\left[f(\mathbf{X}(\mathbf{K}))\right] = E\left[f(\mathbf{X}^o)\right]\,. \tag{5.7.10}$$

This implies that the mean and higher moments of all the performance metrics of the closed subnetwork *N1* converge, as $K\to\infty$, to those of the open network defined by *N2*. The results discussed here can be generalized for the mixed networks. The result in (5.7.10) can further be extended to include networks containing service centers with load dependent service rates. For a more detailed discussion of the topics in this section see [Lav80]. All these conclusions hold true as long as the open network remains stable. If $\rho_s>1$ for any of the servers in subnetwork *N1*, then the open subnetwork becomes unstable as *K* increases.

This result implies that the contraction mappings defined for the upper and lower bound arrival rates converge to the exact solution under the conditions discussed in this section. Therefore, $\lim_{K\to\infty} \lambda_r' = \lambda_r$ for *all* $r \in R(M+1)$.

5.8 Stopping Rules and Error Criteria

We introduce several error criteria for the analysis of our approximate algorithm. The accuracy and the cost of any iterative algorithm are greatly affected by the choice of the stopping criterion used. Two error criteria will be defined as the bases of stopping rules for the iterative contraction mappings. We also define upper and lower bound error criteria to evaluate the accuracy of the approximate upper and lower bounds. Finally, we define yet another error measure for the purpose of comparing approximate and exact results.

A queue length based error criterion that can be used as a stopping rule is defined by:

$$E_s(I) = E_s(\bar{N}_s^{(I)},\underline{N}_s^{(I)}) = \max_s \frac{(\bar{N}_s^{(I)} - \underline{N}_s^{(I)})}{\underline{N}_s^{(I)}}. \tag{5.8.1}$$

The corresponding stopping rule states that the iterations should be halted when $E_s(I) < e_s$, for some given e_s.

It is clear that the number of iterations is a function of the bound chosen for $E_s(I)$. A different error criterion used as a stopping rule and based on mean throughput rates is given by:

$$E_r(\bar{\lambda}_{o,r}^{(I)},\underline{\lambda}_{o,r}^{(I)}) = \max_r \frac{(\bar{\lambda}_{o,r}^{(I)} - \underline{\lambda}_{o,r}^{(I)})}{\underline{\lambda}_{o,r}^{(I)}}\,. \tag{5.8.2}$$

To compare the accuracy of the approximation with the exact solution, we define the following upper bound absolute error criterion:

$$\bar{E}_{r,s}(I) = \max_{r,s} \frac{|N_{rs}(\mathbf{K}) - N_{rs}(\bar{\Lambda}_o^{(I)}, \mathbf{K}_c)|}{N_{rs}(\mathbf{K})} . \tag{5.8.3}$$

In equation (5.8.3) $N_{rs}(\mathbf{K})$ is the exact value of the mean queue length. The corresponding lower bound absolute error criterion is similarly defined as:

$$\underline{E}_{r,s}(I) = \max_{r,s} \frac{|N_{rs}(\mathbf{K}) - N_{rs}(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c)|}{N_{rs}(\mathbf{K})} . \tag{5.8.4}$$

Similar aggregate lower and upper bound absolute error criteria for individual service centers are defined by:

$$\underline{E}_s(I) = \max_{s} \frac{|N_s(\mathbf{K}) - N_s(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c)|}{N_s(\mathbf{K})} , \tag{5.8.5}$$

and

$$\bar{E}_s(I) = \max_{s} \frac{|N_s(\mathbf{K}) - N_s(\bar{\Lambda}_o^{(I)}, \mathbf{K}_c)|}{N_s(\mathbf{K})} . \tag{5.8.6}$$

For simplicity of notation and ease of comparison, we have assumed that the original network is a closed network with population vector $\mathbf{K}_c$. The upper and lower bound absolute error criteria are defined with respect to the mean queue length tolerance, but can similarly be defined for other performance metrics by simply replacing the queue length with those of other metrics. The error magnitude is clearly related to the number of iterations. A rapidly converging and strictly decreasing error function is a good indication of the performance of an iterative algorithm. Another error criterion (also discussed in chapter 4) which is also based on queue lengths is [Cha75]:

$$E(I) = \max_{r,s} \frac{|N_{rs}(\mathbf{K}) - N_{rs}^{*}(\mathbf{K})|}{K_r} . \tag{5.8.7}$$

In equation (5.8.7) $N_{rs}^{*}(\mathbf{K})$ is the estimate of the queue length. The errors evaluated by using (5.8.7) are usually much smaller than those given by equations (5.8.3) and (5.8.4) for a large population vector. This is due to the fact that the denominator of equation (5.8.7) increases more rapidly than those in equations (5.8.3) and (5.8.4). We believe that the error criteria given by equations (5.8.3) and (5.8.4) are more realistic since they are not scaled down by a large number as the one defined in (5.8.7).

The choice of the stopping rule is extremely important for any iterative algorithm. In most studies the effects of the stopping rules and their impact on the ensuing time/space cost are ignored. The accuracy and cost of the iterative algorithm is directly affected by the choice of the stopping rule. The stopping rule we have used in our experiments is the one defined by the error criterion given in (5.8.1). The queue length based stopping rules, while being costly, are best suited for testing the accuracy of approximate QN solution algorithms. The reason for this lies in the fact that queue length values are most sensitive to variations in the parameters of the networks. Therefore, the experimental results are put to extreme tests by using queue length based stopping rules.

In practice, we found a value of e_s=.001 in $E_s(I) \leq e_s$ to be sufficient. This value was usually achieved with an unexpectedly small number of iterations. It is clear that the time complexity (CPU execution time) increases as the value of e_s decreases. The smaller the value of e_s, the larger the number of iterations required. The stopping rule based on (5.8.2) is not as accurate (for the same error bound) as the queue length based rule, but it results in lower computational cost.

5.9 Computational Complexities

To evaluate the time/space complexity of our algorithm, we first compare the time/space complexity of the solution of the mixed network obtained by applying the approximate algorithm to the

original closed network. Suppose the original closed network is composed of R closed chains and M service centers. We further assume that the original network is transformed into a mixed network containing R^c closed chains and $R^o = R - R^c$ open chains.

If the MVA algorithm is used to solve the original network and the approximate mixed network, then the results of sections 2.3 and 2.5 can be used to compare the time/space complexity of the solution to these two networks. We denote the time and space complexities of the approximate mixed network by $T(R^o, R^c, M)$ and $S(R^o, R^c, M)$, respectively. The time and space complexities of the original network are represented by $T(R, M)$ and $S(R, M)$ respectively. The iterative algorithm requires that we solve the mixed network I times. This requires a time complexity roughly equal to I multiplied by the time complexity of the mixed network. We define the space and time complexity of our approximate algorithm by $S(R^o, R^c, M, I)$ and $T(R^o, R^c, M, I)$ respectively, where I represents the number of iterations. Obviously, in the intermediate iterations the network need not be solved completely, as we need only to determine whether the stopping rule is satisfied or not. Furthermore, the visit ratios are determined only once at the beginning. Thus, for the $\underline{E}_s(I) < e_s$ or the $\bar{E}_s(I) < e_s$ stopping rules, the mean queue length values have to be evaluated at each intermediate iteration (iterations 1 through $(I-1)$). The space complexity $S(R^o, R^c, M, I)$ remains the same except for an additional term equal to $2M$ if stopping rule given by (5.8.1) is used, and equal to $2R^o$ if the stopping rule in (5.8.2) is used. Therefore, the space complexities of the solution of the original network, of a single iteration over the approximate network, and of I iterations over the approximate network are as follows:

$$S(R, M) = (R + M + RM)\prod_{i=1}^{R-1}(K_i + 1) + 4RM \ , \tag{5.9.1}$$

$$S(R^o, R^c, M) = (R^c + M + R^c M)\prod_{i=1}^{R^o-1}(K_i + 1) + 5R^o M + 4R^c M \ , \tag{5.9.2}$$

$$S(R^o, R^c, M, I) = S(R^o, R^c, M) + \max(2M, 2R^o) \tag{5.9.3}$$

Clearly, the space complexity of the original network as given by equation (5.9.1) is much larger than those for the mixed network given by equations (5.9.2) and (5.9.3).

Therefore, as the number of closed chains is reduced, the storage space required to solve the resulting approximate mixed queueing network is sharply reduced. This drastic reduction in the space requirements is depicted in Figure 5.3, which shows some semilogarithmic plots for the space complexity of the original and the approximate mixed network as a function of the number of open chains. In each plot it is assumed R=15 and M=15. The population vector for the plot designated as K_i=5 assumes $\mathbf{K}$ = (5,5,...,5), and for the plot designated as K_i = 10 assumes $\mathbf{K}$ = (10,10,...,10). It is evident that the storage space is substantially reduced as the number of closed chains which are opened increases.

We can further investigate the amount of storage space reduction by looking at the difference between the original and approximate networks' storage space requirement. To simplify the analysis, assume that $K_i = K_1$ for all i. If we denote by Δ_1 and Δ_I the following differences:

$\Delta_1 = S(R, M) - S(R^o, R^c, M)$,

$\Delta_I = S(R, M) - S(R^o, R^c, M, I)$,

we can write:

$$\Delta_1 = \Delta_1(R^o, R^c, M) = R^o(1+M)(K_1+1)^{R^o+R^c-1} + (R^c+M+R^cM)(K_1+1)^{R^c-1}\left[(K_1+1)^{R_o} - 1\right] - R^oM \ , \tag{5.9.4}$$

$$\Delta_I = \Delta(R^o, R^c, M, I) = \Delta_1(R^o, R^c, M) - R^oM. \tag{5.9.5}$$

It is immediately clear that $\Delta_1 > 0$ for all nontrivial population vectors ($K_i > 0$ for all i). One important property to notice here is that the reduction in storage space sharply increases as the

population of those closed chains that are transformed into open chains increases.

The derivation of the time complexity for the approximate mixed network and for the case where there are I iterations is more involved. We notice that the time complexity of the MVA solution of the original closed network, derived in section 2.3, is:

$$T(R,M) = 4RM\prod_{r=1}^{R}(K_r + 1) + 3RM + 2RM^3. \tag{5.9.6}$$

The solution of the approximate mixed network in the first iteration has a time complexity that was discussed and derived in section 2.5. Hence, we can write:

$$T(R^o,R^c,M) = 4R^cM\prod_{r=1}^{R^c}(K_r + 1) + 6R^cM + 11R^oM + 2RM^3, \tag{5.9.7}$$

where equation (5.9.7) has been directly taken from Table 3, and the values of X, V^o, and V^c have been replaced by their expressions. Additionally, we have used the fact that:

$$V^o + V^c = 2(R^o + R^c)M^3 + 2R^oM = 2RM^3 + +2R^oM. \tag{5.9.8}$$

To evaluate the time complexity of the solution of the approximate mixed network up to and including the Ith iteration, we assume that, for each iteration step, the mixed QN has to be solved entirely. In addition, for each iteration (except the last), the upper and lower bounds of the external arrival rates to be used in the next iteration have to be evaluated using equations (5.5.1) and (5.5.2). We need M additions per open chain for the first summation in equation (5.5.1). It should be pointed out here that the summation is over the set $S_o(r)$, and correspondingly we need only M additions[3]. We also need one multiplication and one subtraction per open chain. Hence, we need a total of $R^o(M + 2)$ operations to evaluate the upper or lower bounds of the external arrival rates for all open chains in each iteration, and therefore $IR^o(M + 2)$ for I iterations. The stopping rule (5.8.1) or (5.8.2) must then be tested. To do so, we need to find the largest of the upper bound values, i.e., $\max_s \bar{N}_s^{(I)}$ or $\max_r \bar{\lambda}_{o,r}^{(I)}$, and the smallest of the lower bound values, i.e., $\min_s \underline{N}_s^{(I)}$ or $\min_r \underline{\lambda}_{o,r}^{(I)}$. This is because:

$$E_s(I) = \max_s \frac{(\bar{N}_s^{(I)} - \underline{N}_s^{(I)})}{\underline{N}_s^{(I)}} = \max_s \frac{\bar{N}_s^{(I)}}{\underline{N}_s^{(I)}} - 1 = \frac{\max_s \bar{N}_s^{(I)}}{\min_s \underline{N}_s^{(I)}} - 1. \tag{5.9.9}$$

A similar relationship holds for the $E_r(\bar{\lambda}_{o,r}^{(I)}, \underline{\lambda}_{o,r}^{(I)})$ error criterion:

$$E_r(\bar{\lambda}_{o,r}^{(I)}, \underline{\lambda}_{o,r}^{(I)}) = \frac{\max_r \bar{\lambda}_{o,r}^{(I)}}{\min_r \underline{\lambda}_{o,r}^{(I)}} - 1. \tag{5.9.10}$$

The search for the upper or lower bound values in equation (5.9.9) requires $(M-1)$ comparisons per search. Therefore, we need a total of $2(M-1)$ operations per iterations, hence $2I(M-1)$ for all I iterations. Similarly, we need a total of $2I(R^o-1)$ operations for all I iterations of equation (5.9.10). Thus, the time complexity of the approximation for I iterations can be expressed as:

$$T(R^o,R^c,M,I) = IT(R^o,R^c,M) + IR^oM + 2I(\max(R^o,M) - 2). \tag{5.9.11}$$

Figure 5.4 shows plots of the time complexity for the original closed network and the first iteration of the approximate mixed network's solution as a function of the number of open chains in the mixed network. For each plot in Figure 5.4, as in Figure 5.3, we have assumed R=15, M=15. The sharp reduction in time complexity is evident in this plot (note that the vertical axis has a logarithmic scale).

---

3 In reality in all cases we replace $|S_o(r)|$ with M. We therefore are deriving the upper bound for the time and space complexities, since $M < |S_o(r)|$ for all $r \varepsilon O$.

Similarly, the differential reductions in the time complexity required to solve the approximate mixed network for the first and the Ith iterations are defined by:

$$\Gamma_1 = \Gamma_1(R^o, R^c, M) = 4M(K_1+1)^{R^o}\left[(R^o+R^c)(K_1+1)^{R^o} - R^c\right] - 8R^oM - 3R^cM, \tag{5.9.12}$$

and

$$\Gamma_I = \Gamma_I(R^o, R^c, M, I) = T(R, M) - IT(R^o, R^c, M) - IR^oM - 2I(\max(R^o, M) - 2). \tag{5.9.13}$$

In equation (5.9.12), as in the case of the space complexity, we can show that, for all nontrivial cases, $\Gamma_1>0$. However, in equation (5.9.13) the sign and the magnitude of the reduction depends on yet another parameter, which is the number of iterations. It can be shown that substantial reduction in time complexity is achieved for very large networks[4]. For smaller networks, equation (5.9.13) is not very revealing. We found in all the cases considered (in which a small number of iterations actually required to satisfy the stopping rule), that there was always a very substantial reduction. It should be pointed out here that the dependency on the chain population vector is completely removed in the case of open networks. As we shall see in the discussion of the examples in the next section, the time complexity of the iterative algorithm is almost always many orders of magnitude smaller than that of the exact solution.

5.10 Experimental Results

In this section we examine the performance of our algorithm by applying it to the sample model shown in Figure 5.5. In order to evaluate the efficiency and the accuracy of our approximation, we have chosen as example a model that is small enough to be exactly solved. The system modeled in Figure 5.5 consists of a CPU, two disk I/O subsystems, and terminals. The workload is represented by three chains (three classes of jobs). This is a model of a typical site in an aggregation of sites connected by an Ethernet-based distributed system as shown in Figure 5.2. Here it is assumed that all three chains of the model are initially closed chains. The parameters of the model are given in Figure 5.6. Exact and approximate solutions are obtained using RESQ2 [Sau82], and compared with each other. We compare the exact solution with the approximate solution for one open chain, two open chains, and finally a completely open network. The computational complexities of the approximation and of the exact solution are compared. For each case, the errors in the performance measures obtained by solving the approximate network are evaluated. The convergence rate of the iterative estimates will also be examined.

Observe that the original network consists of three closed chains. In the first case we will consider, the network is transformed into a mixed network with one open chain and two closed chains. If error minimization is the objective, then the open chain to be chosen is the one that loads the CPU the least. If we solve the balance equations for an equivalent completely open network, we see that chain 1 should be the first chain to be opened. If the populations were not equal as we have assumed here, and the space and time complexities were the primary concern, then to reduce the complexities one would have to choose the closed chain with the largest population.

We used RESQ2 [Sau82] to solve and compare the approximate and exact solutions. Figure 5.7 shows the CPU utilization versus the number of users (that is, $K_1+K_2+K_3$). Since the CPU is the most utilized server in this model, the maximum error will occur at the CPU. We have chosen $K_1=K_2=K_3$ throughout. Utilization is a robust metric in approximate or exact algorithms for queueing network models. Thus, we expect the error for this performance measure to be the lowest. Figure 5.8 shows the error for the first iteration and for the case where the stopping rule bound, e_s, is chosen to be quite small

4 Networks with large chain populations.

(i.e., e_s=.001). The error in both cases is very small. Figure 5.9 and 5.10 show the mean queue length versus the number of users. The plots for the exact and approximate values coincide in these figures. Figure 5.11 shows the error curves for the first iteration and the case of small stopping rule bound. We should point out here that RESQ2 requires more than 12 Mbytes of memory to solve the model exactly for a population vector larger than (30,30,30), while it solves the approximate model (|O|=1, |C|=2) up to (90,90,90) with only .96 Mbytes. Panacea [Ram82] on the other hand would stop being usable at a utilization of .85 for the CPU. This corresponds to a population vector of (35,35,35). To compare time complexities, we use the processor execution time required to solve the exact and the approximate models. RESQ2 in our environment is running on an IBM 3090/200. Clearly, the time complexity of the approximation is maximum for only one open chain. Figure 12 shows the execution times for the exact and approximate solution to compute utilization, throughput, mean queue length, and mean waiting time for all the chains at all the servers. The curve designated as Approximate (LI), is the execution time of the last iteration. The inclusion of the execution times in the last iterations is for comparison with the total execution times, which include the accumulative execution times over all iterations and is designated as Approximate (AI). It should be pointed out here that this curve should actually be lower (slower slope) since in the intermediate iterations we do not need to solve for all the metrics. Even without eliminating these unnecessary computations, the execution time improvement with respect to that required for an exact solution is clearly evident.

Figures 5.13 - 5.22 compare the exact solution and the approximate solution with two open chains and one closed chain (|O|=2, |C|=1). In this case, while the error slightly increases for all performance metrics, the error is indeed very satisfactorily low.

Figures 5.17 and 5.18 compare the exact and the approximate solution with two open chains and one closed chain. The purpose for including these two plots is to compare two different error criterion. We observed earlier that the error criterion we have chosen is given by (4.6.6). The more widely used criterion is given by (4.6.7). The difference between the two criteria is clearly depicted in figures 5.17 and 5.18. Figure 5.9 is based on the error criterion given in (4.6.6) and Figure 5.10 is based on (4.6.7). Clearly, errors are much smaller for the criterion given by (4.6.7). For this reason, we prefer (4.6.6) over the widely used criterion defined by (4.6.7). In both cases the error is slightly higher than that of one open chain. The error performance for the approximate network with two open chains is indeed very satisfactory.

Figure 5.22, which displays the errors in the mean waiting time and the mean queue length as a function of the number of users, shows clearly how fast and drastic an improvement can be achieved even with a small increase in the number of iterations. This figure also shows that the iterative algorithm converges very rapidly without incurring unacceptable errors.

Figures 5.23 - 5.30 show the curves for the case of 3 open chains and compare them to those given by the exact solution. This is the case of maximum error for the approximation. Again, the iteration converges very rapidly. This is also the situation where the approximation algorithm's time and space complexities become independent of the population vector. While the time complexity of each iteration as a function of the total chain population remains constant, the total time complexity depends (nonlinearly) on the stopping rule bound and the resulting number of iterations. Figure 5.30 compares the CPU execution times for the exact and approximate solutions using RESQ2 for both cases.

To summarize, our algorithm allows a flexible and hierarchical approximate solution of QNs in the following sense. For a stated constraint on accuracy, the number of closed chain->open chain transformations can be chosen to minimize the computational costs. The resulting approximate network can then be solved for a hierarchy of bounds. In this stage the number of iterations can be chosen accordingly, subject to the desirable accuracy. This flexibility is in contrast to the asymptotic expansion in Panacea where the undelying approximation is set at the beginning. As we have seen in section 5.7., the approximation is asymptotically correct as the population becomes very large. However, the condition given by (4.5.3) requires that $\lim_{K\to\infty} K_r \mu_{o,r}$ exist and be finite for the chains that are opened. Therefore, if the values of $\mu_{o,r}$ are kept fixed as the population increases, the error will also increase. The

overall effect of chain population, service rate, and service demand of each chain at each of the service centers is better understood by looking at the utilization of each service center due to the jobs belonging to each open chain. From equations (5.5.5) and (5.5.6), it is clear that, for very high values of ρ_{rs}^{o}, the values of the external arrival rate estimates become very sensitive to variations in ρ_{rs}. On the other hand, the computational complexities of the algorithm are very sensitive to the sizes of closed chain populations. Thus, if a reduction in computational complexity is the objective, one has to choose those chains with the largest population. If the minimization of error is the main objective, then the chains least loading the most utilized service center should be picked. When in the QN model some service centers are under heavy load conditions, known asymptotic expressions can easily be used for the evaluation of performance measures. Once the parameters of a QN model are given, the equations derived in section 5.9 may be used to estimate the space and time complexities of the model's solution. This step allows one to choose the number of chains to be opened initially, so that the maximum allowable limits for time and space complexities are not exceeded. The structure of equations given by (5.5.5) and (5.5.6) suggests that some round-off error might occur for extreme values of open chain utilizations. A scaling algorithm might be needed to avoid this problem.

5.11 Conclusions

In this chapter, we have presented a hierarchical network transformation based approximate technique for solving multichain QNs with large numbers of chains, service centers, and populations. The technique applies to product form QNs consisting of only single server fixed rate and IS service centers. Extensions to certain load dependent QNs are possible. The method provides for a flexible hierarchy of approximations that allows a smooth tradeoff between feasibility, accuracy, and computational cost of the solution. The performance metrics evaluated using this algorithm are many orders of magnitude less expensive than those obtained by exact methods. The approximate solutions are asymptotically correct. The iterative part of the algorithm converges rapidly, and provides upper and lower bounds for the performance metrics. The accuracy, speed, and convergence behavior of the algorithm have been experimentally verified for a number of queueing networks. We have suggested several error criteria that we believe are more realistic and permit a fair comparison between exact and accurate techniques. Finally, the algorithm can be easily implemented with minimal effort.

We can apply our algorithm to the modeling and performance evaluation of large internetwork-based distributed systems. This can be done using a decoupling algorithm (see chapter 6) that transforms the QN models of configurations such as those shown in Figure 5.1 into approximate mixed QN submodels that can be solved independently.

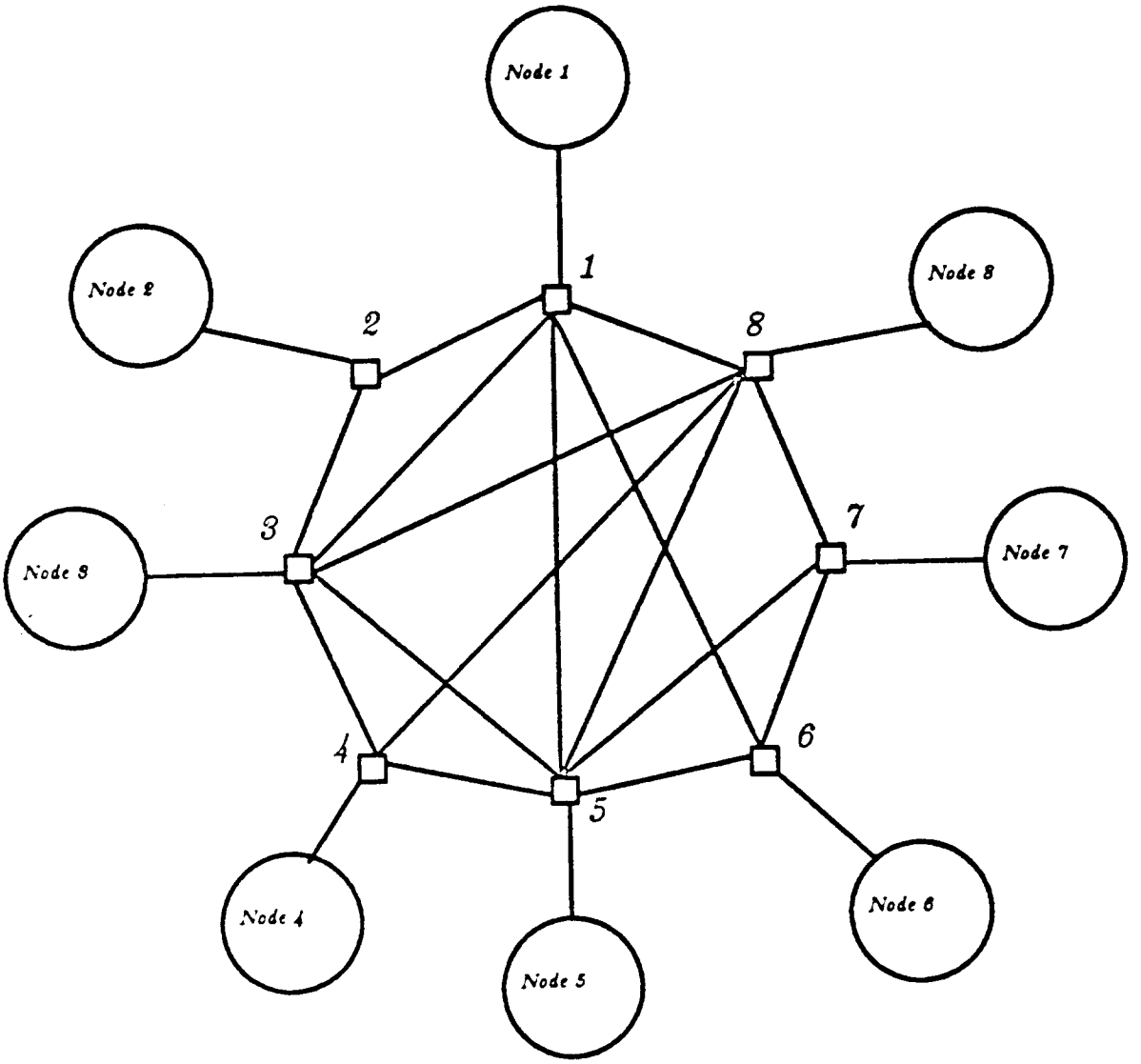

Figure 5.1 Internetwork-Based Distributed System Configuration

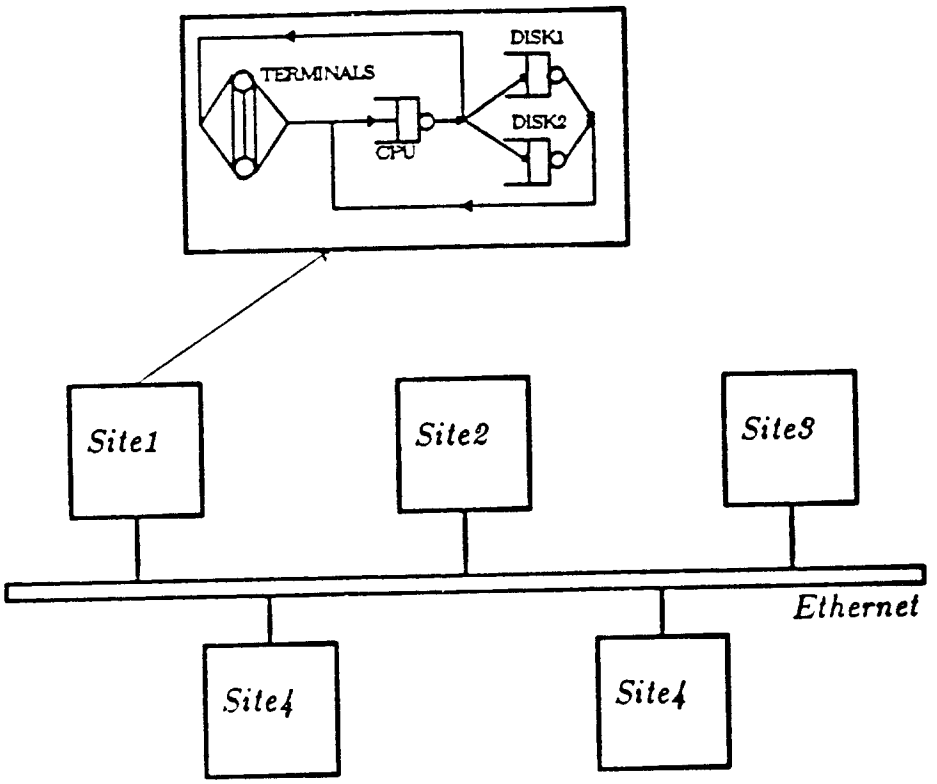

Figure 5.2 A Local Area Network-Based Distributed System Configuration

Space Complexity vs Number of Open Chains

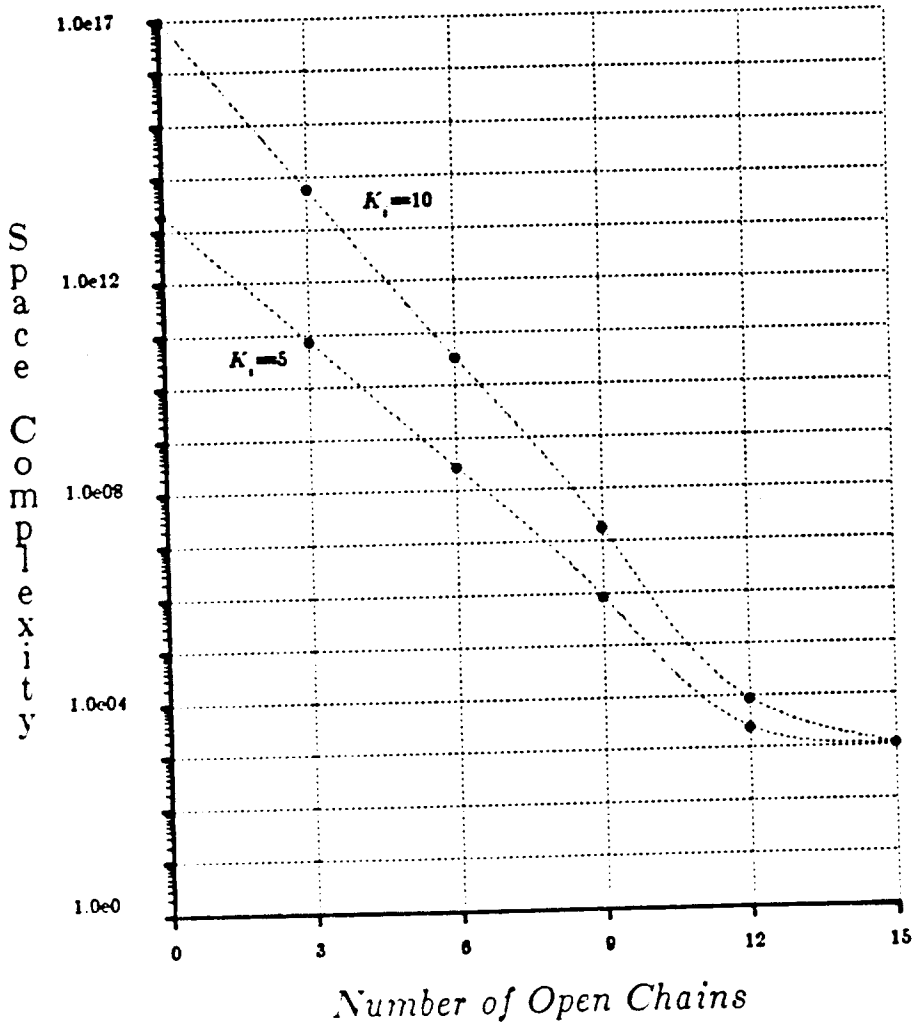

Figure 5.3

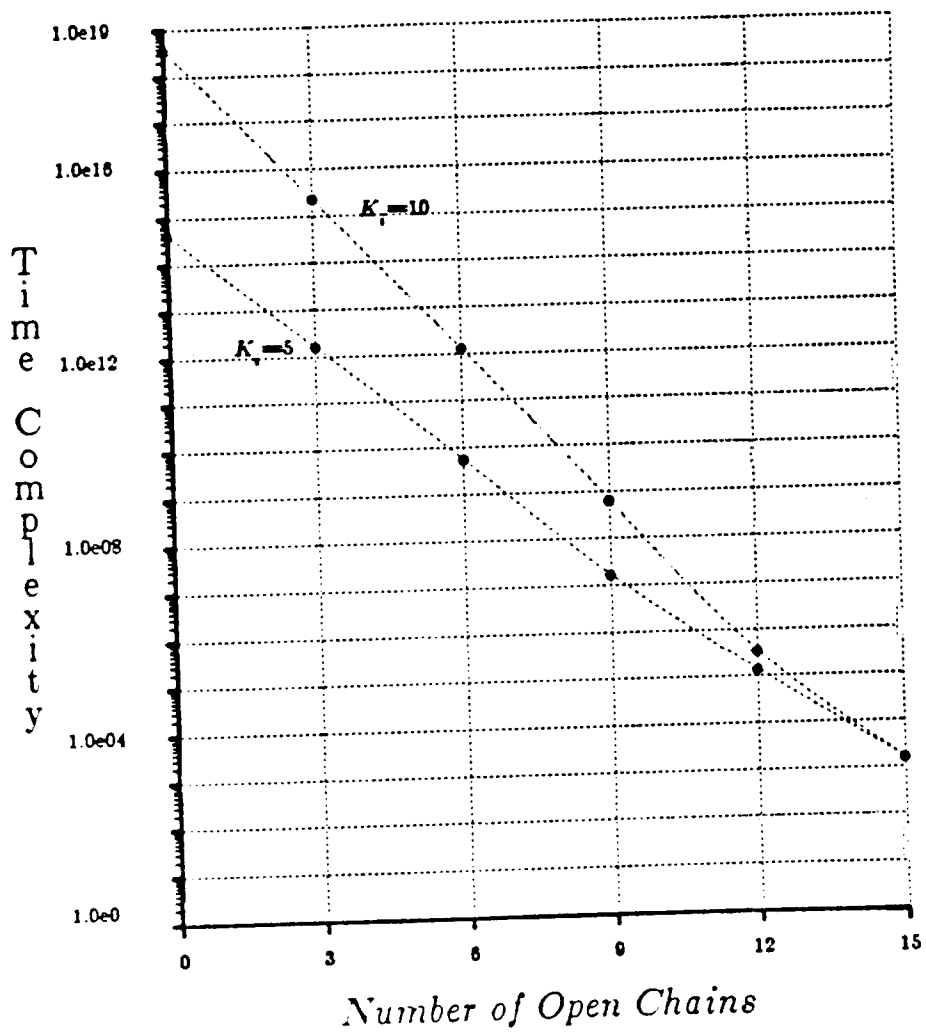
Time Complexity vs Number of Open Chains
1.0e19
1.0e16
1.0e12
1.0e08
1.0e04
1.0e0
Time Complexity
K_c=10
K_c=5
0
3
6
9
12
15
Number of Open Chains

Figure 5.4

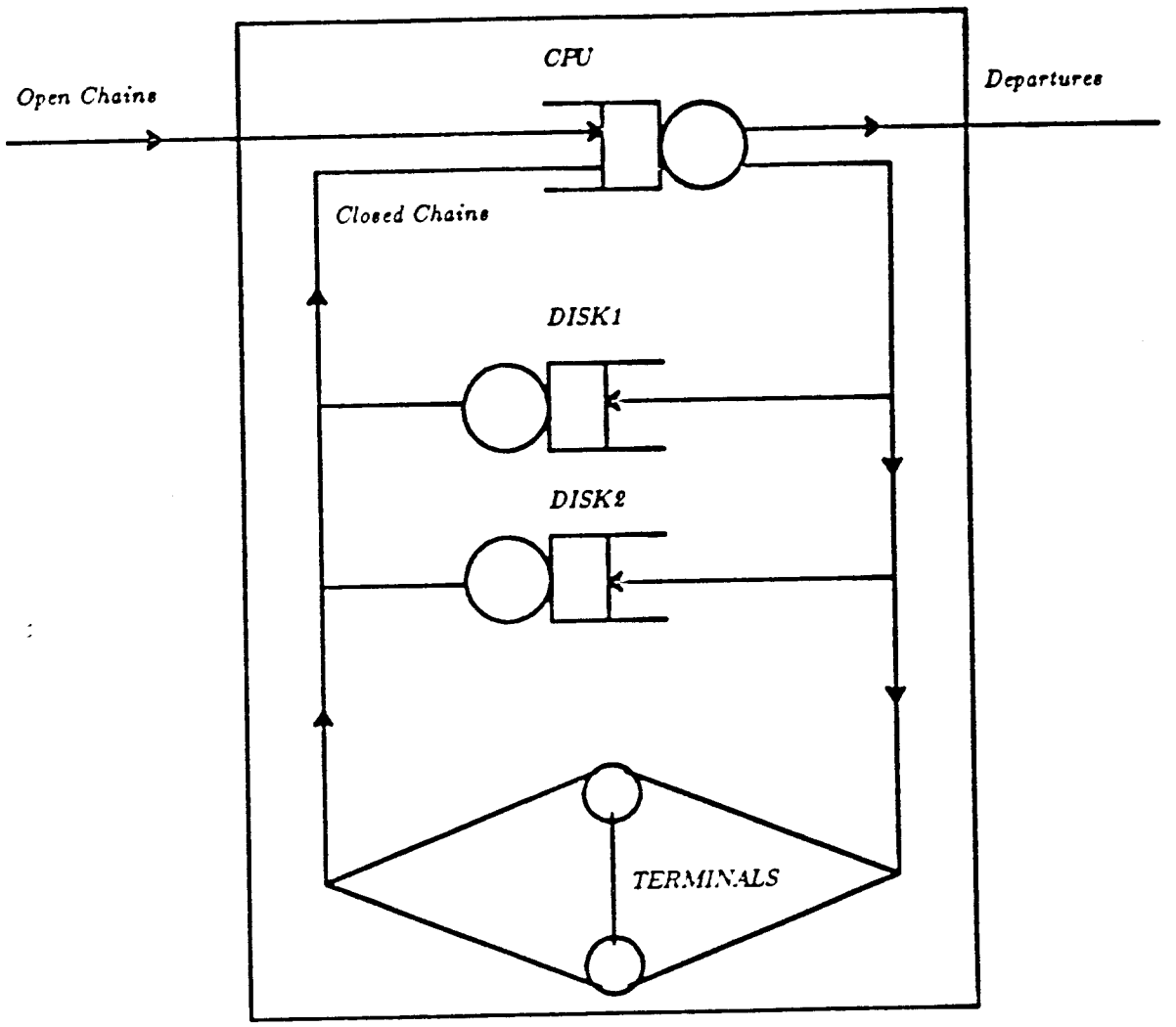

Figure 5.5 Example Model Used for Experimental Results

```
MODEL:CLOSE
METHOD:numerical, NUMERIC PARAMETERS:K1 K2 K3
NUMERIC IDENTIFIERS:s1 s2 s3 t1 t2 t3 q
s1:1, s2:1.5, s3:2, t1:450, t2:150, t3:200, q:1.

QUEUE:terminal, TYPE:IS, CLASS LIST:u1 u2 u3
SERVICE TIMES:t1 t2 t3

QUEUE:cpu, TYPE:PS, CLASS LIST:c1 c2 c3
SERVICE TIMES:s1 s2 s3

QUEUE:disk1, TYPE:FCFS, CLASS LIST:d1 d2
SERVICE TIMES:Q

QUEUE:disk2 TYPE:FCFS CLASS LIST:d3
SERVICE TIMES:q

CHAIN:m1, TYPE:closed, POPULATION:K1
u1->c1, c1->d1 u1; 1/5 4/5, d1->c1

CHAIN:m2, TYPE:closed, POPULATION: K2
u2->c2, c2->d2 u2; 1/8 7/8, d2->c2

CHAIN:m3, TYPE:closed, POPULATION:K3
u3->c3, c3->d3 u3; .1 .9, d3->c3
```

Figure 5.6 The Parameters of the Network in Figure 5.3.

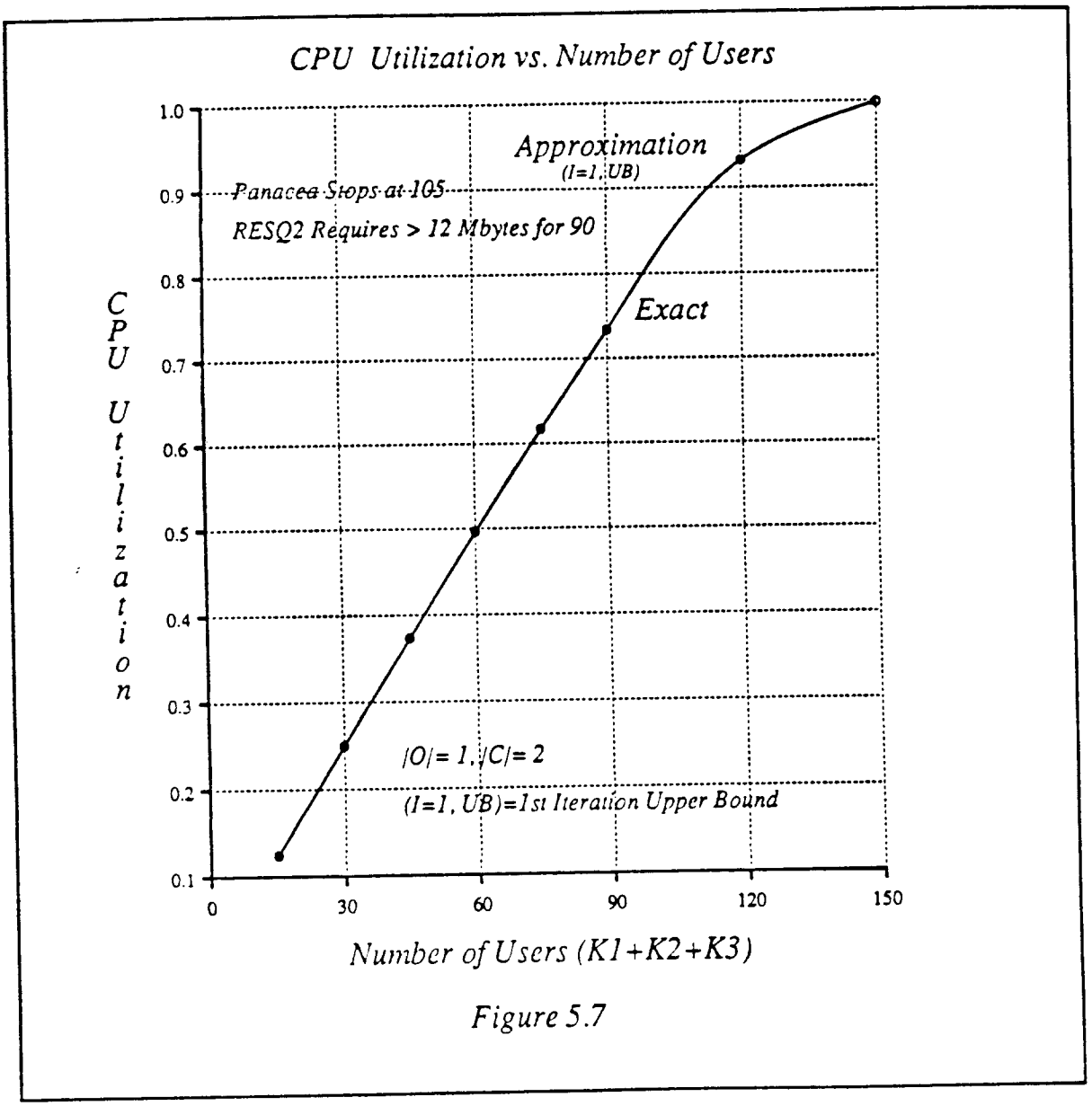
CPU Utilization vs. Number of Users
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
CPU Utilization
Approximation
(I=1, UB)
Panacea Stops at 105
RESQ2 Requires > 12 Mbytes for 90
Exact
|O|= 1, |C|= 2
(I=1, UB)=1st Iteration Upper Bound
0
30
60
90
120
150
Number of Users (K1+K2+K3)

Figure 5.7

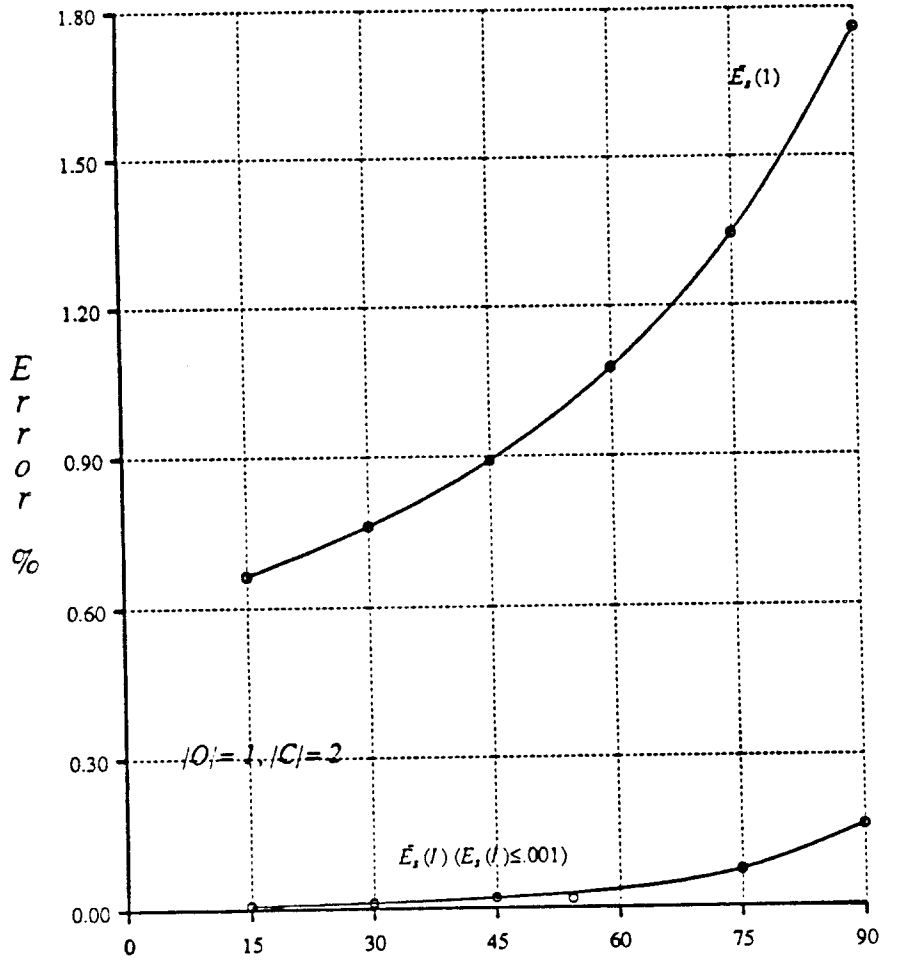
Utilization Error vs. Number of Users
1.80
1.50
1.20
0.90
0.60
0.30
0.00
Error %
$\bar{E}_t(1)$
$|O_i|=1, |C_i|=2$
$\bar{E}_t(I)$ $(E_t(I)\leq.001)$
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.8

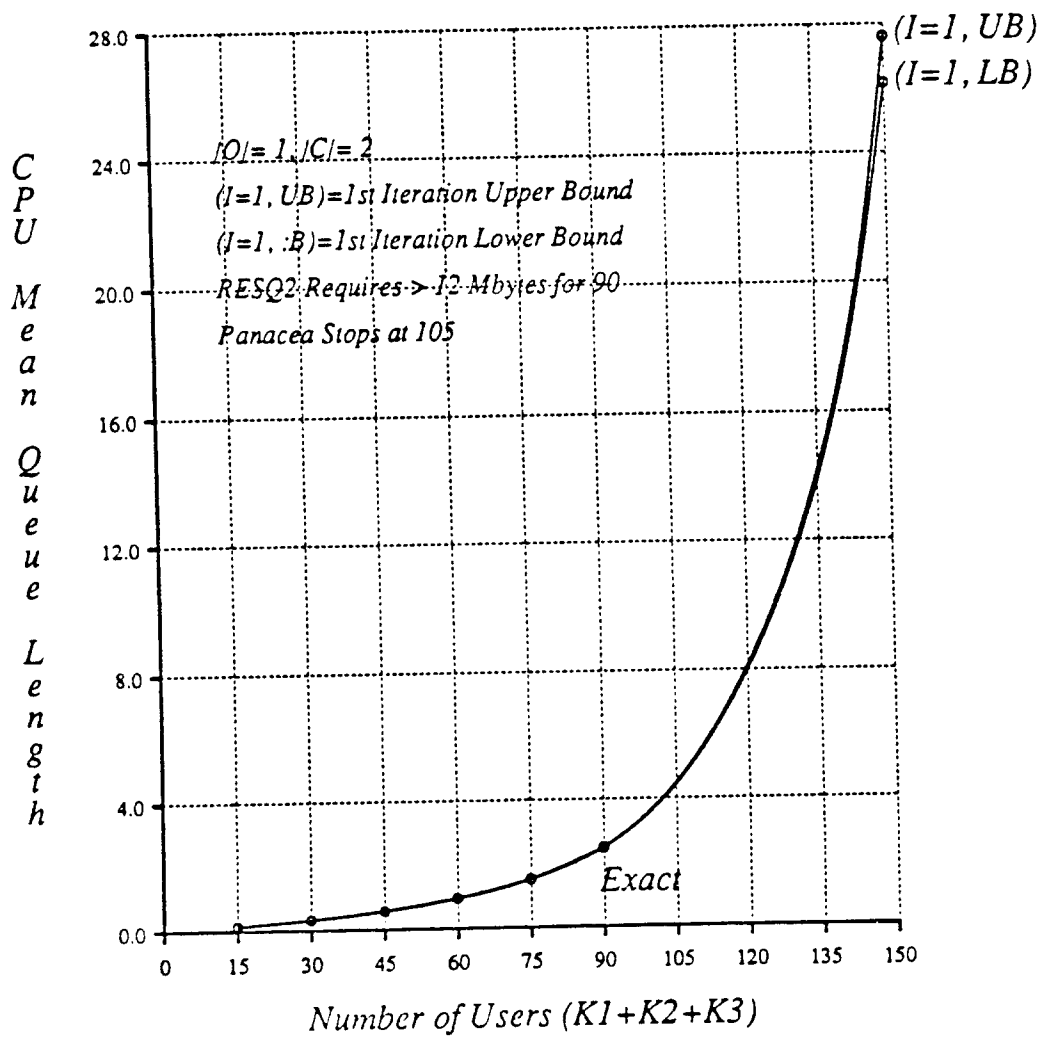
CPU Q-Length vs. Number of Users
(I=1, UB)
(I=1, LB)
|Q|= 1, |C|= 2
(I=1, UB)=1st Iteration Upper Bound
(I=1, LB)=1st Iteration Lower Bound
RESQ2 Requires > 12 Mbytes for 90
Panacea Stops at 105
Exact
CPU Mean Queue Length
28.0
24.0
20.0
16.0
12.0
8.0
4.0
0.0
0
15
30
45
60
75
90
105
120
135
150
Number of Users (K1+K2+K3)

Figure 5.9

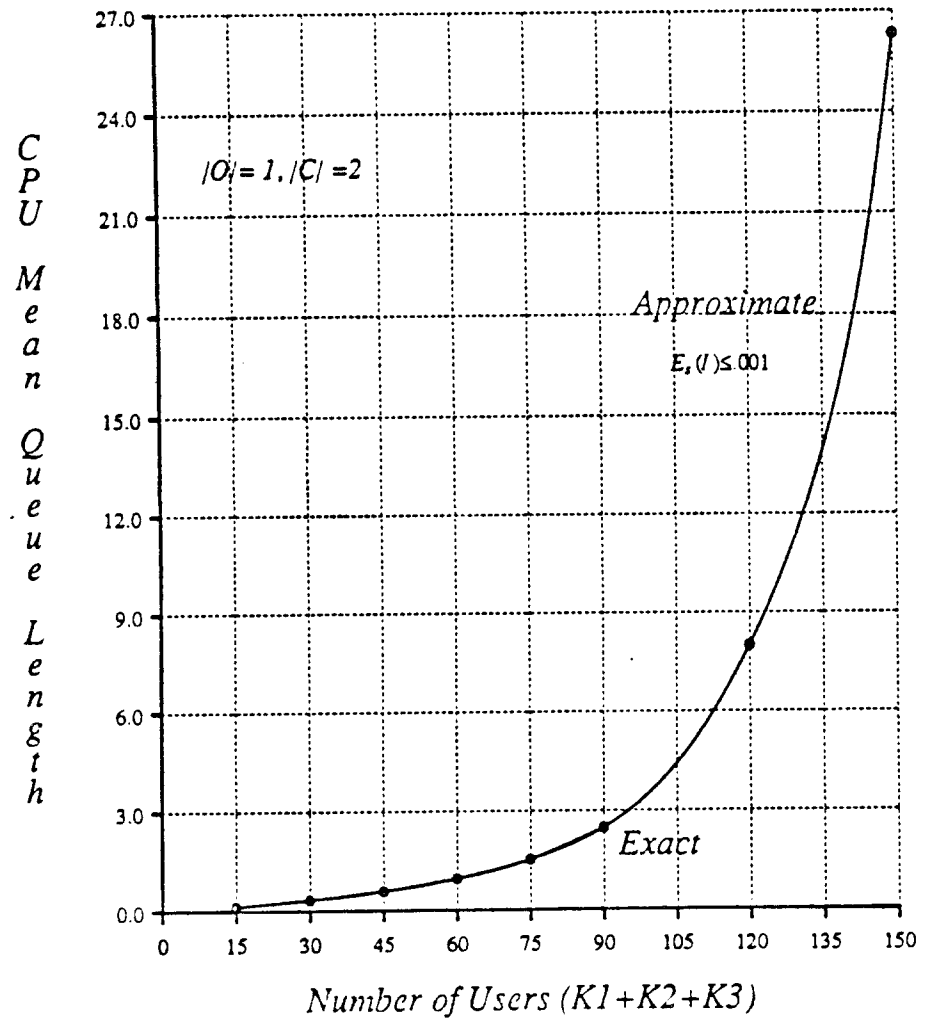
CPU Q-Length vs. Number of Users
CPU Mean Queue Length
27.0
24.0
21.0
18.0
15.0
12.0
9.0
6.0
3.0
0.0
|O| = 1, |C| =2
Approximate
E, (l) ≤ .001
Exact
0
15
30
45
60
75
90
105
120
135
150
Number of Users (K1+K2+K3)

Figure 5.10

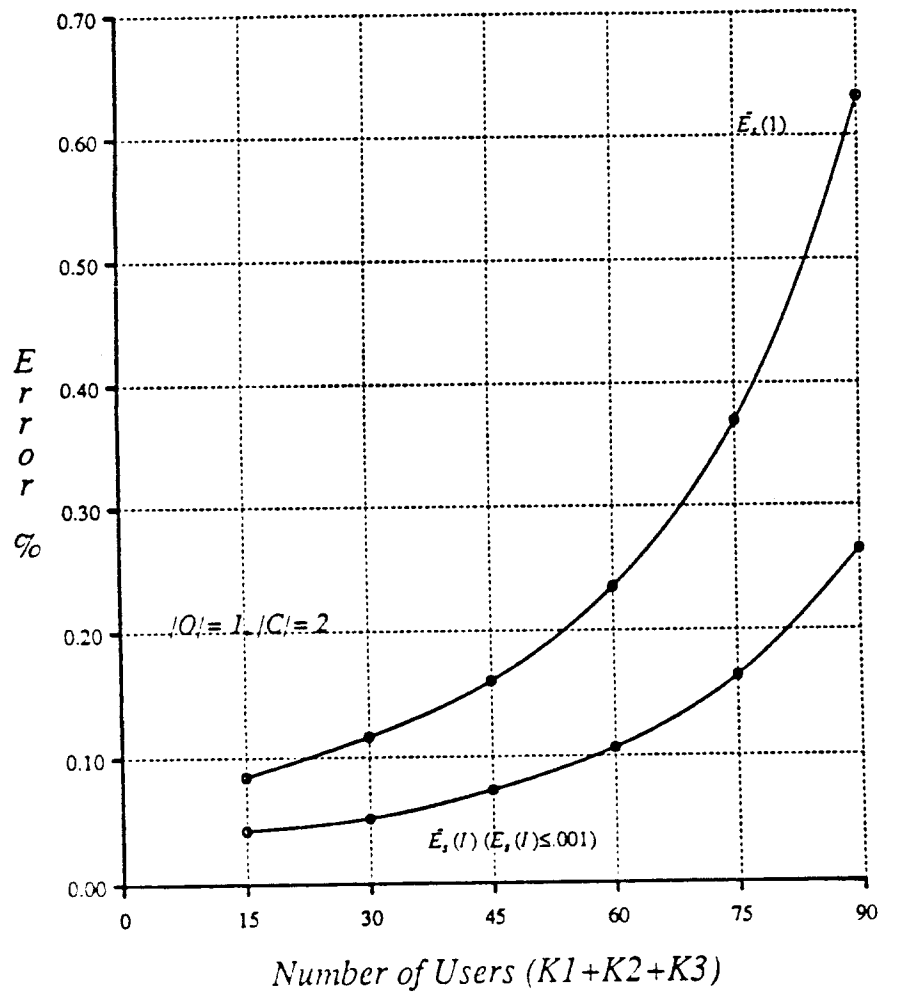
Q-Length Error vs. Number of Users
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Error %
$\hat{E}_1(1)$
$|O_i| = 1, |C_i| = 2$
$\hat{E}_2(I)$ $(E_2(I) \leq .001)$
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.11

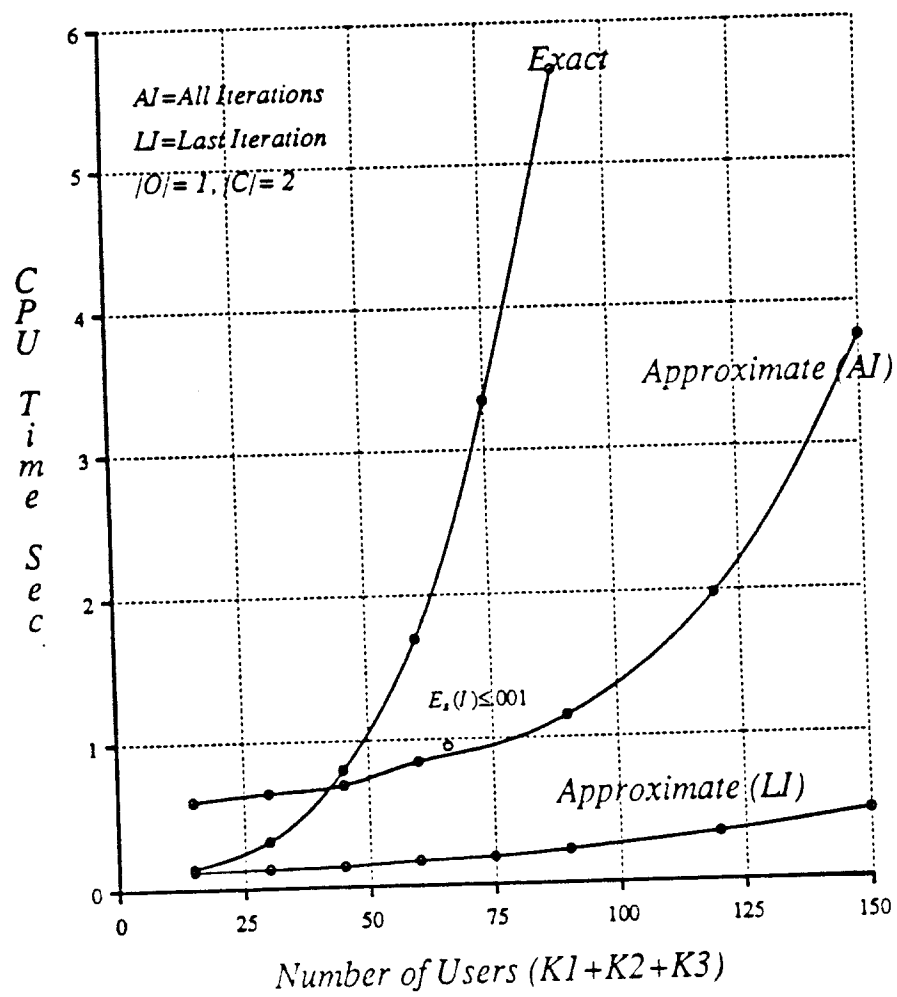
Time Complexity vs. Number of Users
AI=All Iterations
LI=Last Iteration
|O|= 1, |C|= 2
Exact
Approximate (AI)
Approximate (LI)
$E_a(I) \leq .001$
CPU Time Sec
0
1
2
3
4
5
6
0
25
50
75
100
125
150
Number of Users (K1+K2+K3)

Figure 5.12

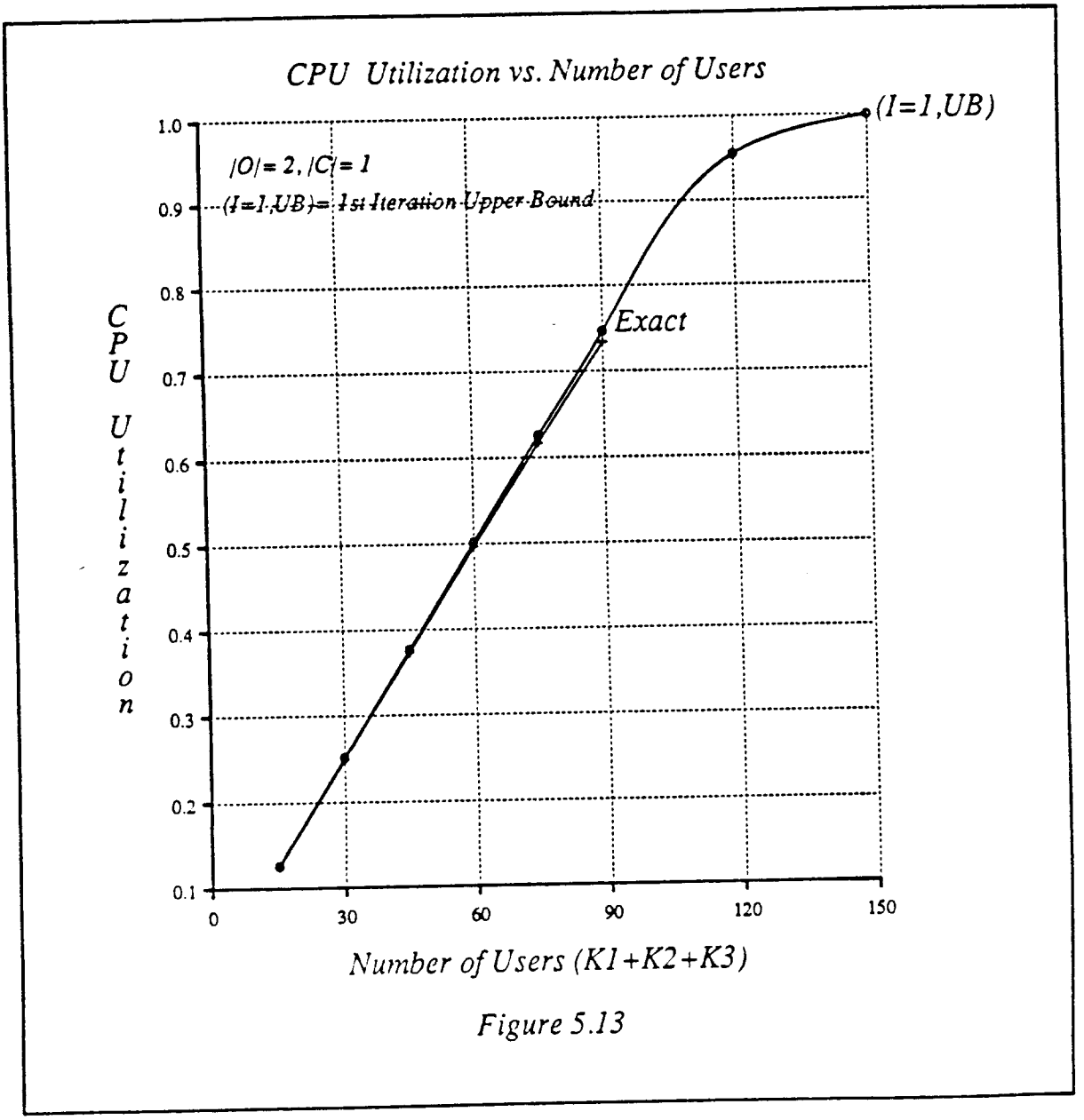
CPU Utilization vs. Number of Users
(I=1,UB)
|O|= 2, |C| = 1
(I=1,UB)= 1st Iteration Upper Bound
Exact
CPU Utilization
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
30
60
90
120
150
Number of Users (K1+K2+K3)

Figure 5.13

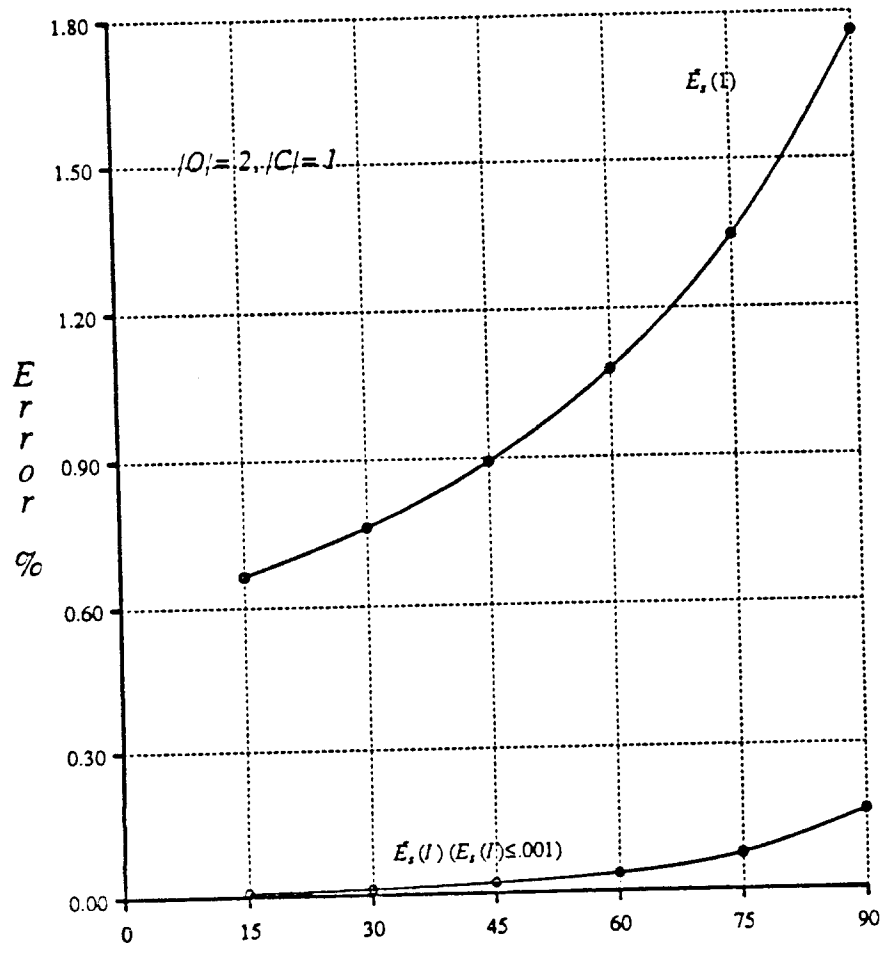
Utilization Error vs. Number of Users
1.80
1.50
1.20
0.90
0.60
0.30
0.00
Error %
|O|=2, |C|=1
$\hat{E}_s(I)$
$\hat{E}_s(I)$ ($E_s(I) \leq .001$)
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.14

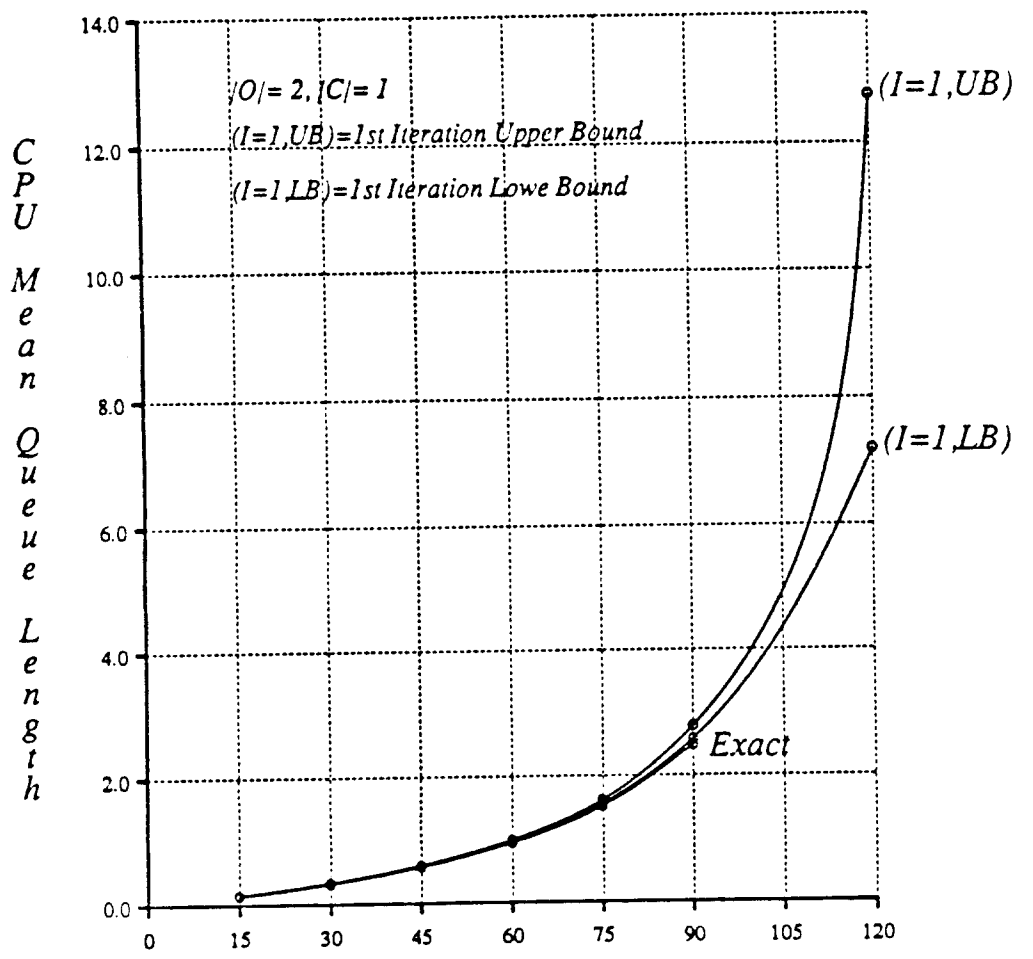
CPU Q-Length vs. Number of Users
|O|= 2, |C|= 1
(I=1,UB)=1st Iteration Upper Bound
(I=1,LB)=1st Iteration Lowe Bound
(I=1,UB)
(I=1,LB)
Exact
CPU Mean Queue Length
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
0
15
30
45
60
75
90
105
120
Number of Users (K1+K2+K3)

Figure 5.15

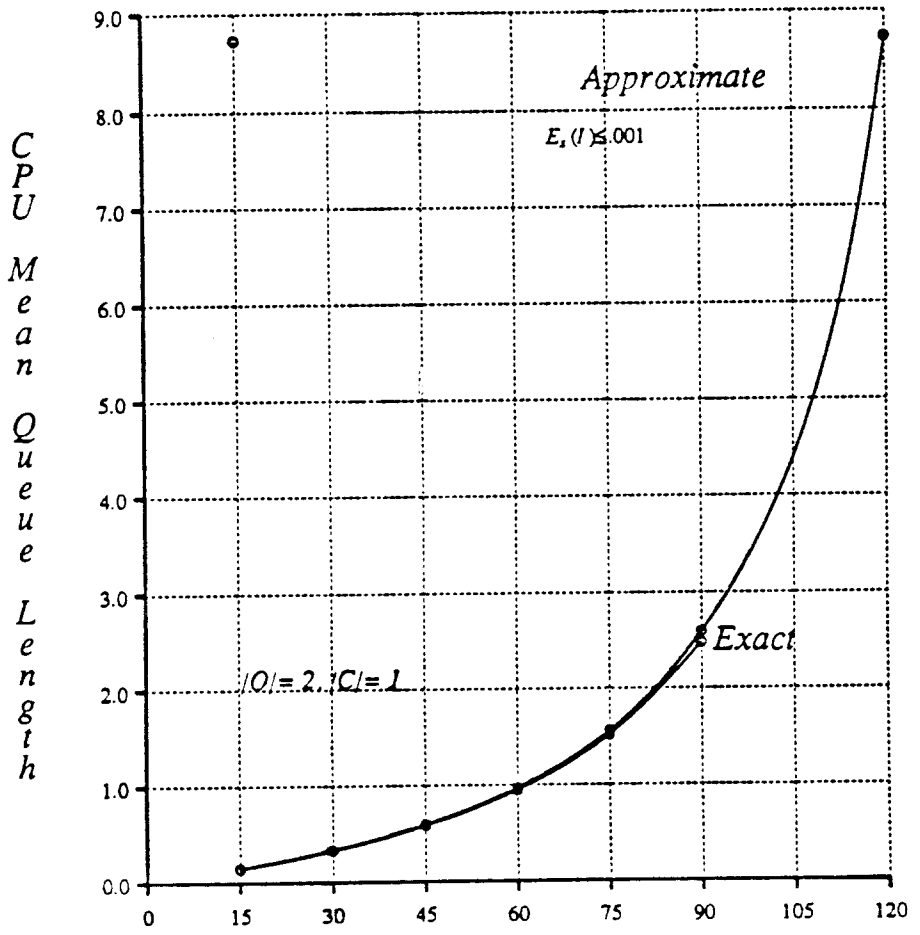
CPU Q-Length vs. Number of Users
CPU Mean Queue Length
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0
15
30
45
60
75
90
105
120
Approximate
$E_1(l) \leq .001$
Exact
/O/= 2, /C/= 1
Number of Users (K1+K2+K3)

Figure 5.16

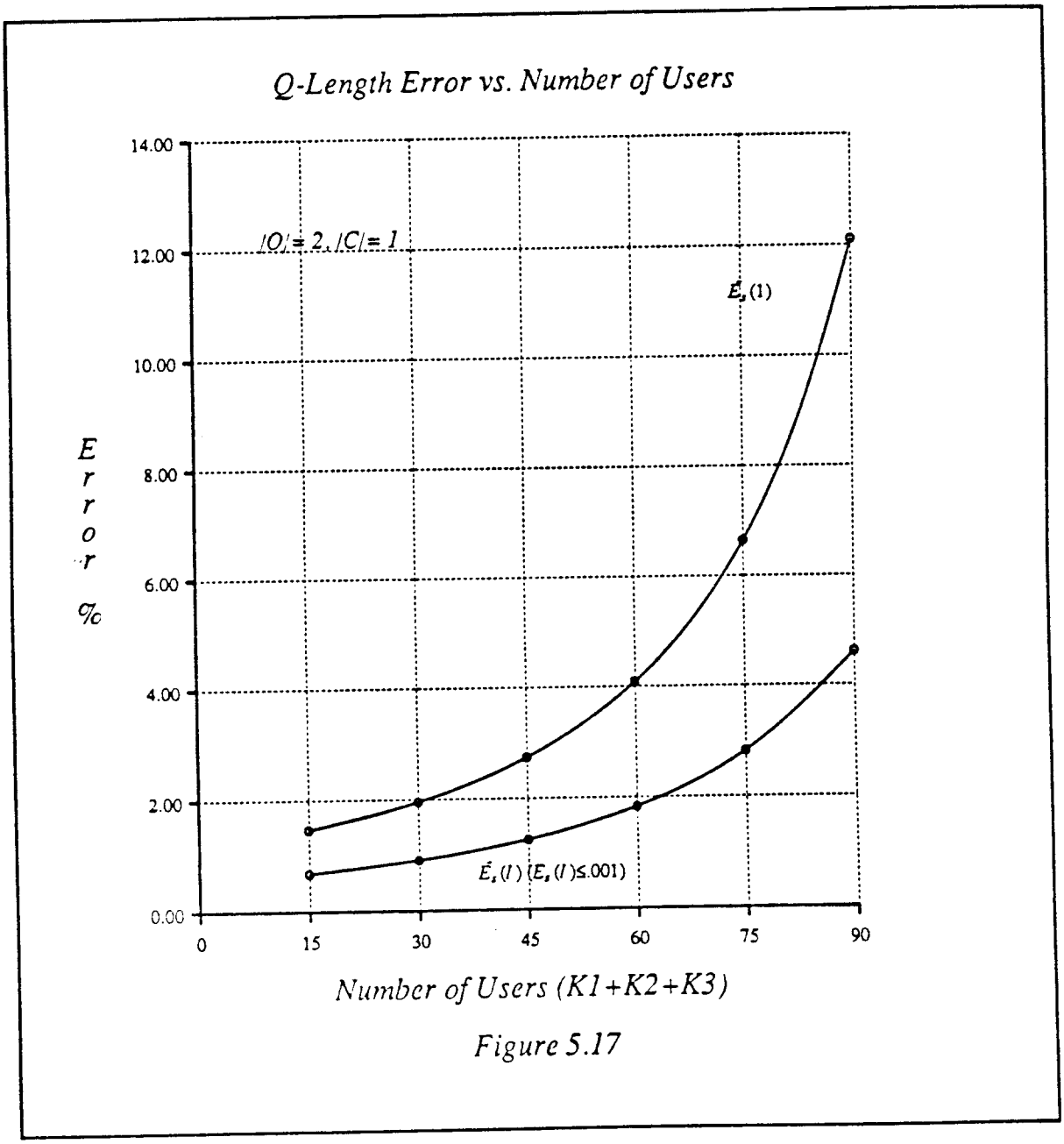
Q-Length Error vs. Number of Users
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Error %
|O|= 2, |C|= 1
$\hat{E}_s(1)$
$\hat{E}_s(l)$ ($E_s(l) \leq .001$)
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.17

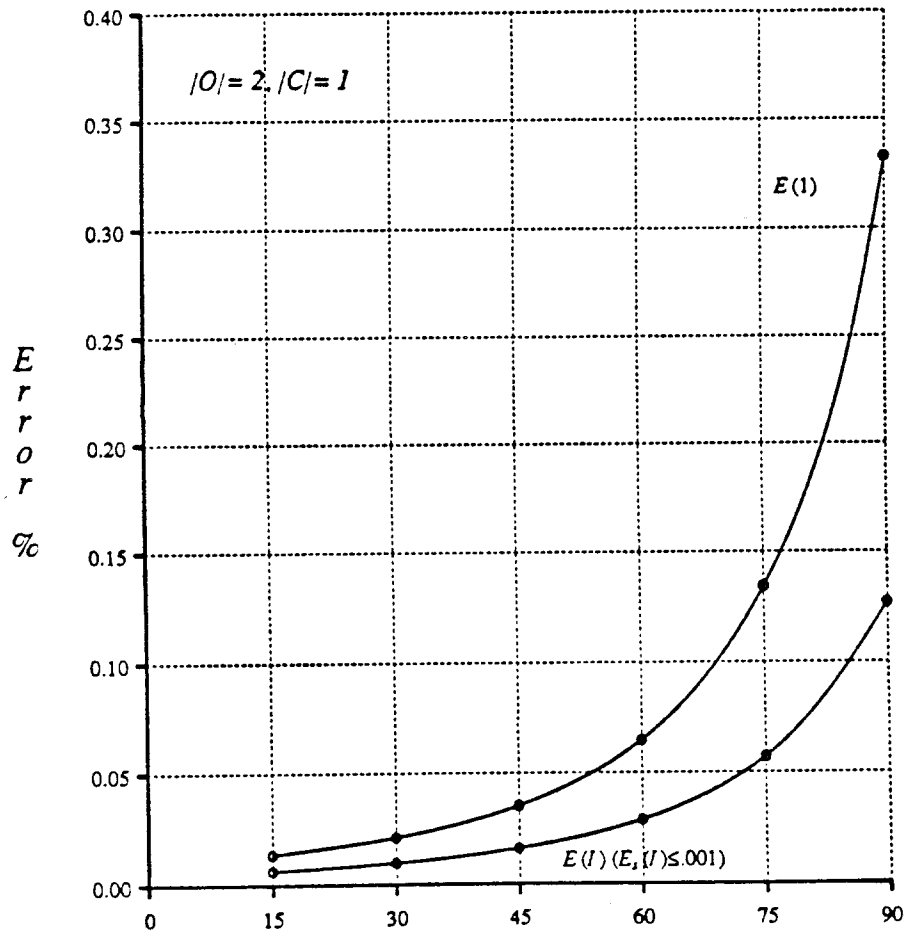
Q-Length Error vs. Number of Users
/O/= 2, /C/= 1
E(1)
E(I) (E, I)≤.001)
Error %
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.18

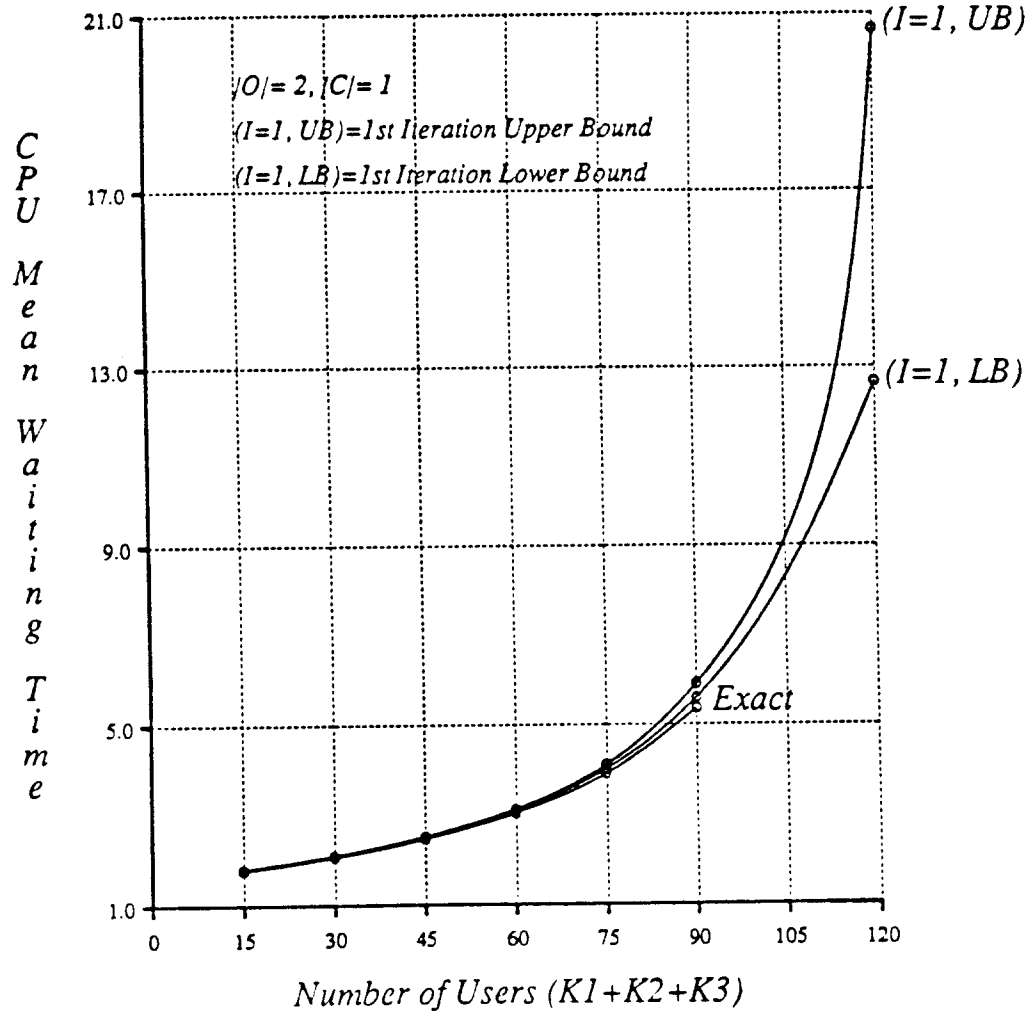
CPU W-Time vs. Number of Users
21.0
17.0
13.0
9.0
5.0
1.0
0
15
30
45
60
75
90
105
120
|O|= 2, |C|= 1
(I=1, UB)=1st Iteration Upper Bound
(I=1, LB)=1st Iteration Lower Bound
(I=1, UB)
(I=1, LB)
Exact
CPU Mean Waiting Time
Number of Users (K1+K2+K3)

Figure 5.19

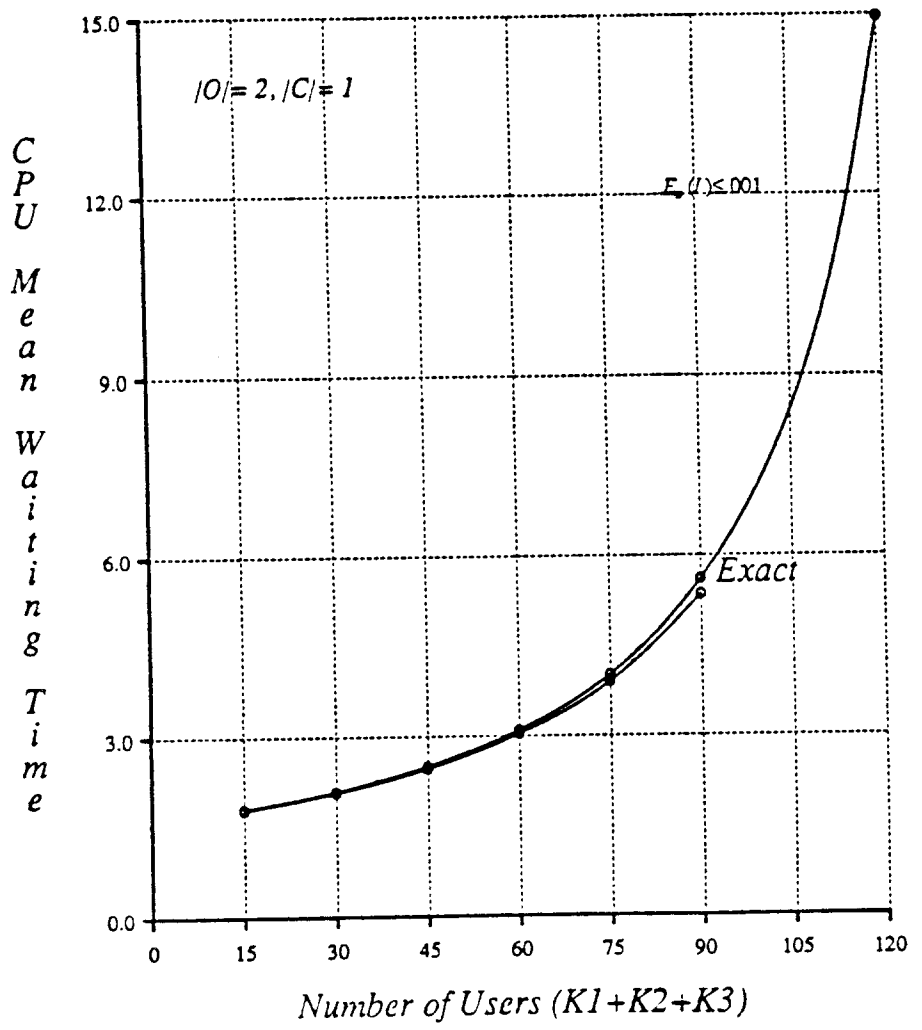
CPU W-Time vs. Number of Users
15.0
12.0
9.0
6.0
3.0
0.0
|O|= 2, |C|= 1
Exact
0
15
30
45
60
75
90
105
120
CPU Mean Waiting Time
Number of Users (K1+K2+K3)

Figure 5.20

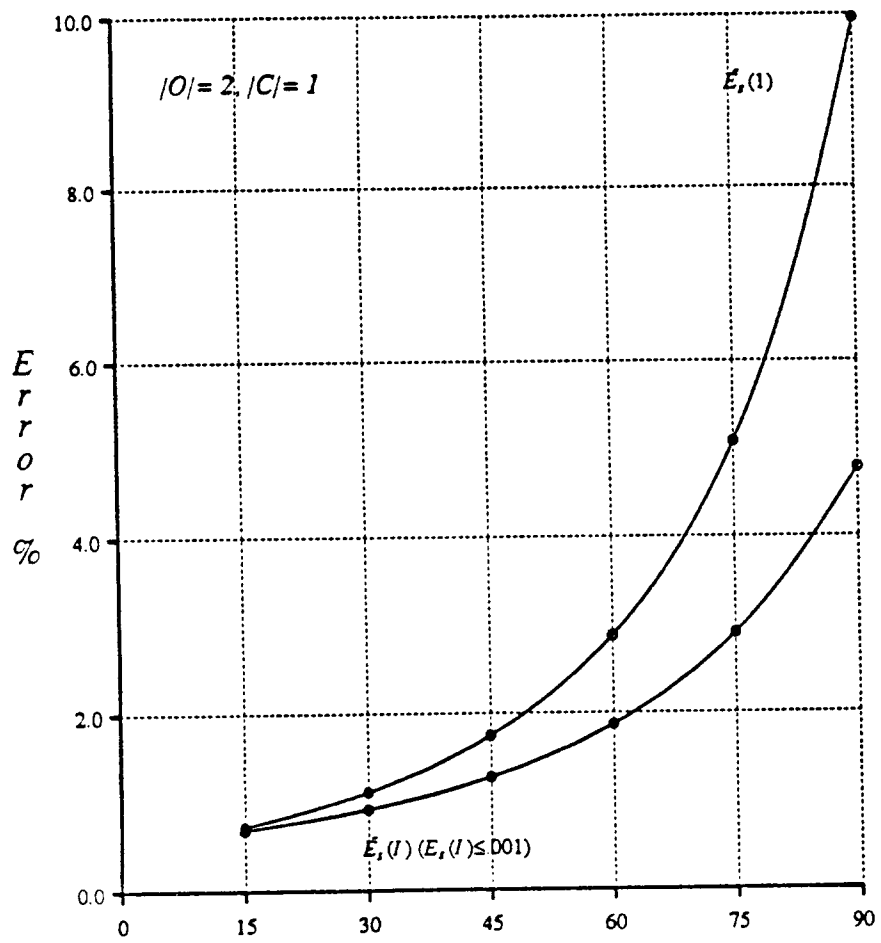
Mean W-Time Error vs. Number of Users
10.0
8.0
6.0
4.0
2.0
0.0
Error %
|O|= 2, |C|= 1
$\dot{E}_t(1)$
$\dot{E}_t(I)$ ($E_t(I) \leq .001$)
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.21

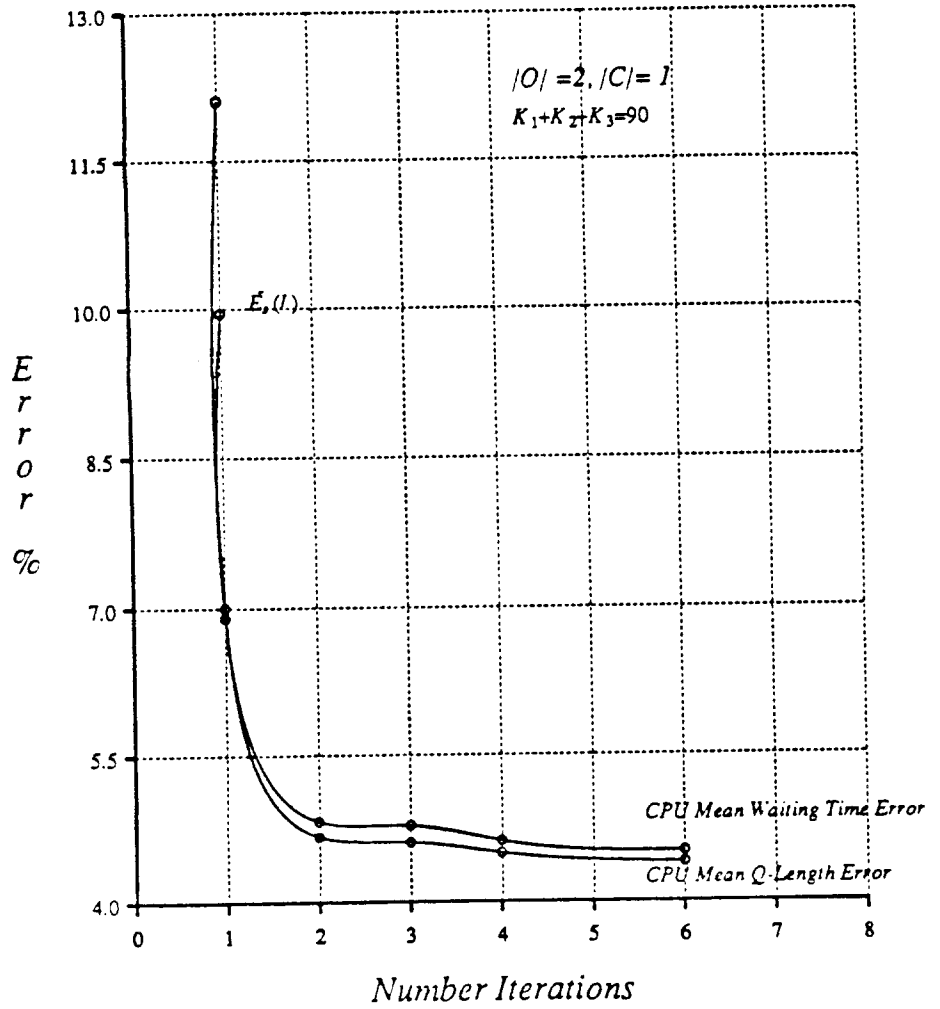
Error vs. Number of Iterations
|O| =2, |C|= 1
K1+K2+K3=90
E1(I)
CPU Mean Waiting Time Error
CPU Mean Q-Length Error
Error %
Number Iterations
13.0
11.5
10.0
8.5
7.0
5.5
4.0
0
1
2
3
4
5
6
7
8

Figure 5.22

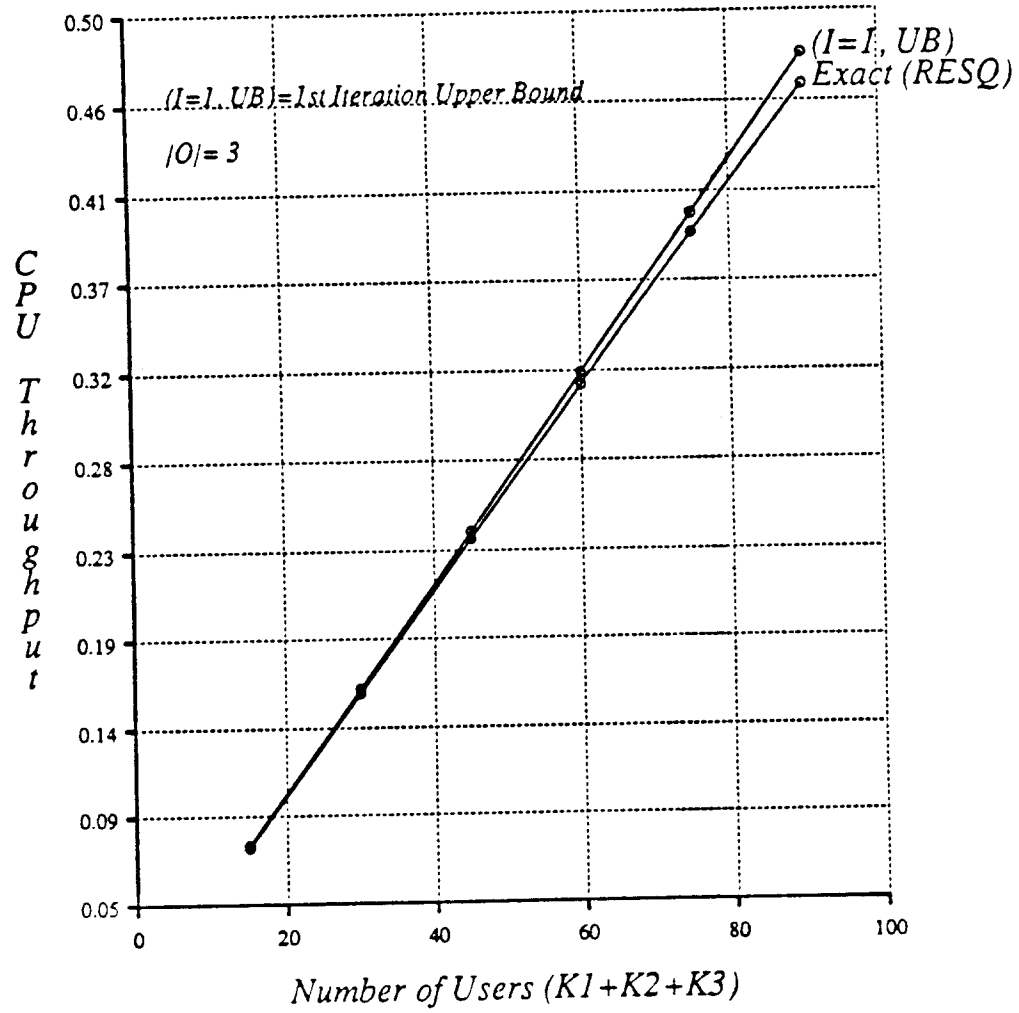
CPU Throughput vs. Number of Users
(I=I., UB)
Exact (RESQ)
(I=1, UB)=1st Iteration Upper Bound
|O|= 3
CPU Throughput
0.50
0.46
0.41
0.37
0.32
0.28
0.23
0.19
0.14
0.09
0.05
0
20
40
60
80
100
Number of Users (K1+K2+K3)

Figure 5.23

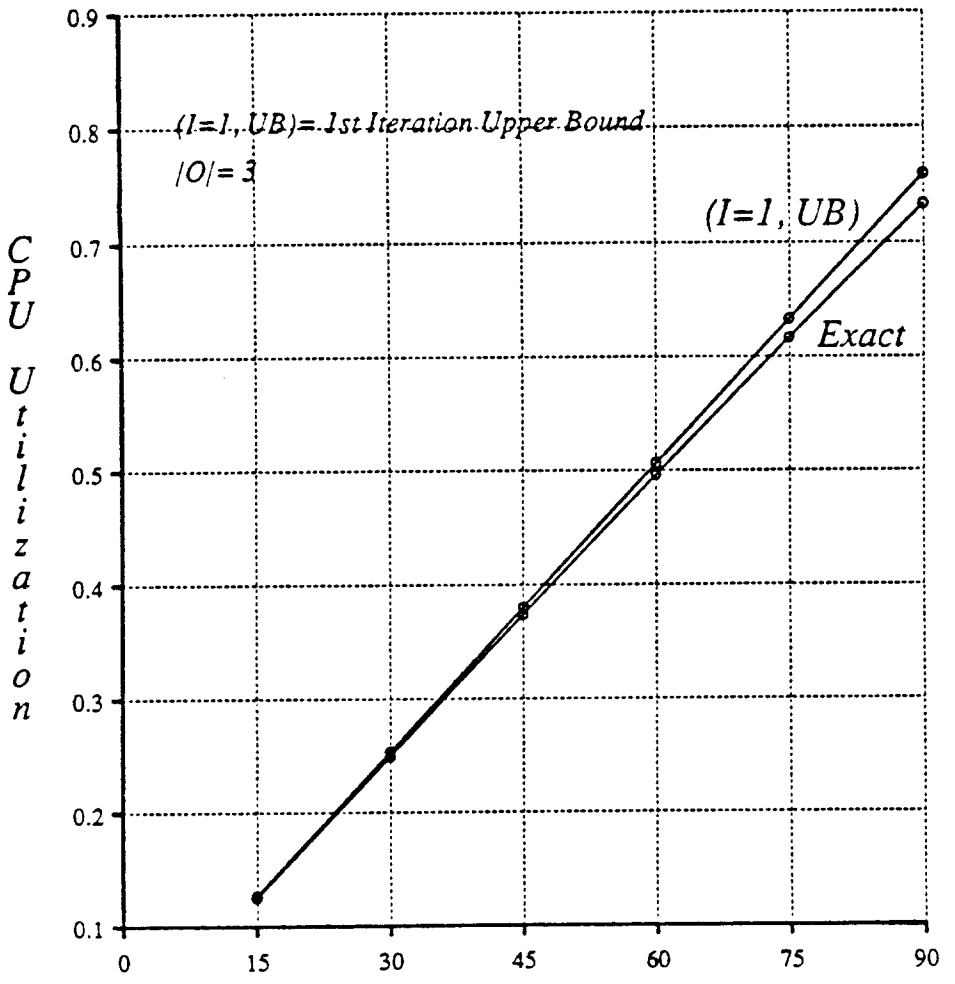
CPU Utilization vs. Number of Users
(I=1, UB)= 1st Iteration Upper Bound
|O|= 3
(I=1, UB)
Exact
CPU Utilization
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.24

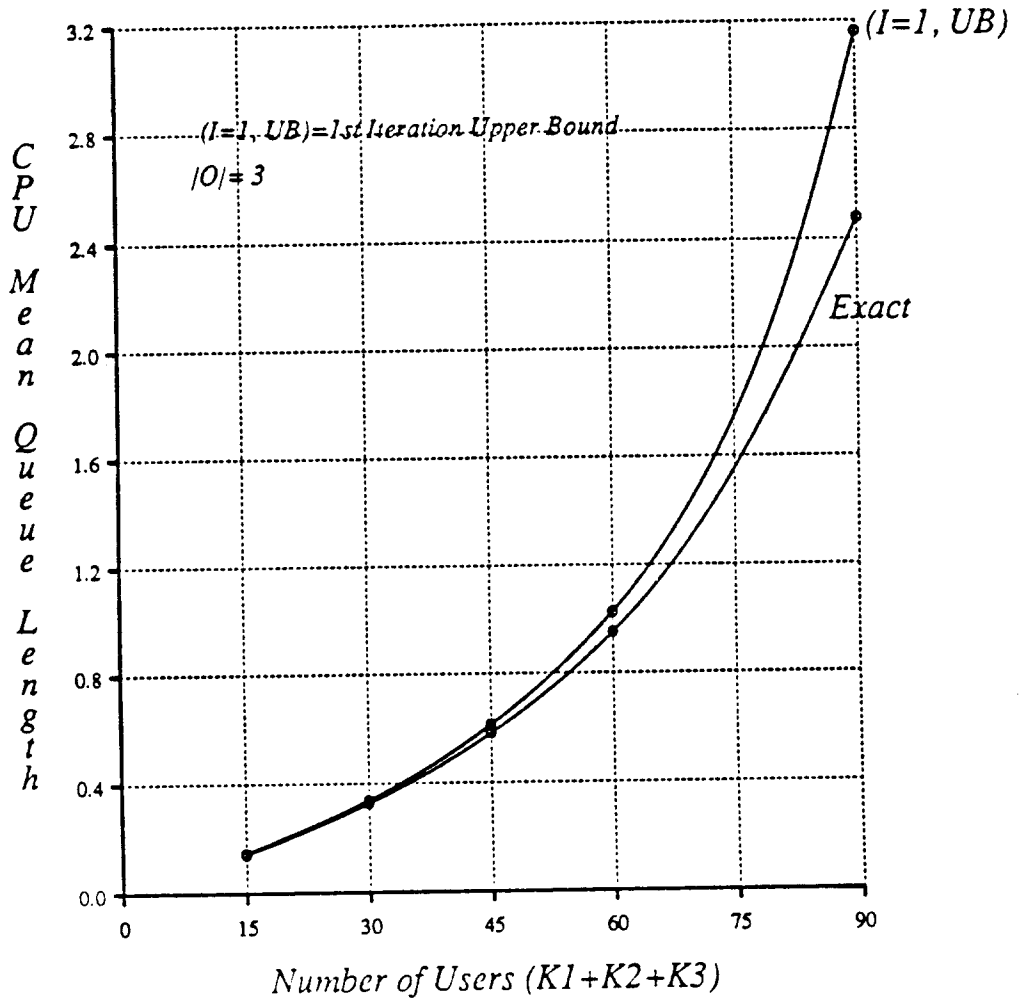
CPU Mean Q-Legth vs. Number of Users
(I=1, UB)
(I=1, UB)=1st Iteration Upper Bound
|O|= 3
Exact
CPU Mean Queue Length
3.2
2.8
2.4
2.0
1.6
1.2
0.8
0.4
0.0
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.25

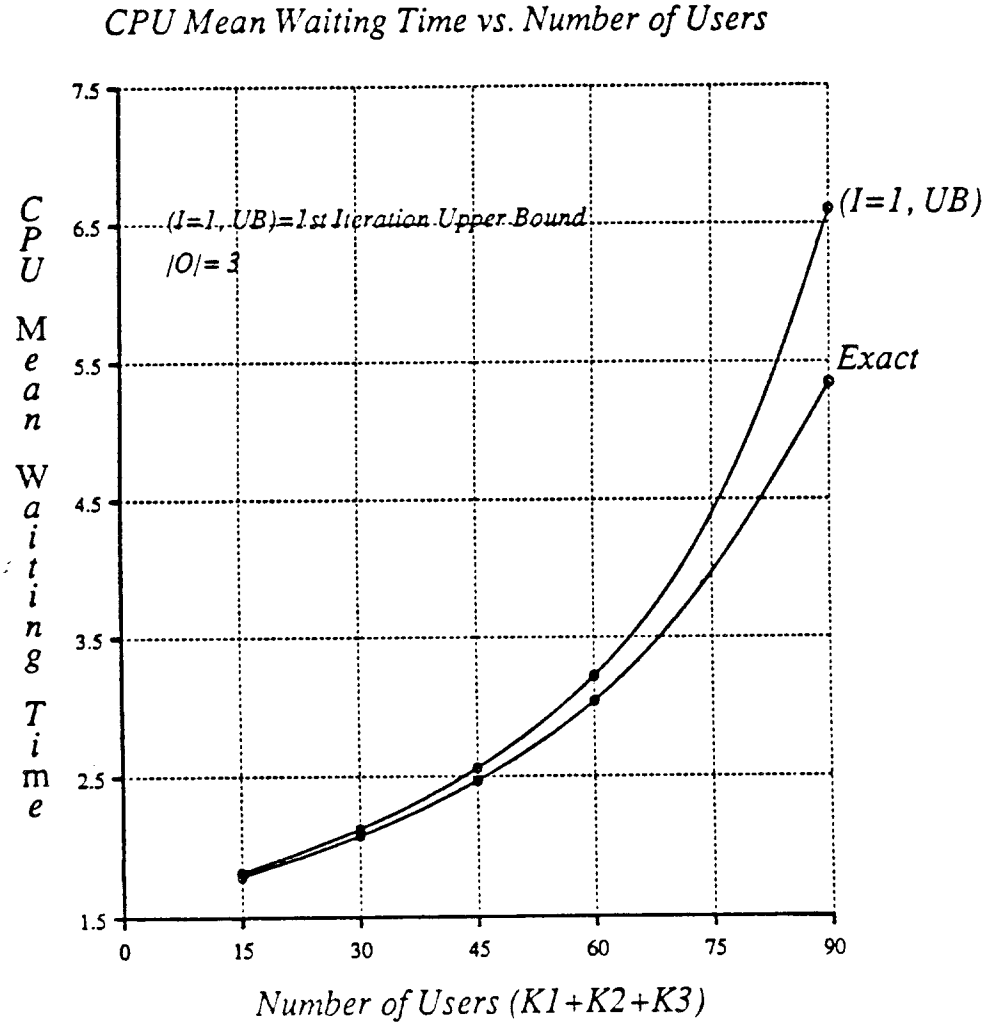
CPU Mean Waiting Time vs. Number of Users
CPU Mean Waiting Time
7.5
6.5
5.5
4.5
3.5
2.5
1.5
(I=1., UB)=1st Iteration Upper Bound
/O/= 3
(I=1, UB)
Exact
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.26

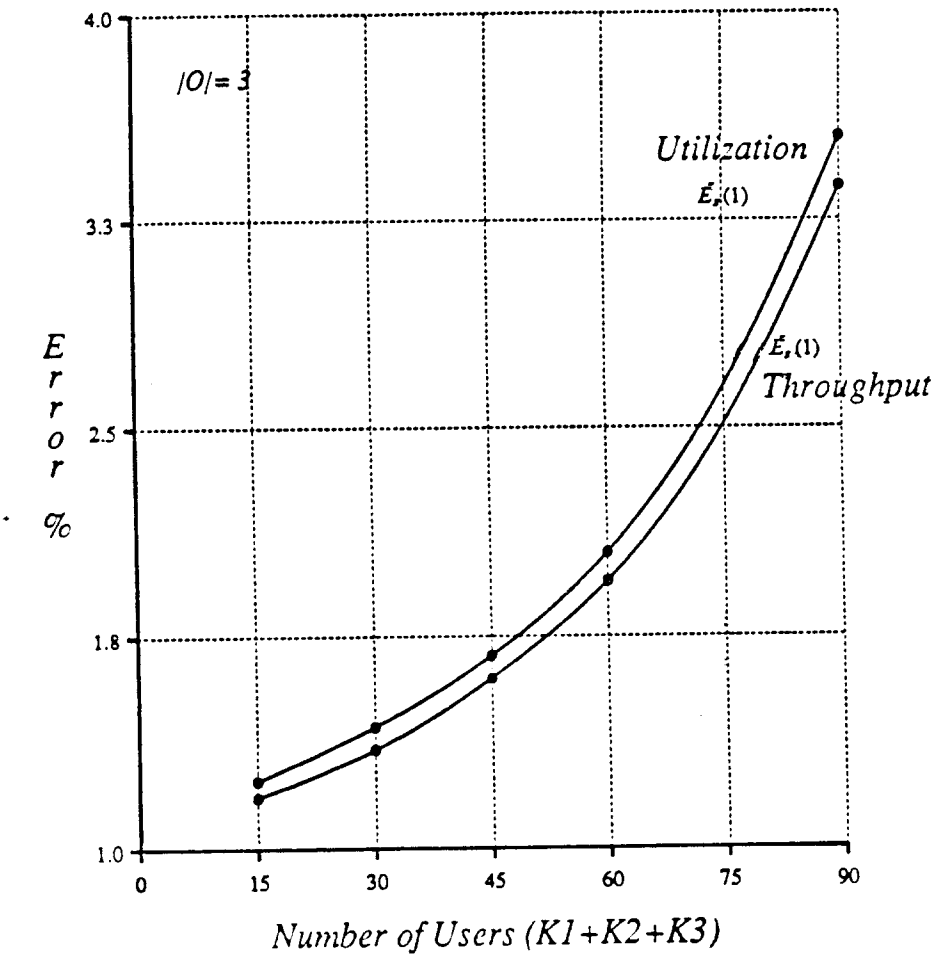
Error vs. Number of Users
4.0
3.3
2.5
1.8
1.0
Error %
|O|= 3
Utilization
$\acute{E}_r(1)$
$\acute{E}_t(1)$
Throughput
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.27

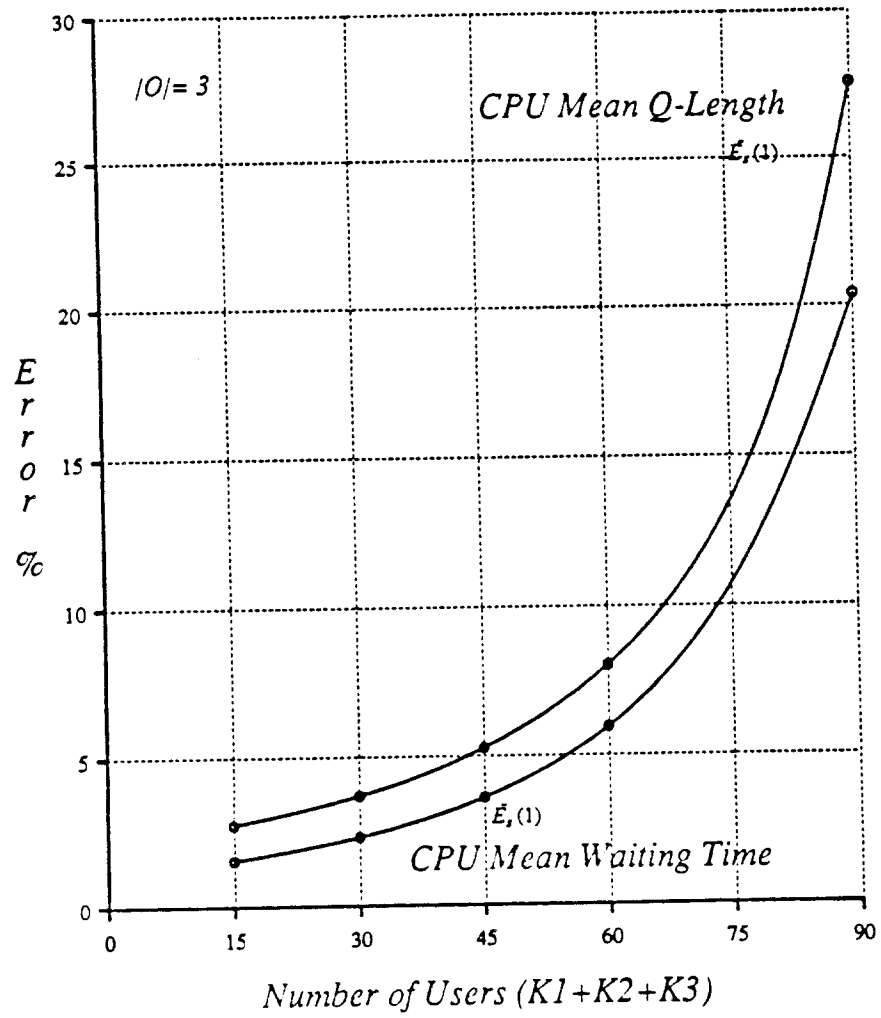
Error vs. Number of Users
|O|= 3
CPU Mean Q-Length
Ē,(1)
Ē,(1)
CPU Mean Waiting Time
Error %
30
25
20
15
10
5
0
0
15
30
45
60
75
90
Number of Users (K1+K2+K3)

Figure 5.28

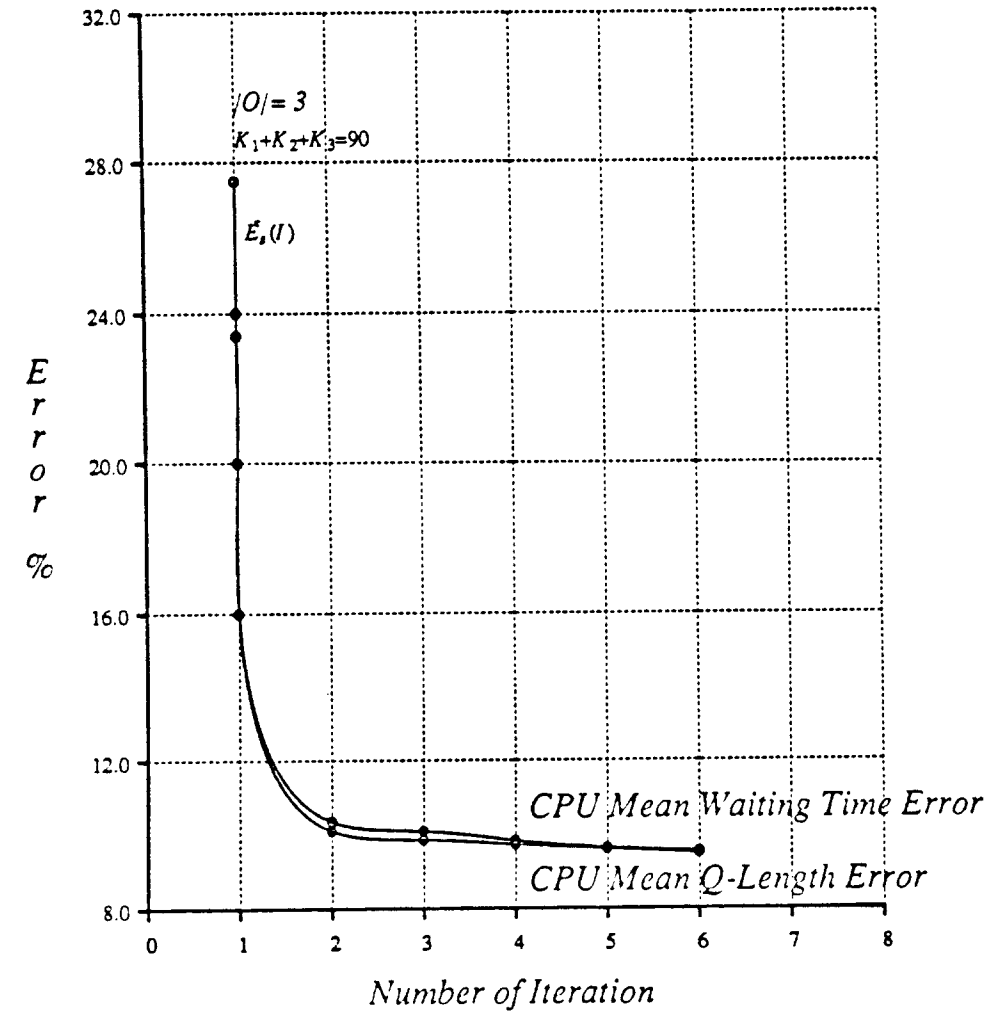
Error vs. Number of Iterations
32.0
28.0
24.0
20.0
16.0
12.0
8.0
Error %
|O|= 3
K1+K2+K3=90
Ê1(l)
CPU Mean Waiting Time Error
CPU Mean Q-Length Error
0
1
2
3
4
5
6
7
8
Number of Iteration

Figure 5.29

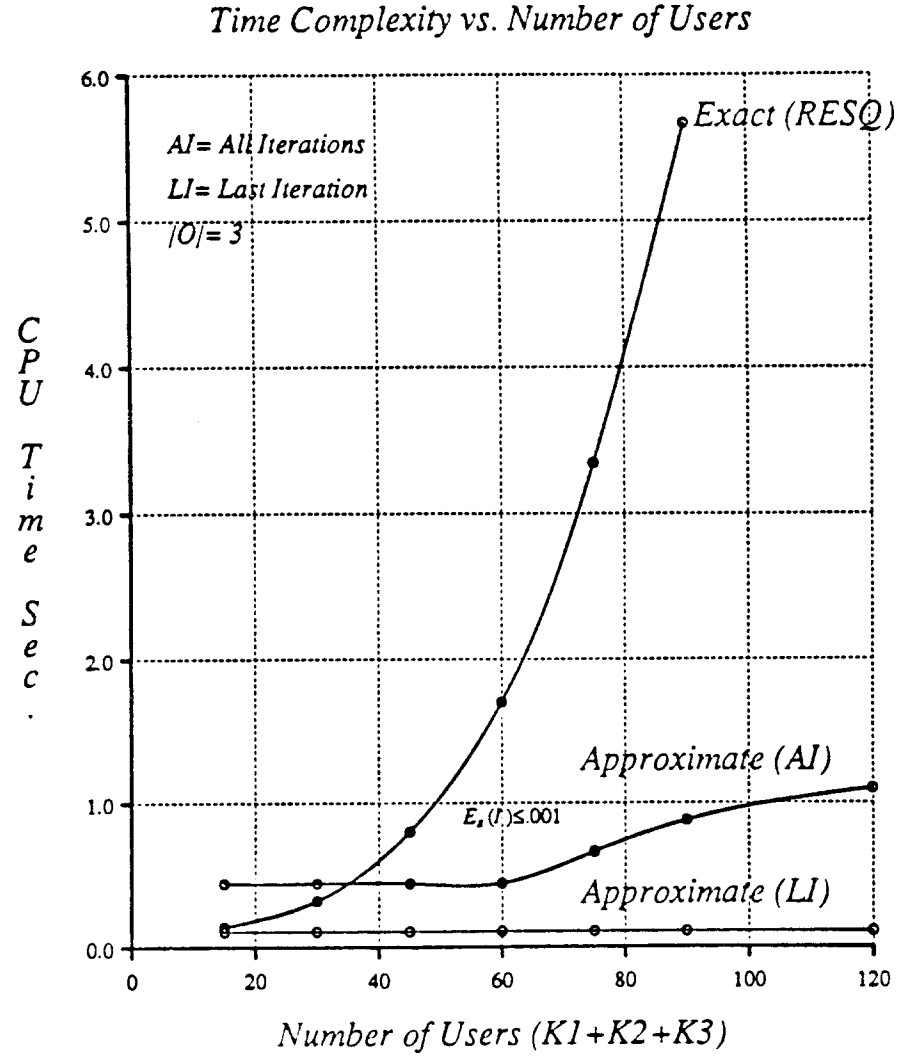
Time Complexity vs. Number of Users
AI= All Iterations
LI= Last Iteration
|O|= 3
Exact (RESQ)
Approximate (AI)
Approximate (LI)
$E_s(I) \leq .001$
CPU Time Sec.
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0
20
40
60
80
100
120
Number of Users (K1+K2+K3)

Figure 5.30

Chapter 6

Applications

6.1 Introduction

In this chapter we present two applications of the algorithms developed in chapters 4 and 5. In Sections 6.2 and 6.3 we present a decoupling algorithm for performance modeling of very large distributed systems. This algorithm is based on the network transformation based approximation proposed in chapter 5. Section 6.4 is devoted to bottleneck detection. In section 6.4.2, we propose an algorithm for bottleneck detection in multiple-class queueing networks. The algorithm uses the MVA-based approximations developed in chapter 4.

It has long been recognized that, when performance modeling of some subsystem in a larger system is carried out, an efficient and cost effective approach is to represent the rest of the system by its interactions with the particular subsystem under study. In this regard, a detailed analysis of the rest of the system is, in most cases, unnecessary. Examples of this approach are decomposition and hybrid simulation techniques. Decomposition, which is an exact technique for product form networks [Cha75], [Lav83], was briefly discussed in chapter 3. The objective in a hybrid simulation is to combine discrete event simulation and analytic models to construct efficient yet accurate system models [Mac87], [Sch78], [Mac70]. This technique has mostly been applied to single-class networks. In section 6.2 we outline an algorithm that uses the network transformation based approximate solution developed in chapter 5 and the decomposition technique to decouple submodels of a network in an accurate, efficient, and cost effective manner. This technique allows for the submodels to be analyzed or simulated independently, while taking into consideration the effects of the rest of the model on the particular submodel under study. Note that the decoupling algorithm, without the approximation introduced in chapter 5, would be practically impossible.

Another significant task in performance evaluation is that of bottleneck detection. Bottleneck detection in multiple-class network models is a difficult and in general unresolved problem. The area of bottleneck detection has many applications in practical distributed system analyses. Examples are distributed banking systems and airline reservation systems [Goo83], [Gra84], where at any moment hundreds of active users are processing thousands of transactions. In these systems, one main objective is to serve large numbers of users with the shortest possible mean response time. Hence, the identification of the bottlenecks in these systems are of critical significance to their designers. In section 6.4, using some known results and combining them with the approximate algorithms we described

in chapter 4, we propose a general algorithm for detecting bottlenecks in multichain network models. To provide further insight, we also discuss some examples.

6.2 A Decoupling Algorithm for Modeling Large Distributed Systems

Autonomy is one of the most desirable characteristics in a distributed system. In an autonomous distributed system an aggregate of sites (or a single independent site) might offer a service to the rest of the system. The owners of these sites might be interested in determining the effects of the outside requests for the particular service they are offering on the performance of their subsystem. Several sites, for example, might participate in a load sharing scheme among themselves and with other sites. Let us assume that the main objective of a set of sites *S1* is to achieve minimum response time in a load sharing scheme. Furthermore, assume that *S1* is also accepting jobs from other sites not in *S1*. As far as *S1* is concerned, the important parameters for characterizing the outside world are the rates of the requests reaching *S1*. In this case a detailed model of the outside world is not essential. Hence, in a modeling approach, the outside requests may be represented by open sources. This was the motivation behind a queueing network model that de-Souza-e-Silva and Gerla [Sil84] proposed recently for the study of an optimal load balancing algorithm in the LOCUS distributed system. They considered a QN model of a multiple site LOCUS distributed system (with a single processor at each site) connected by a local area network. The local users in each site were represented by closed chains, whereas the jobs arriving from other sites were represented by open sources. The objective was to minimize the overall average delay for all the jobs (with respect to the arrival rates for the open chains). The decision to represent the outside load at each site by open sources made the solution of the problem manageable.

However, when modeling large distributed systems, an interactive user is usually represented by a job in a closed chain. The overall model is therefore a large queuing model where users at each terminal originate jobs to the local or remote sites. It is natural, when studying the performance of a cluster of sites, to represent the effect of other sites by sources, thereby eliminating the need to solve or optimize the models of the whole system, which may be very large. This was also the fundamental motivation behind decomposition techniques, where, as we discussed in chapter 3, a subnetwork is represented by a single load dependent service center whose interaction with the rest of the system is considered. However, the successful application of decomposition requires that the closed subnetwork be solved for all possible population vectors in the subnetwork for a given population vector of the original network.

The exact solution of this subnetwork is often impractical or impossible, since it may easily exceed the available resources, when we have multichain networks. A second problem is that the number of possible population vectors in the subnetwork for a given total population (the number of times the subnetwork is to solved) may be very large. These two obstacles severely limit the application of decomposition to multichain product form networks. To remove them, we shall outline a decoupling algorithm that allows for the replacement of the rest of the network by open sources. This replacement would be exact by the decomposition (Norton's) theorem if we had the exact values of the arrival rates for the open chains. To make this methodology clear, we introduce the algorithm and an example in the next section.

6.3 The Decoupling Algorithm

We describe the algorithm by applying it to the example model shown in Figure 6.2. Figure 6.2 is a two site model of the LAN-based distributed system shown in Figure 2.1. We have chosen as example model one that is small enough to be solved exactly. Each site in the system modeled in Figure 6.2 consists of a CPU, two disk I/O subsystems, and a number of user terminals. The workload is represented by four chains, i.e., two chains per site (chains 1 and 2 for site1, and chains 3 and 4 for site2). In each site, one chain is local, that is, none of the jobs in that chain ever leave their site. The jobs in the second chain at each site are remote; they come from the other sites and use not only the local resources, but also the resources of their original site as well as the network. In each chain that interacts with another site, different classes are defined to distinguish the jobs going to the remote site

and the responses returning (this level of detail is usually introduced for the study of routing [Rei79]). In our example, this requires the introduction of four different classes of jobs at each site. Here, it is assumed that all four chains of the model are initially closed chains. The parameters of the model are given in Figure 6.4. The local area network is so modeled that the product form property of the overall model is preserved. Our objective now is to decouple each site from the model and to form two approximate submodels that can be independently solved (analytically) or simulated. We achieve this objective in the following way.

(1) The chains that involve two or more sites are identified.

(2) These chains are replaced by open chains using the iterative algorithm discussed in section 5.5.

(3) The balance equations for the resulting mixed network are solved to determine the throughput rates for the open chains (see (6.3.3)-(6.3.8)).

(4) Each class of jobs in the open chains that are arriving at each site from another site is replaced by an external Poisson source.

(5) The resulting approximate network is solved analytically or simulated.

The accuracy of the algorithm depends on that of the approximation carried out in the second step above. To evaluate the accuracy of the algorithm, exact and approximate solutions were obtained using RESQ [Sau82], and compared with each other. For each performance measure, the maximum errors in that performance measure obtained by solving the approximate network were evaluated. The convergence rate of the iterative algorithm in (2) was also examined. Note that this example is a complement to the examples considered in chapter 5.

Figure 6.5 shows the mean queue length of CPU1 versus the total number of users (that is, $K_1+K_2+K_3+K_4$). Since CPU1 is the most utilized server in this model, the maximum errors affecting performance measures will be at CPU1. Note that we have chosen $K_1=K_2=K_3=K_4$ throughout. The queue length based error used as a stopping rule is defined by:

$$E_s(I) = E_s(\bar{N}_s^{(I)}, \underline{N}_s^{(I)}) = \max_s \frac{(\bar{N}_s^{(I)} - \underline{N}_s^{(I)})}{\underline{N}_s^{(I)}}. \tag{6.3.1}$$

The stopping rule bound is $E_s(I) \leq .001$. Figure 6.6 shows the upper-bound error curves for the first iteration and the case of small stopping rule bound. The upper-bound error is defined by:

$$\bar{E}_s(I) = \max_s \frac{|N_s(\bar{\mathbf{K}}) - N_s(\bar{\Lambda}_o^{(I)}, \bar{\mathbf{K}}_c)|}{N_s(\bar{\mathbf{K}})}. \tag{6.3.2}$$

The maximum number of iterations for the case of small stopping rule bound was 6 (for population vector (12,12,12,12)). We should point out here that RESQ2 requires more than 4 Mbytes of memory to solve the model exactly for a population vector larger than (12,12,12,12), while it solves the approximate model up to (20,20,20,20) with only .96 Mbytes. We use the CPU execution time required by RESQ to compare the time complexities of solving the exact and the approximate models. The processor used to solve the networks in all cases was an IBM 3090/200. Figure 6.7 shows the execution times needed for the exact and approximate solutions to compute utilization, throughput, mean queue length, and mean waiting time for all the chains at all the servers. The curve designated as Approx. represent the execution time of the last iteration. The curve of execution times for all iterations is designated as Approximate (AI). It must be pointed out here that this curve should actually be lower (slower slope), since in the intermediate iterations we do not need to solve for all the performance measures. Even without eliminating these unnecessary computations, the improvement in execution times with respect to those required for an exact solution is evident.

Figure 6.8 shows the errors in the CPU1 mean queue length and the CPU1 mean waiting time as functions of the number of iterations. This figure shows clearly how fast and drastic an improvement

can be realized even with a small increase in the number of iterations. This figure also shows that the iterative algorithm converges very rapidly even when the error bounds are quite acceptably small.

We describe below the decoupling step for the first site. Observe that, of the five classes of jobs at CPU1 (see Figure 6.4 for a description of the model and a definition of the different classes and their routings in the network), class C11 jobs belong to the local closed chain of site 1, and the jobs of classes C14C and C14G originate from site2. Class C12G jobs leave site1 for site2 and class C12C jobs arrive from site2 to site1. We start by evaluating the arrival rates of the external sources for the jobs of classes C14C, C14G, and C12C. Let $\lambda_{o,rs}$ be the external arrival rate of open chain r at service center s. The following relationship holds for the external Poisson arrival rate vector components:

$$\lambda_{o,r} = \sum_{s \varepsilon S_o(r)} \lambda_{o,rs}, \quad r \varepsilon O, \tag{6.3.3}$$

where $p_{ij,rs}$ is the probability that a class i customer, after completing service at service center j, will move to service center s while becoming a customer of type r.

For the open chains, the following balance equation holds:

$$\theta^o_{rs} = p_{o,rs} + \sum_{i \varepsilon R_o(j),j} \theta^o_{ij} p_{ij,rs}, \quad i,r \varepsilon O, \quad j,s=1,2,...,M. \tag{6.3.4}$$

In equation (6.3.4) we assume that:

$$\sum_{s \varepsilon S_o(r)} p_{o,rs} = 1 \tag{6.3.5}$$

It can be shown that the set of equations (6.3.4) always has a unique solution.

The rates $\lambda_{o,rs}$ and the total external arrival rate λ are related by:

$$\lambda = \sum_{i \varepsilon R_o(s),s} \lambda_{o,is}. \tag{6.3.6}$$

If we let $p_{o,rs}$ be the probability of an external class r job arrival at service center s, then $p_{o,rs}$ is related to $\lambda_{o,rs}$ in the following way:

$$p_{o,rs} = \frac{\lambda_{o,rs}}{\sum_{i \varepsilon R_o(s),s} \lambda_{o,is}}, \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{6.3.7}$$

The mean throughput rates for the open chains can then be found using the following set of equations (for a proof of this relationship see [Gel80]):

$$\lambda^o_{rs} = \theta^o_{rs} \sum_{i \varepsilon R_o(s),s} \lambda_{o,is}, \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{6.3.8}$$

Using (6.3.4) and the model of Figure 6.3, we can evaluate the visit ratios in site1. Throughput rates are computed by solving the approximate two site network model. We can then use equation (6.3.6) to determine the total external rate, and equation (6.3.8) to evaluate the external rates for all classes of jobs in the site1 model. To evaluate these values for our model, let us denote by 1, 2, and 3 the jobs of classes C14C, C14G, and C12C, respectively. Assume that in site1 CPU1, Disk1, Disk2 are service centers denoted by 1, 2, and 3, respectively. Then solving (6.3.4) results in the following set of linear equations:

$\theta f_1 = 1 + .22\theta f_2 + .22\theta f_3$,

$\theta f_2 = .5\theta f_1$,

$\theta f_{21} = .5\theta f_1 + .78\theta f_2$,

$\theta f_3 = .5\theta f_1$.

Once the values of the visit ratios are known, we can use equation (6.3.8) to evaluate the external arrival rates at site1. The end result of this algorithm is the decoupling of each site. The analytic solution or the simulation of the submodels can now be carried out independently with great efficiency. Since the step in which the open source rates are evaluated is exact, this is in fact an exact decomposition. The only error is due to the approximate solution of the original network, i.e., the replacement of the closed chains by open chains. However, by the same argument made in chapter 5, since the approximate solution is asymptotically correct, the whole algorithm also produces an asymptotically correct solution. We should recall that one of the applications of decomposition has been in the area of simultaneous resource possession (such as memory partitioning) in small system models. Therefore, the decoupling algorithm can be used for the same purpose and for large networks where the decomposition alone is often impractical. The decoupling algorithm can be applied to models of multinode internetwork-based distributed systems (such as those in Figure 5.1) where each node may be a LAN-based distributed system (Figure 5.2).

6.4 Bottleneck Detection

One of the most important objectives for the application of performance analysis is the detection and elimination of bottlenecks, or at the least the reduction of their effects. A bottleneck may cause severe degradation in a system's performance. Unlike what happens in open queueing networks models, the bottlenecks of a closed multiclass network model cannot be found by inspection of the parameters. In certain classes of networks, once the bottlenecks are identified, the asymptotic throughput rates can be estimated from the knowledge of the bottlenecks. The detection of bottlenecks in queueing network models allows the system analyst to compare different configurations and choose the optimum, or to see how the enhancement of different elements of the system can alleviate its saturation, and how sensitive this improvement is to increases in the speeds of these elements. In this section, we apply algorithm I introduced in chapter 4 as an efficient and cost effective tool for detecting bottlenecks in multichain QN models of distributed systems. The core of the detection algorithm, using this approach, consists of the solution of a set of nonlinear equations.

6.4.1 Bottleneck Detection in Single-Class Networks

In a single class closed QN consisting of fixed rate and IS service centers with population vector $\mathbf{K}$ and $K = \sum_{i=1}^{M} K_i$, a service center m is referred to as a *bottleneck*[1] if

$$\lim_{K\to\infty} \rho_m(\mathbf{K}) = 1 \,. \tag{6.4.1}$$

Once the bottleneck center is identified, asymptotic values of the mean throughput rate, service center utilizations, and mean queue lengths are easily computed. The relationships that can be used to compute the asymptotic values of these performance measures are given below. In these relationships, using the definitions and the notation presented in chapter 3, $L_i = \theta_i s_i$, $i=1,2,...,M$, is the loading factor of service center i, and service center m is the bottleneck center. The derivations of these equations can be found in [Kle76], [Kob78].

$$\lim_{K\to\infty} \lambda(K) = \frac{1}{L_m}, \tag{6.4.2}$$

$$\lim_{K\to\infty} \rho_i(K) = \frac{L_i}{L_m} \,, \quad i=1,2,...,M \,, \tag{6.4.3}$$

$$\lim_{K\to\infty} N_i(K) = \frac{L_i}{L_m - L_i} \,, \quad i=1,2,...,M. \tag{6.4.4}$$

1 There may be more than one service center with this property.

Clearly, the service center with the largest value of the loading factor L_i , i=1,2,...,M, is the bottleneck center. Therefore, a simple technique for the detection of a bottleneck in a single class closed queueing network is to search for the service center with the largest loading factor. This technique does not require the actual solution of the queueing network, but only the evaluation of the visit ratios. An ordering of the loading factors will reveal the most utilized service center when the population of the network grows large. Obviously, the ordering among the loading factors is the same as those of the the utilizations. The computational complexity of finding the maximum loading factor is low, since this operation requires $M-1$ comparisons.

6.4.2 Bottleneck Detection in Multichain Queueing Networks

In contrast to the case of single-class queueing networks, the problem of bottleneck detection in multiple-class queueing network models is a difficult one, with many as yet unresolved questions. Before discussing the problem, let us define some notation. In multichain networks, the concept of population growth is not as clearly defined as in the case of single-class networks. Consider a multichain network with fixed rate single server and IS service centers. Let the population vector $\mathbf{K} = (K_1, K_2, \ldots, K_R)$ be such that $\mathbf{K}=K\gamma$, where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_R)$ is the population mix vector for the multichain network. The following relationships hold:

$$K = \sum_{i=1}^{R} K_i \; ; \quad \gamma_r = \frac{K_r}{K} \; ; \quad \sum_{i=1}^{R} \gamma_i = 1. \tag{6.4.5}$$

In multichain queueing network models, different population mix vectors γ induce different bottlenecks. Formally, a service center m in a multichain queueing network is considered to be a bottleneck if $\lim_{K\to\infty} \rho_m(\mathbf{K}) = 1$. If the number M of service centers is larger than the number R of classes, there may be cases where up to R service centers become bottlenecks (for a fixed γ and as $K\to\infty$) [Bal87]. Thus, in general, there are no known asymptotic values for the performance measures of multichain networks at the bottleneck centers, unlike what happens in the case of single-class networks. There is an exception to this statement that we shall discuss later. Thus, a search for bottlenecks in multichain networks can only be done by solving the network for a given population mix vector γ and for sufficiently large values of K (i.e., as $K\to\infty$). The problem, however, is that solving the network exactly for a large value of K and for a given population mix vector γ, as we have seen, is likely to be impractical and in most cases technically impossible. The approximate MVA-based algorithms we introduced in chapter 4 are essential for solving networks with large K and detecting the bottleneck centers. More importantly, the computational complexities of algorithm I and algorithm II for very large K, unlike those of the exact MVA solution, are independent of K. To clarify the discussion so far, we consider two examples. Let:

$$\Theta(R,M) = [\theta_{ij}], \quad i=1,2,...,R, \quad j=1,2,..,M; \tag{6.4.6}$$

$\Theta(R,M)$ is referred to as the *visit ratio matrix*. Similarly, assume that the mean service time matrix is denoted by:

$$S(R,M) = [s_{ij}], \quad i=1,2,...,R, \quad j=1,2,..,M. \tag{6.4.7}$$

Consider the following visit ratio matrix:

$$\Theta(R,M) = \begin{bmatrix} .445 & .188 & .188 & .219 \\ .400 & .145 & .137 & .317 \\ .364 & .223 & .165 & .248 \end{bmatrix}.$$

This visit ratio matrix represents a general multichain network (the routing probabilities can be chosen arbitrarily as long as they satisfy the flow balance equations) with four service centers and three

classes of jobs. Let us also assume that the service demand matrix is given by:

$$S(R,M) = \begin{bmatrix} 8.989 & 10.638 & 10.638 & 9.132 \\ 10.000 & 13.793 & 14.599 & 6.309 \\ 10.989 & 8.969 & 12.121 & 8.065 \end{bmatrix}$$

Solving the network by algorithm I given in chapter 4, and assuming a population mix vector with $\gamma_1 = \gamma_2 = \cdots = \gamma_R$, reveals that service center 1 is indeed the only bottleneck in this network model. It was not possible to detect the bottleneck using exact solution techniques for this example, since the storage requirement for an exact solution exceeds the available memory of the processor used to solve the model. As a second example, consider the same visit ratio matrix, but a different mean service demand matrix:

$$S(R,M) = \begin{bmatrix} 15.730 & 10.630 & 10.638 & 9.132 \\ 5.000 & 13.793 & 19.197 & 6.309 \\ 8.969 & 22.422 & 12.121 & 8.065 \end{bmatrix} .$$

In this example, all the first three service centers become bottlenecks for the same population mix considered in the first example.

The first example is a special case in which asymptotic values of the performance measures can be computed. Balbo and Serazzi [Bal87] have shown that, if an ordering exists among the loading factors in a multichain queueing network containing fixed rate single server and IS service centers such that, for some service center m,

$$L_{rm} > L_{rj} \quad \text{for } all \ r=1,2,...,R, \ \ j=1,2,...,M, \text{ and } \ j \neq m, \tag{6.4.8}$$

then center m is the bottleneck center. Furthermore, the most heavily loaded center remains the bottleneck no matter what the population mix vector γ is. The following relationships for the asymptotic values of utilizations, mean throughput rates, and mean queue lengths hold in this case [Bal87]:

$$\rho_{rj}^{a} = \lim_{K\to\infty} \rho_{ij}(\mathbf{K}) = \gamma_r \frac{L_{rj}}{L_{rm}}, \quad r=1,2,...,R, \quad j=1,2,...,M, \tag{6.4.9}$$

$$\rho_{rm}^{a} = \lim_{K\to\infty} \rho_{rm}(\mathbf{K}) = \gamma_r, \ r=1,2,...,R, \tag{6.4.10}$$

$$\lambda_r^{a} = \lim_{K\to\infty} \lambda_r(\mathbf{K}) = \gamma_r \frac{1}{L_{rm}}, \quad r=1,2,...,R, \tag{6.4.11}$$

$$N_{rj}^{a} = \lim_{K\to\infty} N_{rj}(\mathbf{K}) = \frac{\gamma_r L_{rj}}{L_{rm}(1 - \sum_{i=1}^{R} \rho_{ij}(\mathbf{K}))}, \ r=1,2,...,R, \ j=1,2,...,M . \tag{6.4.12}$$

It is interesting to observe that the existence of an ordering among the loading factors of a service center in a multichain queueing network implies the existence of the same ordering among the per class utilizations and mean queue lengths as well. This can easily be shown using mean value analysis. From the MVA equations given in (2.2.3) for non IS service centers, we can write:

$$\sum_{i=1}^{M} N_{ri}(\mathbf{K}) = \sum_{i=1}^{M} \lambda_{ri} s_{ri} (1 + N_i(\mathbf{K}-e_r)) = K_r , \quad r=1,2,...,R. \tag{6.4.13}$$

We also know that:

$$\frac{\theta_{ri}}{\theta_{rj}} = \frac{\lambda_{ri}(\mathbf{K})}{\lambda_{rj}(\mathbf{K})}, \quad r=1,2,...,R, \quad i,j=1,2,...,M . \tag{6.4.14}$$

Substituting (6.4.14) into (6.4.13) will produce:

$$\rho_{rj}(\mathbf{K}) = \frac{K_r L_{rj}}{\sum_{i=1}^{M} L_{ri} s_{ri} (1 + N_i(\mathbf{K} - e_r))}, \quad r=1,2,...,R, \quad j=1,2,...,M, \tag{6.4.15}$$

which clearly shows that the same ordering exists for the per chain utilization values, i.e., the existence of relationship (6.4.8) implies that the following relationship also holds:

$$\rho_{rm} > \rho_{rj} \text{ for } all \quad r=1,2,...,R, \quad j=1,2,...,M, \text{ and } j \neq m. \tag{6.4.16}$$

Similarly, we can substitute the values given in equation (6.4.15) for ρ_{rj} into the MVA equations to get the following relationship for the per class mean queue length values:

$$N_{rj}(\mathbf{K}) = \frac{K_r L_{rj} (1 + N_s(\mathbf{K} - e_r))}{\sum_{i=1}^{M} L_{ri} s_{ri} (1 + N_i(\mathbf{K} - e_r))}, \quad r=1,2,...,R, \quad j=1,2,...,M. \tag{6.4.17}$$

Equation (6.4.16) and the ordering assumption on loading factors imply that:

$$N_{rm}(\mathbf{K}) > N_{rj}(\mathbf{K}) \text{ ,for } all \quad r=1,2,...,R, \quad j=1,2,...,M. \tag{6.4.18}$$

The asymptotic values of the performance measures in a load independent network with only two classes and two bottleneck centers has been derived in [Bal87]. But, as was noted earlier, no exact technique for multiple bottleneck detection and the evaluation of the asymptotic values of the performance measures in multichain networks is available.

We now introduce a general algorithm for bottleneck detection in multiple-class queueing networks. The most important step in this algorithm is the application of the MVA-based approximations described in chapter 4.

(1) Evaluate the visit ratios for the given network.

(2) Check if an ordering such as that defined in (6.4.8) exists among the service centers.

(3) If an ordering exists among the loading factors of the queueing network, then the asymptotic values of the performance measures can be obtained using equations (6.4.9) - (6.4.12).

(4) If no ordering exists, use algorithm I (or algorithm II) introduced in chapter 4 to solve the network for a very large value of K, and determine the bottleneck centers.

(5) Evaluate the asymptotic values of the performance measures using algorithm I or algorithm II.

Note that the most crucial steps in the above algorithm are steps (4) and (5). Without the approximate MVA-based algorithms, bottleneck detection is not possible in general.

6.5 Conclusions

In this chapter a decoupling algorithm for analytic modeling and simulation of large queueing network models was proposed. The algorithm applies to multiclass queueing networks with fixed rate single server and IS service centers. It has potential applications in the modeling of nonproduct form networks, such as those in which there is simultaneous resource possession. This algorithm is many orders of magnitude less expensive than exact methods. The solutions it produces are asymptotically correct.

Furthermore, bottleneck detection in single and multiple-class closed networks consisting of only single server and IS service centers was investigated. A general algorithm for bottleneck detection in these networks was introduced. This algorithm combines known results with algorithm I which we

introduced in chapter 4. The algorithm may be used as a cost effective and computationally efficient tool for bottleneck detection in queueing network models of very large distributed systems. Networks with load dependent or multiple server service centers were not discussed in the section on bottleneck detection. This is due to the fact that there are no universally accepted approximate solution techniques for those networks.

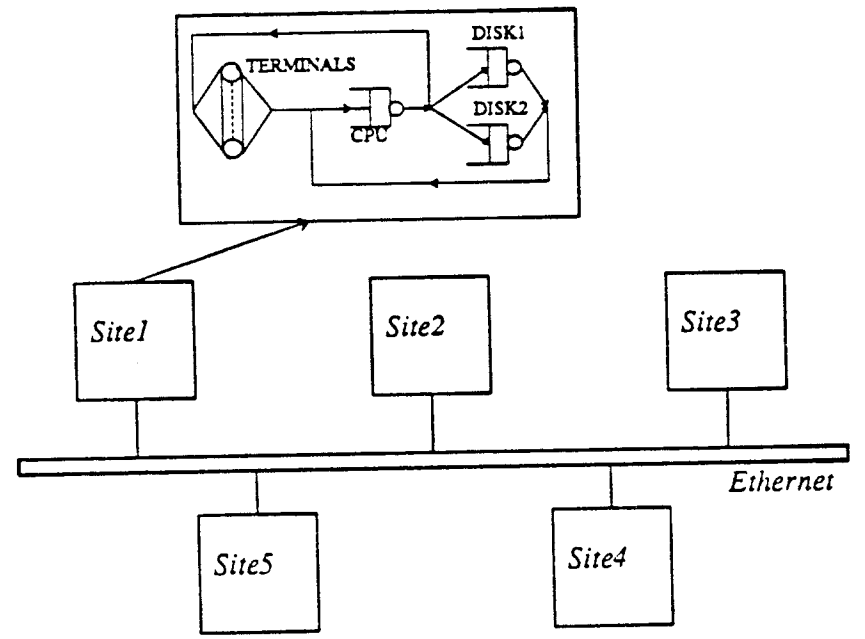

Figure 6.1 A LAN Based Distributed System Configuration

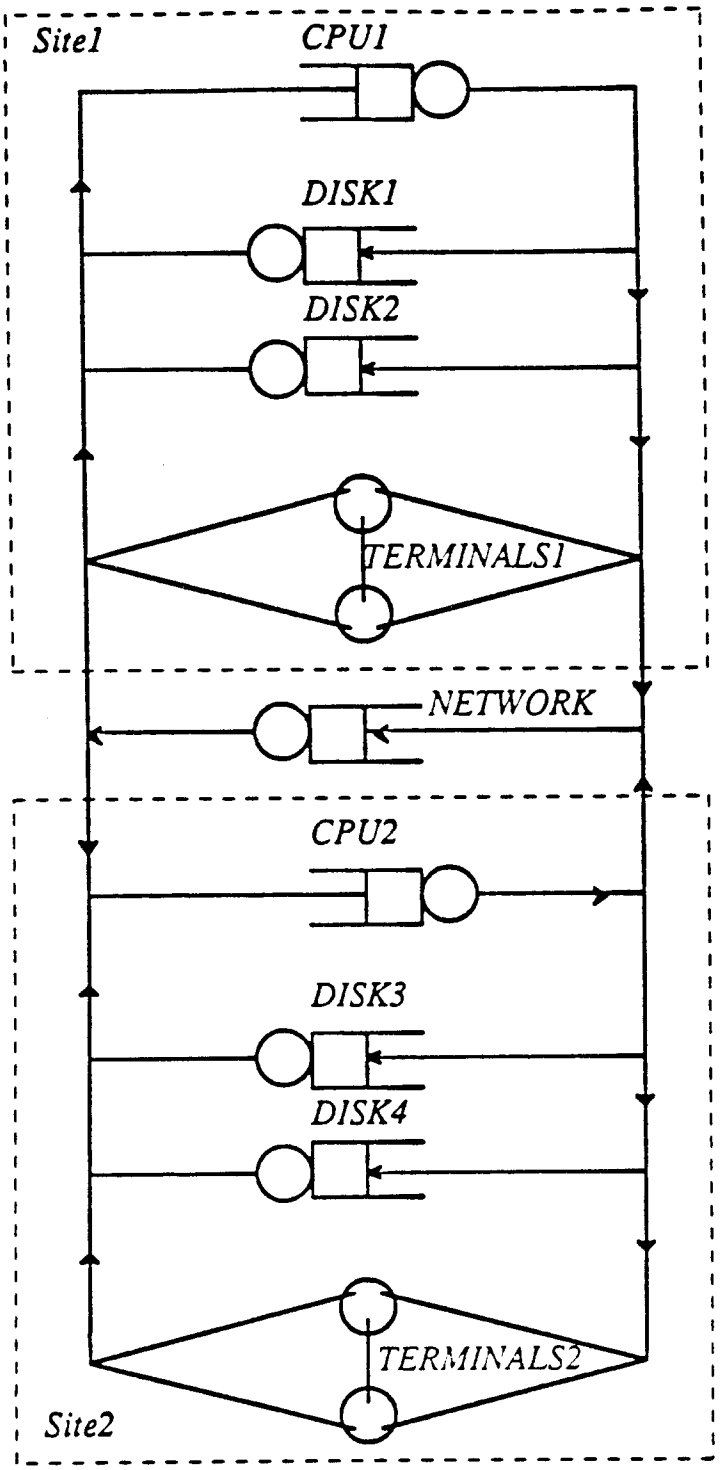

Figure 6.2 A Two Site Queueing Network Model

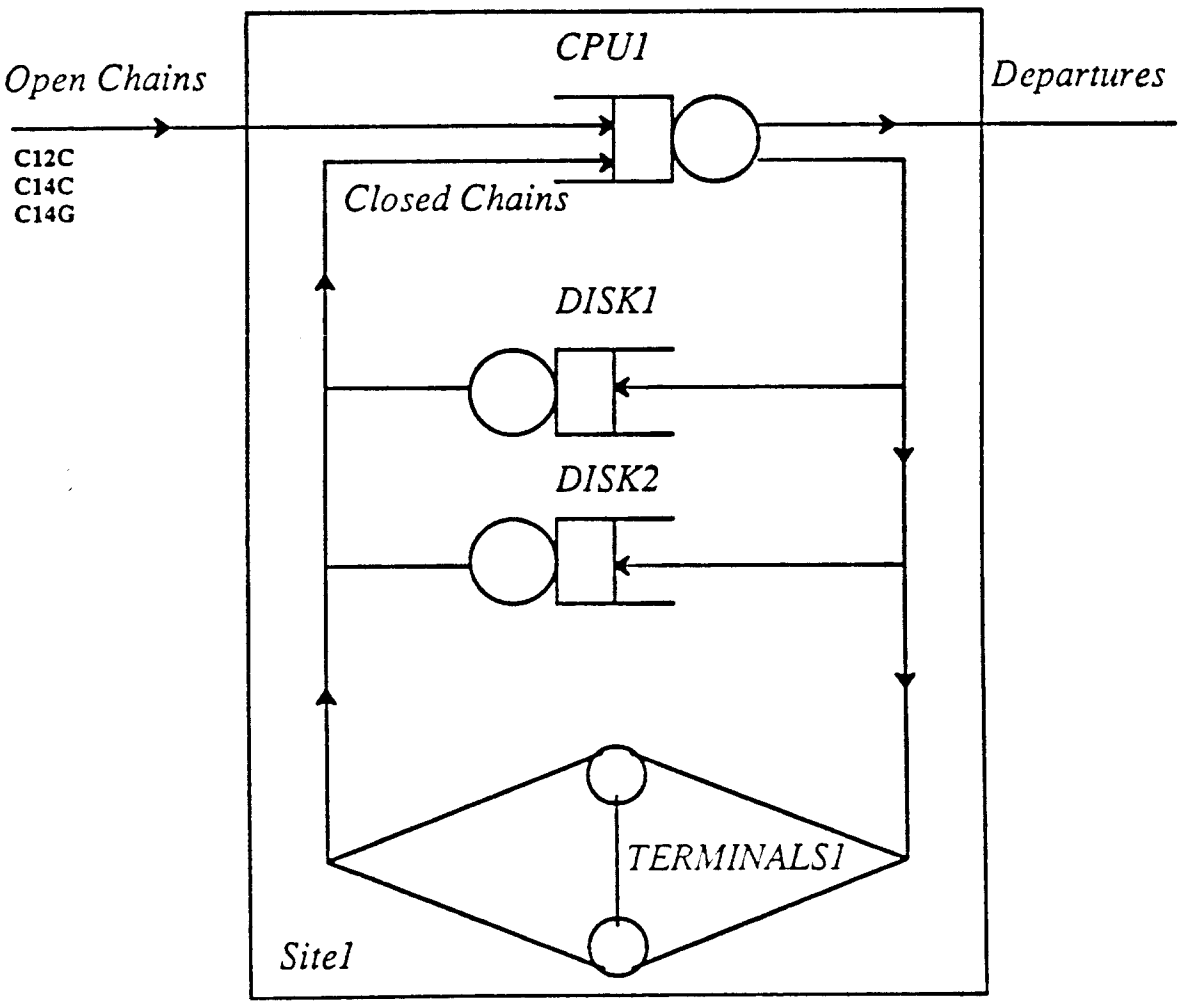

Figure 6.3 Decoupled Site1

```
MODEL:IASCLOSE
NUMERIC PARAMETERS:K1 K2 K3 K4
NUMERIC IDENTIFIERS:c1 c2 c3 c4 t1 t2 t3 t4 d1 d2
c1:2.0, c2:1.2, c3:1.0, c4:2.5, t1:150.0,
t2:250.0,t3:450.0, t4:200.0, d1:1.0, d2:.1
QUEUE:cpu1q TYPE:ps CLASS LIST:C11 C14G C14C C12G C12C
SERVICE TIMES:C1 C4 C4 C2 C2
QUEUE:cpu2q TYPE:ps CLASS LIST:C23 C24G C24C C22G C22C
SERVICE TIMES:C3 C4 C4 C2 C2
QUEUE:disk1q TYPE:fcfs CLASS LIST:D11 D14 SERVICE TIMES:d1
QUEUE:disk2q TYPE:fcfs CLASS LIST:D21 D24 SERVICE TIMES:d1
QUEUE:comq TYPE:fcfs CLASS LIST:COM2C COM2G COM4C COM4G
SERVICE TIMES:d2
QUEUE:disk3q TYPE:fcfs CLASS LIST:d32 d33 SERVICE TIMES:d1
QUEUE:disk4q TYPE:fcfs CLASS LIST:d42 d43 SERVICE TIMES:d1
QUEUE:term1q TYPE:is CLASS LIST:tm1 tm2 SERVICE TIMES:t1 t2
QUEUE:term2q TYPE:is CLASS LIST:tm3 tm4 SERVICE TIMES:t3 t4

CHAIN:chain1 TYPE:closed POPULATION:K1
tm1->c11, c11->d11 d21 tm1; .1 .1 .8 d11->c11, d21->c11

CHAIN:chain2 TYPE:closed POPULATION:k2 TM2->C12G,
C22C->D32 D42; .45 .55, D42->C22C C22G; .25 .75
D32->C22C C22G; .25 .75, C22G->COM2C->C12C->TM2
C12G->COM2G->C22C

CHAIN:CHAIN3, YPE:CLOSED, POPULATION:K3
TM3->C23, C23->D33 D43 TM3; .12 .12 .76
D33->C23 :D43->C23

CHAIN:CHAIN4, TYPE:CLOSED, POPULATION:K4
TM4->C24G->COM4G->C14C, C14C->D14 D24; .5 .5
D14->C14C C14G; .22 .78, D24->C14C C14G; .22 .78
C14G->COM4C->C24C->TM4
```

Figure 6.4 Parameters of the QN Model Shown in Figure 6.2.

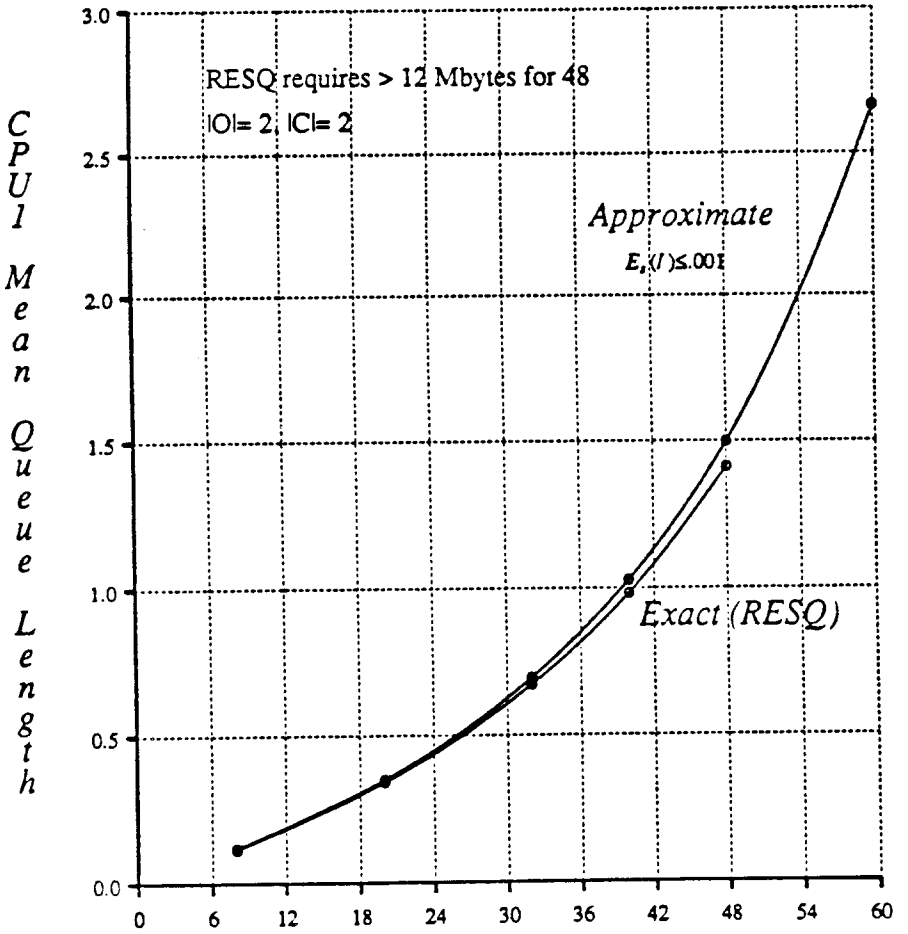
CPU1 Q-Length vs. Number of Users
RESQ requires > 12 Mbytes for 48
|O|= 2, |C|= 2
Approximate
Exact (RESQ)
CPU1 Mean Queue Length
Number of Users (K1+K2+K3+K4)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
6
12
18
24
30
36
42
48
54
60

Figure 6.5

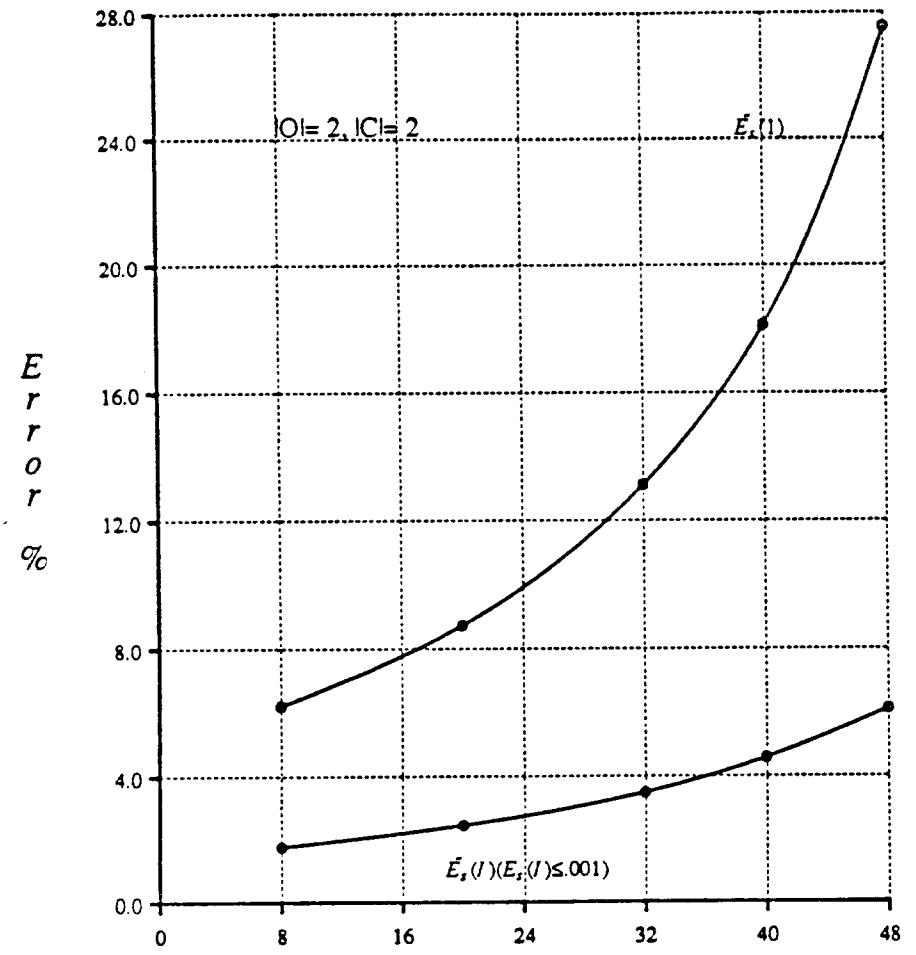
CPU1 Q-Length Error vs. Number of Users
28.0
24.0
20.0
16.0
12.0
8.0
4.0
0.0
Error %
|O|= 2, |C|= 2
$\bar{E}_s(1)$
$\bar{E}_s(J)(E_s(J)\leq.001)$
0
8
16
24
32
40
48
Number of Users (K1+K2+K3+K4)

Figure 6.6

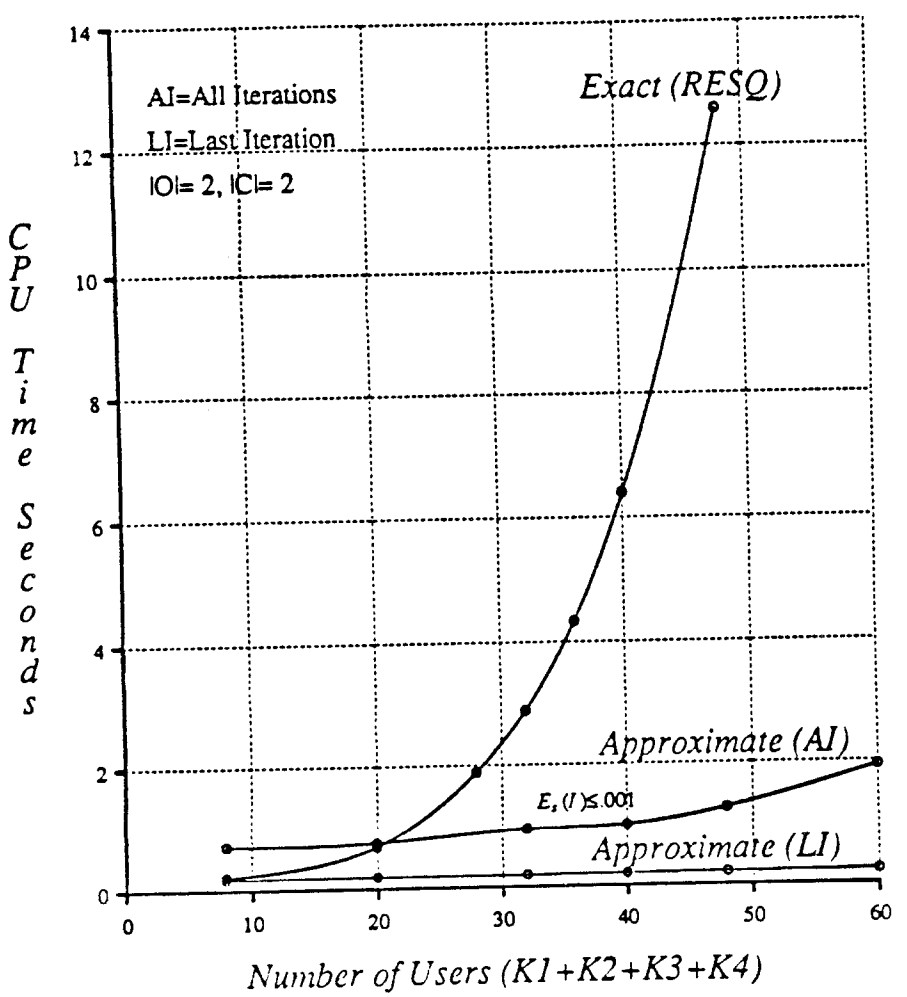
Time Complexity vs. Number of Users
AI=All Iterations
LI=Last Iteration
|O|= 2, |C|= 2
Exact (RESQ)
Approximate (AI)
E, (I)≤.001
Approximate (LI)
CPU Time Seconds
Number of Users (K1+K2+K3+K4)
0
2
4
6
8
10
12
14
0
10
20
30
40
50
60

Figure 6.7

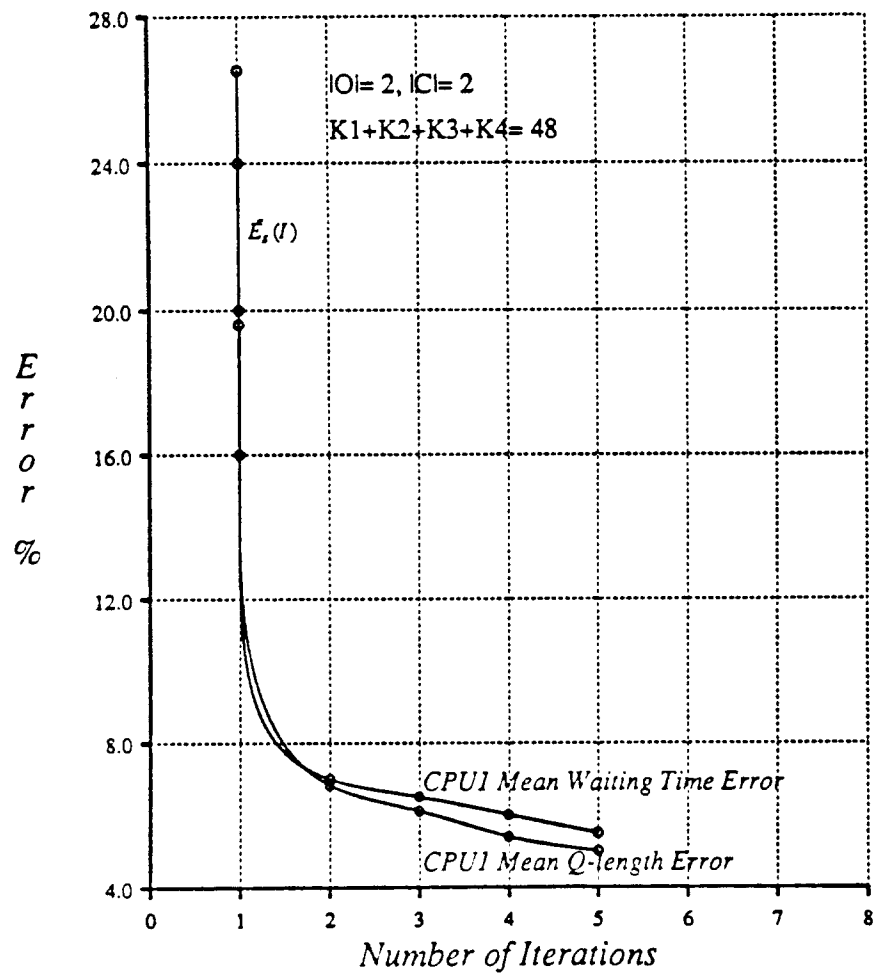
Error vs. Number of Iterations
28.0
24.0
20.0
16.0
12.0
8.0
4.0
Error %
|O|= 2, |C|= 2
K1+K2+K3+K4= 48
$\hat{E}_s(I)$
CPU1 Mean Waiting Time Error
CPU1 Mean Q-length Error
0
1
2
3
4
5
6
7
8
Number of Iterations

Figure 6.8

Chapter 7

Concluding Remarks

7.1 Summary

Performance modeling is one of the principle techniques available for the design, development, and tuning of computer systems. Queueing networks have become an indispensable part of these processes. A key challenge in computer performance modeling is the development of new approximate methods and tools that keep the pace with the emerging new generations of systems brought on by the advances in technology. In light of the growing complexity of these systems, approximate analytic modeling techniques are becoming increasingly important. Approximation will play a vital role in meeting the challenges of future complex distributed processing systems.

The main points and contributions of this dissertation are summarized below:

- In chapter 2 we started with a brief study of exact solution techniques for multichain QNs. By examining the computational complexities of these techniques, we established the need for approximate algorithms for solution of large multichain queueing networks. The MVA equations and the computational complexities for the solution of mixed and open multichain networks were then derived.

- Realizing the need for the development of approximate techniques for the solution of large queueing networks, we carried out a comprehensive overview of the existing approximate techniques. We observed that most of the existing MVA-based approximate algorithms lacked all the three main objectives outlined in section 3.2. Computationally efficient and accurate hierarchical techniques are almost nonexistent. We concluded that there is much urgent need for improving, extending, and refining the existing algorithms as well as for introducing new ones. This was the fundamental motivation behind the work undertaken in this dissertation.

- In chapter 4, we proposed a new MVA-based approximation that applies to multichain product form queueing networks consisting only of fixed rate single server and IS service centers. The algorithm was shown to have a satisfactory accuracy, and to have the property that a unique solution in the feasible region is always guaranteed. This much desirable feature is absent from several other MVA-based approximate techniques. The accuracy and the computational complexities of the algorithm were extensively studied. The approximations we have proposed realizes the three objectives outlined in section 3.2 very adequately.

• Hierarchical approximations are among the most promising approaches to the approximate solution of queueing network models. Their most prominent feature is the gradual introduction of approximation that commensurates with the cost and desirable accuracy. This aspect is in contrast to most other approximate algorithms (MVA-based as well as asymptotic expansion) where the underlying approximation is introduced right at the beginning. In chapter 5, we introduced a hierarchical approximation based on transforming closed networks containing fixed rate single server and IS service centers into mixed network. The computational complexities of the algorithm were derived and studied. Experimental analyses were carried out to investigate the accuracy, stability, and computational costs of the algorithm. The algorithm provides solutions that are asymptotically correct. One distinct feature is its simplicity of implementation.

• Two applications of the results obtained in previous chapters were discussed in chapter 6. We used the network transformation based approximation developed in chapter 5 as a basis for an efficient and cost effective algorithm for decoupling submodels of certain queueing network models from the rest of the model. An important step in studying and improving the performance of large distributed systems consists of detecting their bottlenecks. Short of building expensive simulation models or of the actual construction of different alternative designs, modeling is a cheap first step to be taken. In chapter 6 we applied algorithm I, proposed in chapter 4, as an efficient and cost effective tool for detecting bottlenecks in multichain queueing networks. The detection algorithm, using this approach, requires the solution of a set of nonlinear equations.

It is hoped that the work presented here can be used as a platform upon which new and more powerful queueing network modeling tools can be constructed.

7.2 Directions for Future Work

• The approximate algorithm we proposed in chapter 5 applies to the chains visiting an IS service center in the network. However, as we noted in chapter 5, in the most general case some of the closed chains may not visit any of the IS service centers in the network. The transformation based approximation does not apply to these chains. Hence, the computational complexities of the solution algorithm can still exceed the available resources. The MVA-based approximations we introduced in chapter 4 (algorithm I and II) can be combined with the network transformation based approximation to solve the resulting mixed network.

• Hierarchical approximation techniques provide for a smooth tradeoff of cost and accuracy. Thus, hierarchical modeling and simulation techniques should be further exploited, refined, and extended to include more general types of product form networks; for example, those with load dependent or multiple server service centers or both.

• It is increasingly desirable that tight error bounds for approximate solution algorithms be found. Except for the asymptotic expansion algorithm, no other approximate algorithm discussed in this dissertation provides any error bound for the approximate performance measures.

• In the absence of any error bound, the accuracy of an approximation has to be verified experimentally. This requires that consistent and comprehensive frameworks and standards be developed to facilitate the validation process of approximate techniques, to identify when these approximations fail and when they are successful. The process of validation should include stress networks.

• Most of the existing approximate algorithms we have discussed in this dissertation, including those we have proposed, apply exclusively to product form queueing networks consisting only of fixed rate single server service centers and infinite server service centers. Presently, there is no generally accepted approximate methodology for product form networks with load dependent or multiple server service centers. Attempts should be made to extend these algorithms to networks including multiple server and load dependent service centers.

• Distributed computing systems modeling is in its infancy. A new generation of tools for simulation and analytic modeling of distributed systems ought to be designed and built. These tools must include a multitude of approximate techniques to facilitate the performance analysis of these systems.

References

[And87]
D. P. Anderson, D. Ferrari, P. V. Rangan and S. Y. Tzou, The DASH Project: Issues In the Design of Very Large Distributed Systems, Technical Report No. UCB/Computer Science Dpt. 87/338, University of California Berkeley, Berkeley, CA, January 1987.

[Bah87]
H. R. Bahadori, A Hierarchical Approximation Algorithm for Large Multichain Product Form Queueing Networks , UCB/Computer Science Dpt. Technical Report No. 87/382,, University of California, Berkeley, Berkeley, CA, December 1987.

[Bal77]
G. Balbo, S. C. Bruell and H. D. Schwetman, *Computational Aspects of Closed Queueing Networks with Different Customer Classes*, Department of Computer Science, Purdue University, West Lafayette, IN, July 1977.

[Bal87]
G. Balbo and G. Serazzi, Multi-Class Product Form Closed Queueing Networks Under Heavy Loading Conditions, *AFCET International Workshop on Modeling Technique and Performance Evaluation*, Paris, March 1987.

[Bar78]
Y. Bard, An Analytic Model of the VM/370 System, *IBM Journal of Research and Development 22, No. 5* (September 1978), 498-508.

[Bar79]
Y. Bard, Some Extensions to Multiclass Queueing Network Analysis, in *Performance of Computer Systems*, M. Arato,, A. Butrimenko and E. Gelenbe (editor), North Holland, Amsterdam, 1979, 51-61.

[Bar80]
Y. Bard, A Model of DASD and Multipathing, *Communication of ACM 23* (1980), 564-572.

[Bas75]
F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, Open, Closed, and Mixed Networks of Queues with Different Classes of Customers, *Journal of ACM 22, 2* (April 1975), 248-260.

[Bes84]
J. Bester, M. Gerla and G. Popek, A Dual Priority MVA Model for a Large Distributed System: LOCUS, *Performance 84*, 1984, 51-66.

[Bil86]
P. Billingsley, *Probability and Measure Theory 2nd Edition*, Wiley-Interscience, 1986.

[Bro65]
C. G. Broyden, A Class of Methods for Solving Nonlinear Simultaneous Equations, *Comp. Math.*, 1965, 577-593.

[Bru78]
S. C. Bruell, *On Single and Multiple Job Class Queueing Network Models of Computing Systems*, Ph.D. Thesis, Computer Science Department, Purdue University, December 1978.

[Bur85]
R. L. Burden and J. D. Faires, *Numerical Analysis (Third Edition)*, Prindle, Weber & Schmidt, 1985.

[Buz73]
J. P. Buzen, Computational Algorithms for Closed Queueing Networks with Exponential Servers, *Communications of the ACM 16, 9* (September 1973), 527-531.

[Cha75a]
K. M. Chandy, U. Herzog and L. S. Woo, Parametric Analysis of Queueing Networks, *IBM Journal of Research and Development 19* (1975), 43-49.

[Cha75b]
K. M. Chandy, U. Herzog, L. Woo and F. G. Palacios, Approximate Analysis of General Queueing Networks, *IBM Journal of Research and Development 19, 1* (January 1975), 43-49.

[Cha80]
K. M. Chandy and C. H. Sauer, Computational Algorithms for Product Form Queueing Networks, *Communications of the ACM 23, 10* (October 1980), 573-583.

[Cha82]
K. M. Chandy and D. Neuse, Linearizer: A Heuristic Algorithm for Queueing Network Models of Computer Systems, *Communications of the ACM 25, 2* (February 1982), 126-134.

[Cho83]
W. Chow, Approximations for Large Scale Closed Queueing Networks, *Performance Evaluation 3* (1983), 1-12.

[Cou75]
P. J. Courtois, Decomposability, Instabilities and Saturation in Multiprogramming Systems, *Communication of ACM 18 no. 7* (July 1975).

[Cou77]
P. J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.

[de-83]
de-Souza-e-Silva, S. S. Lavenberg and R. R. Muntz, A Perspective on Iterative Methods for Approximate Analysis of Closed Queueing Networks, *Proceedings of the International Workshop on Apllied Mathematics and Performance /Reliability Models of Computer/Communication Systems*, 1983, 191-210.

[Den78]
P. J. Denning and J. P. Buzen, The Operational Analysis of Queueing Network Models, *Computing Surveys 10, 3* (September 1978), 225-261.

[Eag83]
D. L. Eager and K. C. Sevcik, Performance Bound Hierarchies for Queueing Networks, *ACM Transactions on Computer Systems 1, 2* (May 1983), 99-115.

[Eag86]
D. Eager and K. C. Sevcik, Bound Hierarchies for Multiple-Class Queueing Networks, *Journal of ACM 33*, 1 (January 1986), 179-206.

[Fra73]
L. Fratta, M. Gerla and L. Kleinrock, The Flow Deviation Method-An Approach to Store-and-Forward Communication Network Design, *Networks 3* (1973).

[Gav71]
D. P. Gaver and G. S. Shedler, Multiprogramming System Performance via Diffusion Approximations, IBM Research Report, RJ-938, Yorktown Heights, New York, 1971.

[Gel80]
E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*, Academic Press, 1980.

[Gol83]
A. Goldberg, G. Popek and S. Lavenberg, A Validated Distributed System Performance Model, *Performance '83*, 1983, 251-268.

[Goo83]
J. R. Good, Experience with a Large Distributed Banking System, *Database Engineering 6 No. 2*

(June 1983), 50-56.

[Gra78]
G. S. Graham, (ed.) Special Issue: Queueing Network Models of Computer System Performance, *ACM Computing Surveys 10, 3* (September 1978).

[Gra79]
J. N. Gray, A Discussion of Distributed Systems, *IBM Research Report RJ2699*, September 1979.

[Gra84]
J. Gray, B. Good, D. Gawlick and P. Homan, One Thousand Transactions Per Second, *Tandem Technical Report 85.1*, November 1984, 22.

[Hef82]
H. Heffes, Moment Formulae for a Class of Mixed Multi-Job-Type Queueing Networks, *The Bell System Technical Journal 61, 5* (May-June 1982), 709-745.

[Ker84]
T. Kerola, The Composite Bound Method for Computing Throughput Bounds in Multiple Class Environments, Technical Report Computer Science Dpt.-Tech. Rep. 475, Purdue University, West Lafayette, IN, August 1984.

[Ker85]
T. Kerola, The Composite Bound Method for Computing Throughput Bounds in Multiple Class Environments, Ph.D. Thesis, Computer Science Dept., Purdue University, West Lafayette, IN, 1985.

[Ker86]
T. Kerola, The Composite Bound Method for Computing Throughput Bounds in Multiple Class Environments, *Performance Evaluation*, 1986, 1-9.

[Kie79]
M. G. Kienzle and K. C. Sevcik, Survey of Analytical Queueing Network Models of Computer Systems, *Proc. Conf. on Simulation, Measurement, and Modeling of Comp. Sys.* (August 1979), 113-129.

[Kob78]
H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*, Addison-Wesley, Reading, Mass., 1978.

[Kob83]
H. Kobayashi and M. Gerla, Optimal Routing in Closed Queueing Networks, *ACM Transactions on Computer Systems 1, 4* (November 1983), 294-310.

[Krz84]
A. Krzensinski and J. Greyling, Improved Linearizer Methods for Queueing Networks with Queue Dependent Centers, *Sigmetrics Performance Evaluation*, 1984, 41-51.

[Lam82]
S. S. Lam, Dynamic Scaling and Growth Behavior of Queueing Network Normalization Constants, *Journal of ACM 29, 2* (April 1982), 492-513.

[Lam83]
S. S. Lam and Y. L. Lien, A Tree Convoluted Algorithm for the Solution of Queueing Networks, *Communications of the ACM 26, 3* (March 1983), 203-215.

[Lav80]
S. S. Lavenberg, Closed Multichain Product Form Queueing Networks with Large Population Sizes, *IBM Research Report RC 8496 (#36973)*, September 1980.

[Lav83]
S. S. Lavenberg, *Computer Performance Modeling Handbook*, Academic Press, New York, 1983.

[Lit61]
J. D. C. Little, A Proof of Queueing Formula $L = \lambda W$, *Operations Research 9* (1961), 383-387.

[Mac70]
M. H. MacDougall, Computer System Simulation: An Introduction, *Computing Surveys 2,3* (1970), 191-210.

[Mac87]
M. H. MacDougall, *Simulating Computer Systems*, MIT Press, Cambridge, Massachusetts, 1987.

[McK81]
J. McKenna, D. Mitra and K. G. Ramakrishnan, A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, and Generalizations, *Bell System Technical Journal 60, No. 5* (1981), 599-641.

[McK82]
J. McKenna and D. Mitra, Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage, *Bell Systems Technical Journal 61, 5* (May -June 1982), 661-683.

[McK84]
J. McKenna and D. Mitra, Asymptotic Expansions and Integral Representations of Moments of Queue Length in Closed Markovian Networks, *Journal of Association of Computing Machinery 31, No. 2* (April 1984), 347-360.

[Mit84]
D. Mitra and J. McKenna, Some Results on Asymptotic Expansions for Closed Markovian Networks with State Dependent Service Rates, *Performance 84 E. Gelenbe 84*, Amsterdam, 1984, 377-392.

[Mor80]
J. J. More, B. S. Garbow and K. E. Hillstrom, User Guide for Minpack-1, Report No. ANL-80-74, National Argonne Laboratory, Argonne, Illinois, August 1980.

[Mun74]
R. R. Muntz and J. W. Wong, Asymptotic Properties of Closed Queueing Network Models, *Proceedings 8th Princeton Conference on Information Sciences and Systems*, Princeton, NJ, 1974, 348-352.

[Neu81]
D. Neuse and M. K. Chandy, SCAT: A heuristic Algorithm for Queueing Network Models of Computing Systems, *Performance Evaluation Review 10* (1981), 59-79.

[New71]
G. F. Newell, in *Applications of Queueing Theory*, Chapman and Hall, London, 1971.

[Ort70]
J. M. Ortega and W. C. Rheinholdt, *Iterative Solution of Nonlinear Equations with Several Variables*, Academic Press, Inc., 1970.

[Pit79]
B. Pittel, Closed Exponential Networks of Queues with Saturation: The Jackson-Type Stationary Distribution and Its Asymptotic Analysis, *Math. Operations Research 4* (1979), 357-378.

[Pow70]
M. J. D. Powell, A Hybrid Method for Nonlinear Equations, in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz (editor), 1970.

[Ram82]
K. G. Ramakrishnan and D. Mitra, An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks, *The Bell System Technical Journal 61, 10* (December 1982), 2849-2872.

[Rei74]
M. Reiser and H. Kobayashi, Accuracy of the Diffusion Approximation for Some Queueing Systems, *IBM Journal of Research and Development 18* (1974), 110-124.

[Rei75]
M. Reiser and H. Kobayashi, Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms, *IBM Journal of Research and Development*, May 1975, 283-294.

[Rei79]
M. Reiser, A queueing Network Analysis of Computer Communication Networks with Window Flow Control, *IEEE transactions on Communications Com-27, No.8* (August 1979), 1199-1209.

[Rei80]
M. Reiser and S. S. Lavenberg, Mean-Value Analysis of Closed Multichain Queueing Networks, *Journal of ACM 27, 2* (April 1980), 313-322.

[Rei81]
M. Reiser, Mean-Value Analysis and the Convolutional Method for Queue-Dependent Servers in Closed Queueing Networks, *Performance Evaluation 1* (1981), 7-18.

[Sau82]
C. H. Sauer, E. A. McNair and J. F. Kurose, The Research Queueing Package, Version 2: CMS Users Guide, Vol. IBM Research Report RA 139 (#41127), April 1982.

[Sau83]
C. H. Sauer, Computational Algorithms for State-Dependent Queueing Networks, *ACM Transactions on Computer Systems 1* (1983), 67-92.

[Sch79]
P. Schweitzer, Approximate Analysis of Multiclass Closed Network of Queues, *Proceedings of International Conference on Stochastic Control and Optimization*, 1979.

[Sch78]
H. D. Schwetman, Hybrid Simulation Models of Computer Systems, *Communications of the ACM 21, No. 21* (September 1978), 718-723.

[Sev81]
K. C. Sevcik and I. Mitrani, The Distribution of Queueing Network States at Input and Output Instants, *Journal of ACM 28, 2* (April 1981), 358-371.

[Sho80]
J. F. Shoch and J. A. Hupp, Measurement Performance of an Ethernet Local Network, *Communication of ACM 24, 8* (December 1980), 711-721.

[Sil84]
E. S. Silva and M. Gerla, Load Balancing in Distributed Systems with Multiple Classes and Site Constraints, *Performance '84*, 1984, 17-33.

[Sur83]
R. Suri, Generalized Quick Bounds for Performance of Queueing Network, Technical Report Tech. Rep.-05083, Center for Research in Computing Technology, Harvard University, 1983.

[Wal85]
P. Wallich, Minis and Mainframes, Technology 85, *IEEE Spectrum 22, 1* (January 1985), 42-44.

[Whi84]
W. Whitt, Open and Closed Models for Networks of Queues, *AT&T Bell Laboratories Technical Journal 63, 9* (November 1984), 1911-1979.

[Zah80]
J. Zahorjan, *The Approximate Solution of Large Queueing Network Models*, Ph.D. Dissertation, University of Toronto, Toronto, Canada, 1980.

[Zah81]
J. Zahorjan and E. Wong, The Solution of Separable Queueing Models Using Mean Value Analysis, *ACM Sigmetrics Conference*, 1981, 80-85.

[Zah82]
J. Zahorjan, K. C. Sevcik, D. L. Eager and B. Galler, Balanced Job Bound Analysis of Queueing Networks, *Communications of the ACM 25, 2* (February 1982), 134-141.

[Zah86]
J. Zahorjan, D. L. Eager and H. Sweillam, Accuracy, Speed, and Convergence of Approximate Mean Value Analysis, Technical Report 86-08-07, University of Washington, August 1986.

Appendix A

We adopt the following notation throughout:

$\mathbf{K}$	=	$(K_1, K_2, \ldots, K_R)$ = population vector.
K_r	=	population of chain r (class r jobs).
K	=	total population of the closed network, $K = \sum_{j=1}^{R} K_j$.
$S_p(r)$	=	set of service centers visited by chain r with property p (if no property, p is omitted).
$R_p(s)$	=	set of chains with property p that visit service center s.
R^o, R^c	=	number of open (o) or closed (c) chains.
s_{rs}	=	mean service time demand of a customer of type r at service center s.
$\mu_s(k)$	=	service rate of server s as a function of the number of customers, k, at the center.
θ_{rs}	=	relative frequency of visits of a class r job to center s.
L_{rs}	=	$s_{rs}\theta_{rs}$ = loading factor of class r jobs at service center s
Θ_r	=	$(\theta_{r1}, \theta_{r2}, \ldots, \theta_{rM})$ = the vector of relative frequencies of visits for class r jobs.
λ_{rs}	=	throughput rate of class r jobs at service center s.
λ_s	=	throughput rate at service center s, $\lambda_s = \sum_{j \in R(s)} \lambda_{js}$.
$S_o(r), S_c(r)$	=	set of service centers visited by open (o) or closed (c) chain r.
$R_o(s), R_c(s)$	=	set of open (o) or closed (c) chains visiting service center s.
ρ_{rs}	=	utilization of service center (single server) s due to class r customers.
ρ_s	=	utilization of the single server center s, $\rho_s = \sum_{j \in R(s)} \rho_{js}$.
P_r	=	routing matrix of chain r (for non-hopping class networks).
N_{rs}	=	mean number of jobs of class r at service center s.
N_s	=	mean number of jobs of all classes at service center s, $N_s = \sum_{j \in R(s)} N_{js}$.
W_{rs}	=	mean waiting time (including service time) of a chain r customer at server s.
W_s	=	mean service time of all job classes at server s.
M_s	=	number of servers at service center s.
$p_s(k)$	=	equilibrium marginal queue size probability of service center s.
e_r	=	r dimensional unit vector in the direction of r.
$S(\mathbf{K}, M)$	=	set of all feasible states of a closed network with M servers and population vector $\mathbf{K}$.
$S(\Lambda_o, \mathbf{K}_c, M)$	=	set of all feasible states of a mixed network with external arrival rate vector Λ_o, population vector $\mathbf{K}_c$, and M service centers.
$\lambda_{o,r}$	=	external arrival rate of chain r customers.
Λ_o	=	$(\lambda_{o,1}, \lambda_{o,2}, \ldots, \lambda_{o,R})$ = external arrival rate vector.
$p_{o,rs}$	=	probability of an external job of class r arriving at service center s.
$\mathbf{P}_{o,r}$	=	$(p_{o,r1}, p_{o,r2}, \ldots, p_{o,rM})$ = external arrival probability vector of chain r.
Z^+	=	the set of all positive integers.
$\mathbf{k}_s$	=	$(k_{1s}, k_{2s}, \ldots, k_{Rs})$ = steady state vector of service center s.
$\mathbf{k}$	=	$(\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M)$ = equilibrium state vector of the network. Subscripts indicate the service center.

Appendix B

In this appendix we prove that the alternative forms for the normalizing constant of mixed networks given in (2.4.9), (2.4.10), and (2.4.12) are the same.

Observe that:

$$\left(\sum_{i\varepsilon R_*(s)}\rho_{is}^o\right)^{k_s^o}=\sum_{\mathbf{k}_s^o}\left[\begin{matrix}k_s^o\\ k_{1s}^o,k_{2s}^o,\ldots,k_{R_*s}^o\end{matrix}\right]\prod_{i\varepsilon R_*(s)}(\rho_{is}^o)^{k_{is}^o}= \tag{B.1}$$

$$=\sum_{\mathbf{k}_s^o}\frac{k_s^o!}{k_{1s}^o!k_{2s}^o!\cdots k_{R_*s}^o!}\prod_{i\varepsilon R_*(s)}(\rho_{is}^o)^{k_{is}^o}=$$

$$=\sum_{\mathbf{k}_s^o}k_s^o!\prod_{i\varepsilon R_*(s)}\frac{(\rho_{is}^o)^{k_{is}^o}}{k_{is}^o!},$$

where the second equality holds by the multinomial theorem. Substituting for $\left(\sum_{i\varepsilon R_*(s)}\rho_{is}^o\right)^{k_s^o}$ in equation (2.4.9) results in (2.4.10).

To show that equation (2.4.12) can be derived from (2.4.9), we rewrite (2.4.9) in the following form:

$$G(\Lambda_o,\mathbf{K}_c)=\sum_{\mathbf{k}\varepsilon S(\Lambda_*,\mathbf{K}_*,M)}\prod_{s=1}^{M}\left(\sum_{i\varepsilon R_*(s)}\rho_{is}^o\right)^{k_s^o}\frac{(k_s^c+k_s^o)!}{k_s^o!}\prod_{j\varepsilon R_*(s)}\frac{(\theta_{js}^c s_{js}^c)^{k_{js}^c}}{k_{js}^c!}\frac{(1-\rho_s^o)^{k_s^c}}{(1-\rho_s^o)^{k_s^c}}, \tag{B.2}$$

where we have multiplied (2.4.9) by a term equal to $\frac{(1-\rho_s^o)^{k_s^c}}{(1-\rho_s^o)^{k_s^c}}=1$. After simplifying the result, and observing that $\prod_{j\varepsilon R_*(s)}(1-\rho_s^o)^{k_{js}^c}=(1-\rho_s^o)^{k_s^c}$, we can write:

$$G(\Lambda_o,\mathbf{K}_c)=\sum_{\mathbf{k}\varepsilon S(\Lambda_*,\mathbf{K}_*,M)}\prod_{s=1}^{M}(\rho_s^o)^{k_s^o}(1-\rho_s^o)^{k_s^c}\frac{(k_s^c+k_s^o)!}{k_s^o!}\prod_{j\varepsilon R_*(s)}\frac{\left(\theta_{js}^c\frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{B.3}$$

We multiply the right hand side of equation (B.3) by a term equal to $\frac{k_s^c!}{k_s^c!}=1$. This results in:

$$G(\Lambda_o,\mathbf{K}_c)=\sum_{\mathbf{k}\varepsilon S(\Lambda_*,\mathbf{K}_*,M)}\prod_{s=1}^{M}(\rho_s^o)^{k_s^o}(1-\rho_s^o)^{k_s^c}\frac{(k_s^c+k_s^o)!}{k_s^o!k_s^c!}k_s^c!\prod_{j\varepsilon R_*(s)}\frac{\left(\theta_{js}^c\frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{B.4}$$

The state vector of a mixed network $\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)$ is of the form $\mathbf{k}=(\mathbf{k}^c,\mathbf{k}^o)$. Each element of vector $\mathbf{k}^o$ varies from zero to infinity. We can write (B.4) in the following form:

$$\sum_{k_s^c}\sum_{k_1^o}^{\infty}\cdots\sum_{k_M^o=0}^{\infty}\prod_{s=1}^{M}(\rho_s^o)^{k_s^o}(1-\rho_s^o)^{k_s^c}\frac{(k_s^c+k_s^o)!}{k_s^o!k_s^c!}k_s^c!\prod_{j\varepsilon R_*(s)}\frac{\left(\theta_{js}^c\frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{B.5}$$

For all values of s, the terms $k_s^c!\prod_{j\varepsilon R_*(s)}\frac{\left(\theta_{js}^c\frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}$ are independent of open chain states. From the binomial theorem we know that:

$$\sum_{k_s^o}^{\infty}(\rho_s^o)^{k_s^o}(1-\rho_s^o)^{k_s^c}\frac{(k_s^c+k_s^o)!}{k_s^o!k_s^c!}=\left[(\rho_s^o)+(1-\rho_s^o)\right]^{k_s^o+k_s^c}=1, \tag{B.6}$$

Therefore, except for the case when $k_s^c=0$, all the summations over the open chain state vector add up to 1. When the k_s^c terms are zero for each s, the summation in (B.6) is the same as $\sum_{k_s^o}(\rho_s^o)^{k_s^o}=\frac{1}{1-\rho_s^o}$ (assuming that $\rho_s^o<1$ for all s). Using these results, equation (2.4.12) can easily be obtained.