A Hierarchical Approximation Algorithm for Large Multichain Product Form Queueing Networks†

H. R. Bahadori

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley

ABSTRACT

In recent years we have witnessed an increasing proliferation of local area network-based distributed systems. Very large distributed systems based on wide-area networks are already in the design stages in numerous research organizations. In these systems, resources such as processing power, databases, and software are shared among users and jobs at different sites. Modeling and evaluating the performance of such large systems typically require the solution of queueing network models with large numbers of chains (classes), service centers, and populations. These large models preclude any use of exact solution techniques. Therefore, it is important that efficient and cost effective approximate algorithms for the solution of large multichain queueing networks be devised to aid in the modeling, configuration, planning, performance evaluation, and design of the systems these models represent. In this paper we propose a hierarchical approximation technique for multiclass separable queueing networks. This technique, which relies on network transformations, provides us with a smooth tradeoff between cost and accuracy. The key elements of the approach entail transforming queueing networks containing multiple infinite servers into ones containing a single infinite server model in the first step. 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 error 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 which compare this method with those yielding exact values and with other approximate algorithms.

1. Introduction

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. A good example of a VLDS operating system project is the DASH project currently under way at Berkeley [And87]. 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

† This work was supported in part by the National Science Foundation under grant DMC-8503575. The views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of the sponsoring institution.

it increasingly important that quantitative tools be devised to aid in the modeling, configuration, planning, design, and performance evaluation of these systems. Modeling and evaluating the performance of such large 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.

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 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 must be used to reflect these different usage patterns.

The most obvious result of this trend is that the system analyst must employ queueing network models with increasing numbers of customers, classes, and service centers. Thus, it is important to develop accurate and cost effective methods for analysis of large queueing networks. There are two major algorithms that have been used to solve exactly product form multichain QN models: the convolution algorithm [Rei75], [Lav83], and the mean value analysis (MVA) [Rei80]. The computational complexities of these algorithms become prohibitively expensive and make their use practically impossible for solving large closed multichain QNs. Recently, a few new approaches have been proposed to handle large closed QNs. Pittel [Pit79] investigated the asymptotic behavior of multichain separable QNs with capacity constraints. Lavenberg considered the asymptotic properties of a class of product form QNs [Lav80]. Whitt uses an approach that transforms the same type of QNs considered in [Lav80] into an approximate open QN [Whi84]. The method of asymptotic expansions [McK82], [Ram82], and the performance bound hierarchies (PBH) algorithm [Eag86], both provide lower and upper bounds on performance measures. The asymptotic expansion algorithm is applicable to QNs in which all chains visit an infinite server service center and all other centers have utilizations that are not too high, e.g., no more than .85. Extensions of this algorithm have been made to include load dependent centers [Mit84]. The PBH algorithm does not have the same restrictions as those in the asymptotic expansion and can be applied to load dependent service centers. However, in the PBH algorithm, as with the MVA algorithm, the computational complexities grow combinatorially with the number of servers.

The key to our methodology is a hierarchical approximation based on queueing network transformations that provide for a smooth tradeoff between accuracy and computational solution cost. Our approach gracefully introduces approximations commensurate with the increasing complexity of the model. A typical example is shown in Figure 1, where the nodes of a distributed system are shown connected by an internetwork. Each of the nodes may consist of several sites connected by a local area network. The typical node and site configurations are shown in Figure 2. Assuming that the system can be modeled by a closed multiclass product form network, we then proceed to transform the queueing networks (QN) into smaller mixed networks by decoupling the smaller building blocks from the whole network. This is done by replacing the closed chains by open chains at each stage of this hierarchical approximation. Employing this transformation at each stage, we gradually decrease the computational complexity up to the point where the solution becomes feasible.

We will start in section 2 with a review of product form queueing network steady state distribution and mean value analysis. We then provide a brief discussion of exact solution techniques and their computational complexities in sections 2.1, 2.2, and 2.3. In sections 2.4 and 2.5 the closed multichain separable queueing network MVA solution is extended to the case of mixed networks. There we derive the mean value equations for open and closed chains. The complexity of the mixed networks is then examined. In sections 2.6 and 2.7 we derive the solution of open multichain product form networks and discuss the computational complexities of such a solution. In section 3 we describe

our approximation algorithm and prove a theorem for the transformation of a class of multichain QNs into those we can apply our algorithm to. In sections 4.1, 4.2, 4.3, 4.4 we describe our algorithm for solving very large multichain QNs. In sections 4.5, 4.6, 4.7, and 4.8, the asymptotic properties, error criteria and stopping rules, and computational complexities of the algorithm are presented in this order. The experimental results and conclusions are discussed in sections 5 and 6.

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.
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	=	$(\lambda_{o,1},\lambda_{o,2}, \ldots ,\lambda_{o,R^o})$ = 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.

2. Product Form Queueing Network Solution

In this section we will look at the steady state solution of a subset of the class of queueing network models. Queueing networks in this subset are referred to as multiple class product form queueing networks (alternatively, multiple class separable queueing networks, or BCMP networks). In the first subsection we describe the characteristics of these networks and their steady state solution using the convolution algorithm. In the second subsection we outline the mean value analysis algorithm for solving such networks.

2.1. Separable Queueing Networks

We 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:

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

Suppose that the queueing network is a nonhopping-class closed network, and that the routing matrix P_r is irreducible over $S(r)$. Then, from standard Markov chain analysis we can write:

$$\overline{\Theta}_r = \overline{\Theta}_r P_r\ , \qquad r = 1,2,...,R, \tag{2.1.2}$$

where $\overline{\Theta}_r = (\theta_{r1},\theta_{r2}, \ldots ,\theta_{rM})$. 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:

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

Furthermore, we can write:

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

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:

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

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 all job classes.

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:

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

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:

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

where:

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

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

We also have:

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

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

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

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

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 written:

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

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.

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

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)

Reiser and Lavenberg [Rei80] have provided another approach and derived a recursive relation 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 general form of state dependent service rates [Rei81] [Sau83].

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

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

If service center s is FCFS, PS, or LCFSPR center with unit service rate ($\mu_s(i)=1$), then:

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

and, if the service center is IS, then $W_{rs} = s_{rs}$. It can be shown that if the service rate of service center s is a constant, i.e., $\mu_s(i) = c$, then the mean service demands s_{rs} in equation (2.2.2) are multiplied by $\frac{1}{\mu_s(i)} = \frac{1}{c}$.

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:

$$W_{rs}(\bar{K}) = \begin{cases} s_{rs}\left(1 + \sum_{j=1}^{R} N_{js}(\bar{K} - e_r)\right) & \text{FCFS, LCFSPR, PS,M}_s\text{=1, unit service rate} \\ s_{rs} & \text{IS.} \end{cases} \tag{2.2.3}$$

From Little' law [Lit61]

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

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 $W_{rs'}(\bar{K})\theta_{rs'}/\theta_{rs}$. Thus:

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

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

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

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

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

The utilization of service center s for class r jobs can be determined by:

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

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 Complexity 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 convolution algorithms.

The evaluation of visit ratios for both the convolution algorithm and the MVA 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 discard 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].

---

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

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 a solution algorithm. 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 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 where 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 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 these 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 servers. In Table 2 we show the computational complexities for load dependent product form networks.

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 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 values given for the number of operations in the convolution algorithm are those for the computation of the normalizing constant, throughput rates, utilizations, and the mean queue lengths. In the case of MVA, they represent the total number of operations for the computation of mean queue lengths, throughputs, and mean waiting times for all classes. 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, a load independent queueing network with $R=8$, $K_r=5$ $r=1,2...,10$, and $M=5$ needs about 14.84 Mbytes of memory and 268.74 Mflops if the mean values of the performance measures for all chains are to be obtained.

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 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^o_{rs} 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. We define $\lambda_{o,rs}$ to 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:

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

For the open chains, the following balance equation holds:

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

In equation (2.4.2) we assume 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 nonhopping cluster (i.e., that none of the jobs in any of the open chains change class), then the above set of equations becomes a set of R^o matrix equations of the form:

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

Here $\mathbf{P}_{o,r}$ is the probability vector of external arrival rates of class r jobs at the mixed network. Hence:

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

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

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

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 equations require the same space and time complexities to solve.

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

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

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]):

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

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 new 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 containing 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:

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

Proof:

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

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

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

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

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 A for a proof). l^o_s is given by:

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

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]:

$$G_c(\mathbf{K}_c) = \sum_{\mathbf{k}^c \varepsilon S} \prod_{s=1}^{M} k_s^c! \prod_{j \varepsilon R_c(s)} \frac{\left[\theta_{js}^c \dfrac{s_{js}^c}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o}\right]^{k_{js}^c}}{k_{js}^c!}. \tag{2.4.13}$$

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:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho_{rs}^o} = \left[\frac{G_c(\mathbf{K}_c)}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o} \sum_{j \varepsilon R_c(s)} N_{js}^c(\mathbf{K}_c)\right]\left(\prod_{s=1}^{M} \frac{1}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o)}\right) + \left[\frac{1}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o}\right]^2 \left[\prod_{\substack{j=1 \\ j \neq s}}^{M} \frac{1}{1 - \sum_{i \varepsilon R_o(j)} \rho_{ij}^o}\right] G_c(\mathbf{K}_c), \tag{2.4.14}$$

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

$$\frac{\partial G_c(\mathbf{K}_c)}{\partial \rho_{rs}^o} = \sum_{\mathbf{k} \varepsilon S} \prod_{s=1}^{M} k_s^c! \sum_{j \varepsilon R_c(S) j \varepsilon R_c(s)} \prod \frac{\left[\theta_{js}^c \dfrac{s_{js}^c}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o}\right]^{k_{js}^c - 1}}{k_{js}^c!} (k_{js}^c) \frac{\theta_{js}^c s_{js}^c}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o)^2}, \tag{2.4.15}$$

which is the same as:

$$\frac{\partial G_c(\mathbf{K}_c)}{\partial \rho_{rs}^o} = \frac{G_c(\mathbf{K}_c)}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o} \sum_{j \varepsilon R_c(s)} N_{js}^c(\mathbf{K}_c), \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{2.4.16}$$

Equation (2.4.14) is then simplified to:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho_{rs}^o} = \left[\frac{G_c(\mathbf{K}_c)}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o} \sum_{j \varepsilon R_c(s)} N_{js}^c(\mathbf{K}_c)\right]\left[\prod_{s=1}^{M} \frac{1}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o)}\right] + \frac{G(\Lambda_o,\mathbf{K}_c)}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o},$$

$$r \varepsilon R_o(s), \quad s = 1,2,...,M,$$

which in turn is reduced to:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho_{rs}^o} = \frac{G(\Lambda_o,\mathbf{K}_c)}{1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o}\left\{1 + \sum_{j \varepsilon R_c(s)} N_{js}^c(\Lambda_o,\mathbf{K}_c)\right\}, \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{2.4.17}$$

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:

$$\frac{\partial G(\Lambda_o,\mathbf{K}_c)}{\partial \rho_{rs}^o} = \frac{G(\Lambda_o,\mathbf{K}_c)}{\rho_{rs}^o} N_{rs}^o(\Lambda_o,\mathbf{K}_c), \quad r \varepsilon R_o(s), \quad s = 1,2,...,M. \tag{2.4.18}$$

---

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

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

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

■

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:

$$N^c_{rs}(\Lambda_o,\mathbf{K}_c) = \frac{\lambda^c_{rs}s^c_{rs}(1 + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r))}{(1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is})}, \qquad r\varepsilon R_c(s), \qquad s=1,2,...,M. \tag{2.4.20}$$

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 is given here [Zah81]. 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:

$$\begin{aligned}
N^c_{rs}(\Lambda_o,\mathbf{K}_c) &= \lambda^c_{rs}s^c_{rs}\left[1 + \sum_{i\varepsilon R_o(s)} N^o_{is}(\Lambda_o,\mathbf{K}_c - e_r) + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r)\right] \\
&= \lambda^c_{rs}s^c_{rs}\left[1 + \frac{\sum_{i\varepsilon R_o(s)} \rho^o_{is}(1 + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r))}{(1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is})} + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r)\right] \\
&= \lambda^c_{rs}s^c_{rs}\left[1 + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r)\right]\left(1 + \frac{\sum_{i\varepsilon R_o(s)} \rho^o_{is}}{1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is}}\right) \\
&= \frac{\lambda^c_{rs}s^c_{rs}(1 + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c - e_r))}{(1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is})},
\end{aligned}$$

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

■

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

$$N^o_{rs}(\mathbf{K}) = \rho^o_{rs}, \quad r\varepsilon R_o(s), \qquad s = 1,2,...,M. \tag{2.4.21}$$

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

$$N^c_{rs}(\mathbf{K}_c) = \frac{\lambda^c_{rs}s^c_{rs}}{(1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is})}, \qquad r\varepsilon R_c(s), \qquad s=1,2,...,M. \tag{2.4.22}$$

But, since at the infinite servers we have $\sum_{i \epsilon R_o(s)} \rho_{is}^o = 0$, equation (2.4.22) is further simplified to:

$$N_{rs}^c(\mathbf{K}_c) = \lambda_{rs}^c s_{rs}^c \qquad r \epsilon R_c(s), \qquad s = 1,2,...,M. \tag{2.4.23}$$

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:

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

and that for closed chain jobs:

$$W_{rs}^c(\mathbf{K}) = \left[\frac{s_{rs}^c}{1 - \sum_{i \epsilon R_o(s)} \rho_{is}^o} \right] (1 + \sum_{j \epsilon R_c(s)} N_{js}^c(\mathbf{K} - e_r)) \qquad r \epsilon R_c(s), \qquad s = 1,2,...,M. \tag{2.4.25}$$

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

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

$$s = 1,2,...,M. \tag{2.4.26}$$

Equation (2.4.26) can be further simplified to:

$$N_s(\mathbf{K}) = \sum_{r \epsilon R_c(s)} \frac{\lambda_{rs}^c s_{rs}^c (1 + \sum_{j \epsilon R_c(s)} N_{js}^c(\mathbf{K} - e_r))}{(1 - \sum_{i \epsilon R_o(s)} \rho_{is}^o)} + \frac{\rho_s^o}{1 - \rho_s^o}(1 + N_s^c(\mathbf{K})), \qquad s = 1,2,...,M, \tag{2.4.27}$$

where $\rho_s^o = \sum_{i \epsilon R_o(s)} \rho_{is}^o$ 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 $\frac{1}{(1 - \sum_{i \epsilon R_o(s)} \rho_{is}^o)}$. 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 $\rho_s^o = \sum_{i \epsilon R_o(s)} \rho_{is}^o$, 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 Complexity of Mixed Multichain Network Solution

We can determine the computational complexity 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 derived 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 Chain 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 $N_s^c(\Lambda_o,\mathbf{K}_c) = \sum_{j \in R_{c(s)}} N_{js}^c(\Lambda_o,\mathbf{K}_c)$. 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 nonhopping class network. If the original queueing network is a class hopping one, we have to transform it to a nonhopping class network. This transformation allows us to compute the visit ratios of the QN in the sense that the visit ratios 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 nonhopping class 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 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 this case in 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. l 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 .

Table 3 Space/Time Complexity (Load Independent Servers)

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

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

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 servers and infinite servers, 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:

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

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

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:

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

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

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

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):

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

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 server of type 1,2, or 3 is given by:

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

2.7. The Computational Complexity of Open Network Solutions

Since in mixed networks the open and closed chains interact, the complexities of solving open networks are best determined by the 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 computed in section 2.5.1. We need an amount of storage space equal to RM^2 to determine visit ratios of all classes simultaneously and 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:

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

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 solution of the 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 increase (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_r,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}

3. An Approximation Based on Network Transformation

We are now ready to formulate our methodology for an accurate and cost effective hierarchical technique to approximately solve large multichain separable queueing network models, for instance, those models that represent large distributed systems. 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 multiclass separable queueing network with 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. When modeling distributed systems, an 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 composite infinite server service center. An outline of this transformation is discussed in section 3.1. 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 lower and upper bound constant rate Poisson sources dependent on the mean values 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 methodology is discussed in sections 4.

3- N3 is then solved using exact solution algorithms. The choice of bound hierarchies and number of iterations for the open chains depends on the desirable accuracy and computational complexity (both space and time) costs. The stopping rules, error criteria, computational complexities, and experimental results are presented in sections 4.6, 4.7, 4.8, and 5. 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 no class hopping and with the same steady state solution.

3.1. Transformation into a Network with One Infinite Server

Theorem 3.1.1:

> Consider a multichain product form queueing network containing fixed rate single servers and infinite servers 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 all the chains in the original network visit 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 $R(T) = R(1) \bigcup R(2) \bigcup \cdots \bigcup R(I)$. Note that the sets $R(1),R(2),...,R(I)$ are mutually exclusive sets, i.e., $R(i) \bigcap R(j) = \varnothing$ 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 $R(1),R(2),...,R(I)$ 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 transition probability of reaching each of the new classes at the aggregate infinite server is the same as the transition probability reaching each of the infinite servers in the original network. The result is a class-hopping product network with a single infinite server.

Once again, the proof that the steady state probability distribution of the original network can be obtained from that of the constructed can be shown by directly manipulating the distribution of the multiple infinite server QN and arriving at the solution for the QN with one infinite server. 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 nonclass-hopping QN using the Reiser and Kobayashi algorithm [Rei75]. Henceforth, without any loss of generality, we assume that this procedure has been implemented, and that we have a separable queueing network with at least one infinite server visited by all the chains.

■

4. Closed-to-Mixed Network Transformation

In this section we present an algorithm that generates a hierarchical approximation for multichain product form QNs with fixed rate single server service centers and 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 solving 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 a desirable balance between error and complexity is achieved. In practice, the 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 formats. Pittel [Pit79] investigated the asymptotic behavior of multichain separable QNs with capacity constraints. Lavenberg [Lav80] considers closed multichain networks where all the chains visit an IS service center and proposes a fixed initial value

(given by equation (4.5.4)) for the arrival rates of the open chains resulting from transforming closed networks into open networks. He then derives the asymptotic values for the performance measures of the originally closed networks. Whitt [Whi84] uses an algorithm that transforms the same type of closed networks into an open network. Our algorithm includes both of the last two cases as the limiting cases.

4.1. Open Chain Arrival Rates Estimation

Assume that the network has been transformed to an equivalent network containing one 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.

Let us 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 open chain constant arrival vector 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 4.5). Hence, applying these two laws to the mixed network, and using equations (2.4.8) and (2.4.20), we can write:

$$\lambda_{o,r}s_{o,r} = K_r - \sum_{s\varepsilon S_o(r)} \frac{\rho^o_{rs}(\Lambda_o)(1 + \sum_{j\varepsilon R_c(s)} N^c_{js}(\Lambda_o,\mathbf{K}_c))}{(1 - \sum_{i\varepsilon R_o(s)} \rho^o_{is}(\Lambda_o))}, \qquad r\varepsilon O, \tag{4.1.1}$$

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

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

$$r\varepsilon O. \tag{4.1.2}$$

Observing that $s_{o,r} = (\mu_{o,r})^{-1}$ and simplifying equation (4.1.2), we obtain:

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

$$r\varepsilon O. \tag{4.1.3}$$

Equations (4.1.1) and (4.1.2) 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 (4.1.1) will also apply to the original closed network if certain asymptotic conditions (section 4.5) are satisfied. A discussion of the asymptotic properties of equations (4.1.1) - (4.1.3) will be presented in section 4.5. We recall that the constraint of stability for the mixed network should always be satisfied. This, as we discussed earlier, requires that $\rho^o_s < 1$ for *all* s.

4.2. Basic Definitions and Theorems

In this section we recall some basic definitions and theorems relating to the contraction mapping which we will use to solve the set of equations represented by equations (4.1.1) - (4.1.3) see Ortega [Ort70].

Definition 4.2.1:

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:

$$||\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})|| \leq ||\mathbf{x} - \mathbf{y}||, \quad \text{for } all \ \ \mathbf{x},\mathbf{y} \varepsilon D_o. \tag{4.2.1}$$

The map is said to be strictly nonexpansive on D_o if strict inequality holds in (4.2.1) 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 4.2.2:

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 $\gamma \varepsilon (0,1)$ such that $||\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})|| \leq \gamma ||\mathbf{x} - \mathbf{y}||$ for all $\mathbf{x},\mathbf{y} \varepsilon D_o$. Obviously, any contraction mapping is strictly nonexpansive.

Theorem 4.4.1 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 4.2.1:

If we form the sequence $\mathbf{x}^{(I+1)} = \mathbf{F}(\mathbf{x}^{(I)})$, $I = 1,2,....$, then the sequence $\left\{\mathbf{x}(I), \quad I = 1,2,...\right\}$ 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 4.2.2 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 $\mathbf{x}^{(i)} \varepsilon D_o$ for some $i \varepsilon I$.

For a complete proof of this and the other theorems see [Ort70]. We give a sketch of the proofs in Appendix A.

4.3. Approximate Arrival Rate Estimation

We now define two sets of iterative equations 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

$\underline{\Lambda}_o^{(I)} = (\underline{\lambda}_{o,1}^{(I)}, \underline{\lambda}_{o,2}^{(I)}, \ldots, \underline{\lambda}_{o,R^o}^{(I)})$ 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:

$$\overline{\lambda}_{o,r}^{(I+1)} = K_r\mu_{o,r} - \sum_{s\varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^o(\underline{\Lambda}_o^{(I)})(1 + \sum_{j\varepsilon R_c(s)} N_{js}^c(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c))}{(1 - \sum_{i\varepsilon R_o(s)} \rho_{is}^o(\underline{\Lambda}_o^{(I)}))}, \qquad r\varepsilon O, \quad I = 1,2,3,\ldots, \tag{4.3.1}$$

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

$$\underline{\lambda}_{o,r}^{(I)} = K_r\mu_{o,r} - \sum_{s\varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^o(\overline{\Lambda}_o^{(I)})(1 + \sum_{j\varepsilon R_c(s)} N_{js}^c(\overline{\Lambda}_o^{(I)}, \mathbf{K}_c))}{(1 - \sum_{i\varepsilon R_o(s)} \rho_{is}^o(\overline{\Lambda}_o^{(I)}))}, \qquad r\varepsilon O, \quad I = 1,2,3,\ldots. \tag{4.3.2}$$

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:

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

$$r\varepsilon O, \quad I=1,2,3,\ldots, \tag{4.3.3}$$

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

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

$$r\varepsilon O, \quad I=1,2,3,\ldots. \tag{4.3.4}$$

Finally, equations (4.3.3) and (4.3.4) are reduced to:

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

$$r\varepsilon O, \quad I=1,2,3,\ldots, \tag{4.3.5}$$

and:

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

$$r \varepsilon O, \quad I = 1,2,3,.... \tag{4.3.6}$$

Though equations (4.3.3) and (4.3.4) can be used to solve for the approximate mean throughput rate, equations (4.3.5) and (4.3.6) 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 set of equations in (4.3.5) and (4.3.6) are reduced to those in [Whi84]. The last two equations also show their sensitivity to ρ_s^o and suggest ways to reduce the effects of ρ_s^os.

4.3.1. Initial Conditions for Open Chain Arrival Rates

The initial condition for the iterative relationships given by equations (4.3.5) and (4.3.6) is defined by:

$$\overline{\lambda}_{o,r}^{(1)} = \alpha K_r \mu_{o,r}, \quad r \varepsilon O, \quad \alpha \varepsilon (0,1], \tag{4.3.7}$$

where α is chosen to satisfy:

$$K_r > \sum_{s=1}^{M} \frac{\rho_{rs}^o(\overline{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon R_c(s)} N_{js}^c(\overline{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^o(\overline{\Lambda}_o^{(1)}))}, \qquad r \varepsilon O. \tag{4.3.8}$$

In most cases the choice $\alpha = 1$ is sufficient. However, there are cases where $\alpha = 1$ does not satisfy (4.3.8), and a different value of α should be chosen to satisfy (4.3.7) and (4.3.8).

4.4. Convergence of the Iterative Algorithm

In this section, we prove that the solutions of the iterative mappings given by equations (4.3.1) and (4.3.2) for the upper and lower bounds of the arrival rates converge to a single fixed point.

Theorem 4.4.1

Consider a mixed multichain separable queueing network containing fixed rate single servers. Then, the mean queue lengths given by the relationship expressed 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:

$$\lambda_{rs}^o = \theta_{rs}^o \sum_{i \varepsilon O} \lambda_{o,i}. \tag{4.4.1}$$

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

$$\rho^o_{rs} = \lambda^o_{rs} s^o_{rs} = s^o_{rs} \theta^o_{rs} \sum_{i \varepsilon O} \lambda_{o,i}. \tag{4.4.2}$$

Therefore the following relationship also holds:

$$\sum_{i \varepsilon R_o(s)} \rho^o_{is} = \sum_{i \varepsilon R_o(s)} \sum_{j \varepsilon O} \theta^o_{is} s^o_{is} \lambda_{o,j}. \tag{4.4.3}$$

Substituting for the ρ^o_{rs} in equation (2.4.8), and using equation (4.4.3), we obtain:

$$N^o_{rs}(\Lambda_o, \mathbf{K}_c) = \frac{\theta^o_{rs} s^o_{rs} \sum_{i \varepsilon O} \lambda_{o,i} (1 + \sum_{j \varepsilon R_c(s)} N^c_{js}(\Lambda_o, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \sum_{j \varepsilon O} \theta^o_{is} s^o_{is} \lambda_{o,j})}, \qquad r \varepsilon R_o(s), \qquad s = 1,2,...,M. \tag{4.4.4}$$

Taking the partial derivative of the function given in equation (4.4.4), we have:

$$\frac{\partial N^o_{rs}(\Lambda_o, \mathbf{K}_c)}{\partial \lambda_{o,r}} = \frac{\theta^o_{rs} s^o_{rs}}{(1 - \sum_{i \varepsilon R_o(s), j \varepsilon O} \theta^o_{is} s^o_{is} \lambda_{o,j})^2} (1 + \sum_{j \varepsilon R_c(s)} N^c_{js}(\Lambda_o, \mathbf{K}_c)) + \frac{\rho^o_{rs}}{1 - \rho^o_s} \frac{\partial}{\partial \lambda_{o,r}} \left[\sum_{j \varepsilon R_c(s)} N^c_{js}(\Lambda_o, \mathbf{K}_c) \right]. \tag{4.4.5}$$

Using equation (2.4.20) we can write:

$$\frac{\partial}{\partial \lambda_{o,r}} \left[\sum_{j \varepsilon R_c(s)} N^c_{js}(\Lambda_o, \mathbf{K}_c) \right] = \sum_{j \varepsilon R_c(s)} \frac{\partial}{\partial \lambda_{o,r}} \frac{\lambda^c_{js} s^c_{js} (1 + \sum_{k \varepsilon R_c(s)} N^c_{ks}(\Lambda_o, \mathbf{K}_c - e_j))}{(1 - \sum_{i \varepsilon R_o(s)} \rho^o_{is})}, \qquad r \varepsilon O. \tag{4.4.6}$$

Equation (4.4.6) is a recursive relationship. Under the assumption of a stable mixed network the first term in equation (4.4.5) 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:

$$\frac{\partial}{\partial \lambda_{o,r}} \left(\frac{1}{(1 - \sum_{\substack{i \varepsilon R_o(s) \\ j \varepsilon O}} \theta^o_{is} s^o_{is} \lambda_{o,j})^m} \right) = \frac{m \theta^o_{rs} s^o_{rs}}{(1 - \sum_{\substack{i \varepsilon R_o(s) \\ j \varepsilon O}} \theta^o_{is} s^o_{is} \lambda_{o,j})^{m+1}}, \quad \text{for all } m \varepsilon Z^+. \tag{4.4.7}$$

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. If the QN contains IS service centers, the proof is trivial.

■

Theorem 4.4.2

Consider a closed multiclass product form queueing network containing fixed rate single servers. Suppose an iterative relation is defined on this network. This relationship is characterized by equation (4.3.1) and the corresponding initial condition provided by (4.3.7) and (4.3.8). 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 $\alpha = 1$. The extension of the proof to the case where $\alpha \neq 1$ is trivial. For I = 1, equation (4.3.2) can be written in the following form:

$$\underline{\lambda}_{o,r}^{(1)} = \bar{\lambda}_{o,r}^{(1)} - \sum_{s \varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^{o}(\bar{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon R_c(s)} N_{js}^{c}(\bar{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^{o}(\bar{\Lambda}_o^{(1)}))}, \qquad r \varepsilon O. \tag{4.4.8}$$

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

$$\underline{\lambda}_{o,r}^{(1)} < \bar{\lambda}_{o,r}^{(1)}, \quad \text{for } all\ r \varepsilon O\ . \tag{4.4.9}$$

From equation (4.3.1) and for I = 1, it follows:

$$\bar{\lambda}_{o,r}^{(2)} = \bar{\lambda}_{o,r}^{(1)} - \sum_{s \varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^{o}(\underline{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon R_c(s)} N_{js}^{c}(\underline{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^{o}(\underline{\Lambda}_o^{(1)}))}\ . \qquad r \varepsilon O, \tag{4.4.10}$$

Observing that the second term on the right hand side of equation (4.4.10) is positive, we have:

$$\bar{\lambda}_{o,r}^{(2)} < \bar{\lambda}_{o,r}^{(1)}, \qquad \text{for } all\ r \varepsilon O\ . \tag{4.4.11}$$

We can also write:

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

$$\sum_{s \varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^{o}(\bar{\Lambda}_o^{(2)})(1 + \sum_{j \varepsilon R_c(s)} N_{js}^{c}(\bar{\Lambda}_o^{(2)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^{o}(\bar{\Lambda}_o^{(2)}))} - \sum_{s \varepsilon S_o(r)} \frac{\mu_{o,r}\rho_{rs}^{o}(\bar{\Lambda}_o^{(1)})(1 + \sum_{j \varepsilon R_c(s)} N_{js}^{c}(\bar{\Lambda}_o^{(1)}, \mathbf{K}_c))}{(1 - \sum_{i \varepsilon R_o(s)} \rho_{is}^{o}(\bar{\Lambda}_o^{(1)}))}. \tag{4.4.12}$$

By (4.4.11) 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 (4.4.12), by theorem 4.4.1, 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:

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

By a similar argument, we can conclude that:

$$\bar{\lambda}_{o,r}^{(I)} < \bar{\lambda}_{o,r}^{(I-1)} < \cdots < \bar{\lambda}_{o,r}^{(2)} < \bar{\lambda}_{o,r}^{(1)} \qquad \text{for } all\ r \varepsilon O. \tag{4.4.13}$$

Observe that by our assumption we also have:

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

■

Corollary 4.3.1

The following relation holds for the lower bound sequence:

$$\underline{\lambda}_{o,r}^{(I)} > \underline{\lambda}_{o,r}^{(I-1)} > \cdots > \underline{\lambda}_{o,r}^{(1)} < \bar{\lambda}_{o,r}^{(1)} \qquad \text{for } all\ r \varepsilon O. \tag{4.4.14}$$

The proof of the corollary follows directly from the proof of theorem 4.4.2.

■

The iterative algorithm, by the contraction mapping theorem and the monotone bounded sequence theorem given in section 4.2, converges. The sequence of lower and upper bound estimates as computed by equations (4.3.1) and (4.3.2) always converges to a unique fixed point as the number of iterations approaches infinity.

$$\lim_{l\to\infty} \bar{\lambda}_{o,r}^{(l)} = \lim_{l\to\infty} \underline{\lambda}_{o,r}^{(l)} = \lambda_r^{'}, \qquad r\varepsilon O. \tag{4.4.15}$$

Obviously, if the set of nonlinear equations (4.1.1) and (4.1.2) are asymptotically correct (they hold true in the limit as K becomes large), then the iterative algorithm is also asymptotically correct and convergent in the limit as the population becomes large.

4.5. Asymptotic Property

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 IS center visited by all the chains. Then let:

$$K_r = \left\lfloor p_r K \right\rfloor, \qquad r\varepsilon\, R(M+1), \tag{4.5.1}$$

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

$$\sum_{i\varepsilon O} p_i = 1. \tag{4.5.2}$$

Furthermore, assume that:

$$\lim_{K\to\infty} K_r\mu_{o,r} \quad \textit{exists} \text{ and } \textit{is finite} \text{ for } \textit{all}\ r\varepsilon\, R(M+1). \tag{4.5.3}$$

Suppose, using the notation in equations (2.1.2) and (2.1.3), that *N1* is the subnetwork of the original closed network N by excluding the IS 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:

$$\lambda_r = K_r\mu_{rM+1}, \qquad r\varepsilon\, R(M+1). \tag{4.5.4}$$

and

$$\rho_{r,s} = s_{rs}\theta_{rs} \sum_{i\varepsilon\, R(M+1)} \lambda_i, \qquad r\varepsilon R(s), \quad s=1,2,\ldots,M, \tag{4.5.5}$$

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

$$\rho_s < 1 \ \text{ for } \textit{all}\ s \neq M+1. \tag{4.5.6}$$

Then it can be shown that [Lav80]:

$$\lim_{K\to\infty} p_1(\mathbf{k}^*, \mathbf{K}) = \prod_{s=1}^{M} \frac{k_s!}{(1 - \sum_{i\varepsilon R(s)} \rho_{is})} \prod_{j\varepsilon R(s)} \frac{(\rho_{js})^{k_{js}}}{k_{js}!}. \tag{4.5.7}$$

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

$$\lim_{K\to\infty} \lambda_{r,s} = \theta_{rs} \sum_{i \in R(M+1)} \lambda_i , \quad r \varepsilon R(s) , \quad s = 1,2,...,M , \tag{4.5.8}$$

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

$$\lim_{K\to\infty} p_1(\mathbf{k}^*,\mathbf{K}) = \mathbf{O} . \tag{4.5.9}$$

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 servers 1,2,...,M and obtained from N by replacing the IS service center (the IS service center visited by all chains) with constant external Poisson arrival rates as defined by (4.5.4).

Furthermore, let $\mathbf{X}(\mathbf{K})$ be the random vector with probability distribution given by $\left\{p_1(\mathbf{k}^*,\mathbf{K}) : \mathbf{k}^* \varepsilon S^*(\mathbf{K})\right\}$. i.e., $\mathbf{X}(\mathbf{K})$ is the equilibrium state vector of 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:

$$\lim_{K\to\infty} E\left[f(\mathbf{X}(\mathbf{K}))\right] = E\left[f(\mathbf{X}^o)\right] . \tag{4.5.10}$$

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 (4.5.10) 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 the conclusions hold true as long as the open network remains stable. If $\rho_s^o > 1$ 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* $r \varepsilon R(M+1)$.

4.6. Stopping Rules and Error Criteria

We propose several error criteria for the analysis of our approximate algorithm. The accuracy and cost of any iterative algorithm is 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 used as a stopping rule is defined by:

$$E_s(I) = E_s(\bar{N}_s^{(I)}, \underline{N}_s^{(I)}) = \max_s \frac{(\bar{N}_s^{(I)} - \underline{N}_s^{(I)})}{\underline{N}_s^{(I)}} . \tag{4.6.1}$$

The corresponding stopping rule is to halt the iterations 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:

$$E_r(\bar{\lambda}_{o,r}^{(I)}, \underline{\lambda}_{o,r}^{(I)}) = \max_r \frac{(\bar{\lambda}_{o,r}^{(I)} - \underline{\lambda}_{o,r}^{(I)})}{\underline{\lambda}_{o,r}^{(I)}} . \tag{4.6.2}$$

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

$$\bar{E}_{r,s}(I) = \max_{r,s} \frac{\left| N_{rs}(\mathbf{K}) - N_{rs}(\bar{\Lambda}_o^{(I)}, \mathbf{K}_c) \right|}{N_{rs}(\mathbf{K})}. \tag{4.6.3}$$

In equation (4.6.3) $N_{rs}(\mathbf{K})$ is the exact value of the mean queue length. The corresponding lower bound error criterion is similarly defined as:

$$\underline{E}_{r,s}(I) = \max_{r,s} \frac{\left| N_{rs}(\mathbf{K}) - N_{rs}(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c) \right|}{N_{rs}(\mathbf{K})}. \tag{4.6.4}$$

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

$$\underline{E}_s(I) = \max_{s} \frac{\left| N_s(\mathbf{K}) - N_s(\underline{\Lambda}_o^{(I)}, \mathbf{K}_c) \right|}{N_s(\mathbf{K})}, \tag{4.6.5}$$

and

$$\bar{E}_s(I) = \max_{s} \frac{\left| N_s(\mathbf{K}) - N_s(\bar{\Lambda}_o^{(I)}, \mathbf{K}_c) \right|}{N_s(\mathbf{K})}. \tag{4.6.6}$$

For simplicity of notation and ease of comparison, we have assumed that the original network is a closed network with a population vector of $\mathbf{K}_c$. The upper and lower bound 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 which is also based on queue lengths is [Cha75]:

$$E(I) = \max_{r,s} \frac{\left| N_{rs}(\mathbf{K}) - \hat{N}_{rs}(\mathbf{K}) \right|}{K_r}. \tag{4.6.7}$$

In equation (4.6.7) $\hat{N}_{rs}$ is the estimate of the queue length. The errors given by (4.6.3) - (4.6.6) are usually much smaller than those given by equations (4.6.3) and (4.6.4) for a large population vector. This is due to the fact that the denominator of equation (4.6.7) increases more rapidly than those in equations (4.6.3) and (4.6.4). We believe that the error criteria given by equations (4.6.3) and (4.6.4) are more realistic since they are not scaled down by a large number as the one defined in (4.6.7).

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 is 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 (4.6.1). 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 $e_s = .0001$ 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 (4.6.2) is not as accurate (for the same error bound) as the queue length based rule, but it results in lower computational cost.

4.7. Computational Complexity

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 $\overline{E_s}(I) < e_s$ stopping rules, the mean queue length values have to be evaluated at each intermediate iteration (iterations 1 through (I-1)). The stopping rule does not substantially worsen the space complexity burden. The complexity is not increased for other performance metrics. The space complexity remains the same except for an additional term equal to $2R^oM$. Therefore, the space complexities of the of the solution of the original network, a single iteration over the approximate network, and I iterations also over the approximate network are as follows:

$$S(R,M) = (R+M+RM)\prod_{i=1}^{R-1}(K_i + 1) + 4RM, \tag{4.8.1}$$

$$S(R^o,R^c,M) = (R^c + M + R^cM)\prod_{i=1}^{R^c-1}(K_i + 1) + 5R^oM + 4R^cM, \tag{4.8.2}$$

$$S(R^o,R^c,M,I) = S(R^o,R^c,M) + R^oM. \tag{4.8.3}$$

Clearly, the space complexity of the original network as given by equation (4.8.1) is much larger than those for the mixed network and given by equations (4.8.2) and (4.8.3). This is due to the fact that, for most large networks, the *dominant term* in all these equations is the closed chain populations product term $\prod_{i=1}^{R-1}(K_i + 1)$.

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, which shows some 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 that $R = 15$ and $M = 15$. 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:

$$\Delta_1 = \Delta_1(R^o,R^c,M) = R^o(1+M)(K_1+1)^{R^o+R^c-1} +$$

$$(R^c+M+R^cM)(K_1+1)^{R^c-1}\left[(K_1+1)^{R_o} - 1\right] - R^oM\ , \tag{4.8.4}$$

$$\Delta_I = \Delta(R^o,R^c,M,I) = \Delta_1(R^o,R^c,M) - R^oM. \tag{4.8.5}$$

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 to open chains increases.

The derivation of 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:

$$T(R,M) = 4RM\prod_{r=1}^{R}(K_r + 1) + 3RM + 2RM^3. \tag{4.8.6}$$

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

$$T(R^o,R^c,M) = 4R^cM\prod_{r=1}^{R^c}(K_r + 1) + 6R^cM + 11R^oM + 2RM^3, \tag{4.8.7}$$

where equation (4.8.7) is directly taken from Table 3, and the values of X, V^o, and V^c are substituted for. Additionally, we have used the fact that:

$$V^o + V^c = 2(R^o + R^c)M^3 + 2R^oM = 2RM^3 + +2R^oM. \tag{4.8.8}$$

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 (4.3.1) and (4.3.2). We need M additions per open chain for the first summation in equation (4.3.1). 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 (4.6.1) or (4.6.2) must then be tested. To do so, we need to find the smallest and 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:

$$E_s(I) = \max_s \frac{(\bar{N}_s^{(I)} - \underline{N}_s^{(I)})}{\underline{N}_s^{(I)}} = \max_s \frac{\bar{N}_s^{(I)}}{\underline{N}_s^{(I)}} - 1 = \frac{\max_s \bar{N}_s^{(I)}}{\min_s \underline{N}_s^{(I)}} - 1. \tag{4.8.9}$$

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

$$E_r(\bar{\lambda}_{o,r}^{(I)},\underline{\lambda}_{o,r}^{(I)}) = \frac{\max_r \bar{\lambda}_{o,r}^{(I)}}{\min_r \underline{\lambda}_{o,r}^{(I)}} - 1. \tag{4.8.10}$$

The search for the upper or lower bound values in equation (4.8.9) 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 (4.8.10). Thus, the time complexity of the approximation for I iterations can be expressed as:

$$T(R^o,R^c,M,I) = IT(R^o,R^c,M) + IR^oM + 2I(\max(R^o,M) - 2). \tag{4.8.11}$$

---

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

Figure 6 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 6, as in Figure 5, 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).

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:

$$\Gamma_1 = \Gamma_1(R^o,R^c,M) = 4M(K_1+1)^{R^c}\left[(R^o+R^c)(K_1+1)^{R^o} - R^c\right] - 8R^oM - 3R^cM, \qquad (4.8.12)$$

and

$$\Gamma_I = \Gamma_I(R^o,R^c,M,I) = T(R,M) - IT(R^o,R^c,M) - IR^oM - 2I(\max(R^o,M) - 2). \qquad (4.8.13)$$

In equation (4.8.12), as in the case of the space complexity, we can show that for all nontrivial cases $\Gamma_1>0$. However, in equation (4.8.13) the sign and the magnitude of the reduction depends on yet another parameter, which is the number of iterations. Although we can prove that substantial reduction in time complexity is achieved for very large networks[4], and for the intermediate size networks, the relation given by equation (4.8.13) 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. Experimental Results

In this section we examine the performance of our algorithm by applying it to an example model, shown in Figure 3. In order to evaluate the efficiency and the accuracy of our approximation, we have chosen as example model one that is small enough to be exactly solved. The system modeled in Figure 3 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 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 4. 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 also a 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 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

[4] Networks with large chain populations.

queueing network models. Thus, we expect that error for this performance measure to be the lowest. Figure 8 shows the error for the first iteration and for the case where the stopping rule bound, e_s, is chosen to be small. The error in both cases is very small. Figure 9 and 10 show the mean queue length versus the number of users. Figure 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 over 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 13 - 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 performance is indeed very satisfactorily low. Figure 22 which shows the error 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 17 and 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 17 and 18. Figure 9 is based on the error criterion given in (4.6.6) and Figure 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.

Figures 23 - 30 show the curves for the case of 3 open chains and compare it to 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 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 iteration 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 discussed in the asymptotic properties in section 4.5., 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 more understood by looking at the utilization of each service center due to jobs belonging to each open chain. From equations (4.3.5) and (4.3.6), it is clear that at very high values of ρ_{rs}^{o} the values of external arrival rate estimate become very sensitive to variations in ρ_{rs}. On the other hand, the computational complexities of the algorithm is very sensitive to 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 objective, the 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, equations derived in section 4.7 may be used to roughly estimate the space and time complexities of solving the model. 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 (4.3.5) and (4.3.6) suggest that some round off error might occur for extreme values of open chain utilizations. Some scaling algorithm might be needed to avoid this problem. We can apply the algorithm for modeling and performance evaluation of large internetwork-based distributed systems. This is done by a decoupling algorithm [Bah87] that transforms the QN models of configurations such as those shown in Figure 1 by decoupling them into approximate mixed QN submodels that can be solved independently.

6. Conclusions

In this paper we have argued that modeling and evaluating the performance of large distributed systems requires efficient and cost effective approximate algorithms for the solution of large multichain QNs. We have presented an iterative and hierarchical approximate technique for solving multichain QNs with large numbers of chains, service centers, and populations. The technique applies to product form QNs with single server fixed rate service centers and IS service centers. Extensions to certain load dependent QNs are possible. Our approximation 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 for exact methods. The approximate solutions are asymptotically correct. Our algorithm is iterative, but 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 new error criteria that we believe are more realistic and permits a fair comparison among exact and accurate techniques. Finally, the algorithm can be easily implemented with minimal effort.

7. Acknowledgements

I wish to express my deep appreciation and gratitude to Professor Domenico Ferrari for his invaluable advice and support.

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*, Berkeley, January 1987.

[Bah87]

H. R. Bahadori, A Hierarchical Modeling and Hybrid Simulation Technique for Large Distributed Systems, *IASTED International Conference on Applied Identification, Modelling, and Simulation*, New Orlean, November 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.

[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.

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

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

[Bur85]
R. L. Burden and J. D. Faires, *Numerical Analysis (3rd edition)*, Prindle, Weber & Schmidt, 1985.

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

[Cha75]
K. M. Chandy, U. Herzog, L. Woo and F. G. Placios, 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 ACM 23, 10* (October 1980), 573-583.

[Eag86]
D. Eager and K. C. Sevcik, Boud Hierarchies for Multiple-Class Queueing Networks, *Journal of ACM 33, No. 1* (January 1986), 179-206.

[Gel80]
E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*, Academic Press, 1980.

[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 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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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*, 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.

APPENDIX A

In this appendix we give a proof that the alternative forms of the normalizing constants of mixed networks as given by (2.4.9), (2.4.10), and (2.4.12) are the same:

Observe that:

$$\left(\sum_{i\varepsilon R_o(s)} \rho_{is}^o\right)^{k_s^o} = \sum_{\mathbf{k}_s^o} \binom{k_s^o}{k_{1s}^o, k_{2s}^o, \ldots, k_{R^o s}^o} \prod_{i\varepsilon R_o(s)} (\rho_{is}^o)^{k_{is}^o} = \tag{A.1}$$

$$= \sum_{\mathbf{k}_s^o} \frac{k_s^o!}{k_{1s}^o! k_{2s}^o! \cdots k_{R^o s}^o!} \prod_{i\varepsilon R_o(s)} (\rho_{is}^o)^{k_{is}^o} =$$

$$= \sum_{\mathbf{k}_s^o} k_s^o! \prod_{i\varepsilon R_o(s)} \frac{(\rho_{is}^o)^{k_{is}^o}}{k_{is}^o!}$$

where the second equality holds by the multinomial theorem. Substituting for $\left(\sum_{i\varepsilon R_o(s)} \rho_{is}^o\right)^{k_s^o}$ in equation (2.4.9) results in (2.4.10).

To show that equation (2.4.12) can be derived from that of (2.4.9), we rewrite (2.4.9) in the following form:

$$G(\Lambda_o,\mathbf{K}_c) = \sum_{\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)} \prod_{s=1}^{M} \left(\sum_{i\varepsilon R_o(s)} \rho_{is}^o\right)^{k_s^o} \frac{(k_s^c+k_s^o)!}{k_s^o!} \prod_{j\varepsilon R_c(s)} \frac{(\theta_{js}^c s_{js}^c)^{k_{js}^c}}{k_{js}^c!} \frac{(1-\rho_s^o)^{k_s^c}}{(1-\rho_s^o)^{k_s^c}}, \tag{A.2}$$

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 $\prod_{j\varepsilon R_c(s)} (1-\rho_s^o)^{k_{js}^c} = (1-\rho_s^o)^{k_s^c}$, we can write:

$$G(\Lambda_o,\mathbf{K}_c) = \sum_{\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)} \prod_{s=1}^{M} (\rho_s^o)^{k_s^o} (1-\rho_s^o)^{k_s^c} \frac{(k_s^c+k_s^o)!}{k_s^o!} \prod_{j\varepsilon R_c(s)} \frac{\left(\theta_{js}^c \frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{A.3}$$

We multiply the right hand side of equation (A.3) by a term equal to $\frac{k_s^c!}{k_s^c!} = 1$. This results in:

$$G(\Lambda_o,\mathbf{K}_c) = \sum_{\mathbf{k}\varepsilon S(\Lambda_o,\mathbf{K}_c,M)} \prod_{s=1}^{M} (\rho_s^o)^{k_s^o} (1-\rho_s^o)^{k_s^c} \frac{(k_s^c+k_s^o)!}{k_s^o! k_s^c!} k_s^c! \prod_{j\varepsilon R_c(s)} \frac{\left(\theta_{js}^c \frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{A.4}$$

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 the vector $\mathbf{k}^o$ varies from zero to infinity. We can write (A.4) in the following form:

$$\sum_{\mathbf{k}_s^c} \sum_{k_1^o=0}^{\infty} \cdots \sum_{k_M^o=0}^{\infty} \prod_{s=1}^{M} (\rho_s^o)^{k_s^o} (1-\rho_s^o)^{k_s^c} \frac{(k_s^c+k_s^o)!}{k_s^o! k_s^c!} k_s^c! \prod_{j\varepsilon R_c(s)} \frac{\left(\theta_{js}^c \frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}. \tag{A.5}$$

For all value of s, the terms $k_s^c! \prod_{j\varepsilon R_c(s)} \frac{\left(\theta_{js}^c \frac{s_{js}^c}{1-\rho_s^o}\right)^{k_{js}^c}}{k_{js}^c!}$ are independent of open chain states. From binomial theorem we know:

$$\sum_{k_s^o=0}^{\infty} (\rho_s^o)^{k_s^o}(1-\rho_s^o)^{k_s^c}\frac{(k_s^o+k_s^c)!}{k_s^o!k_s^c!} = \left[(\rho_s^o) + (1-\rho_s^o)\right]^{k_s^o+k_s^c} = 1, \qquad \text{(A.6)}$$

Therefor, except for the case when $k_s^c=0$, all the summations over the open chain state vector add up to 1. When k_s^c term are zero for each s the summation in (A.6) is the same as $\sum_{k_s^o}\rho_s^{o\,k_s^o}=0=\frac{1}{1-\rho_s^o}$ (assuming that $\rho_s^o<1$ for all s). Using these results the desired form of (2.4.12) can easily be obtained.

Figure 1. Internetwork-Based Distributed System Configuration

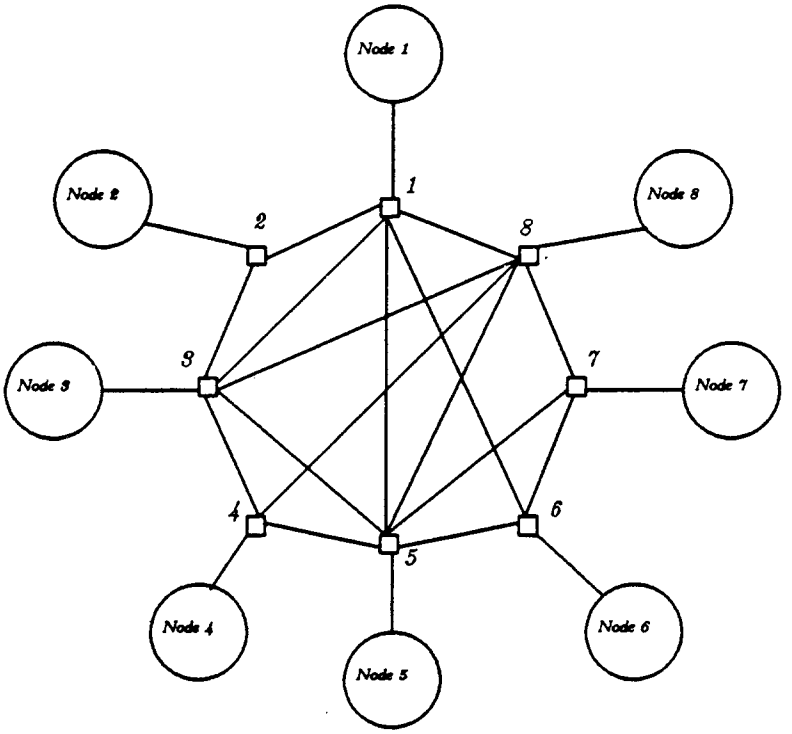


Figure 2 A Local Area Network-Based Distributed System Configuration

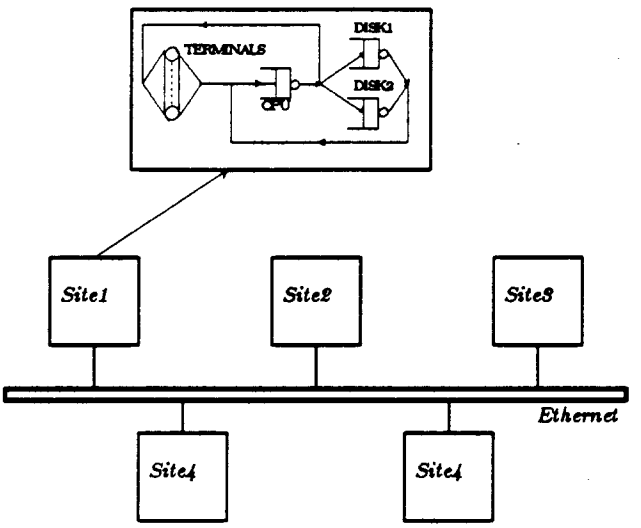
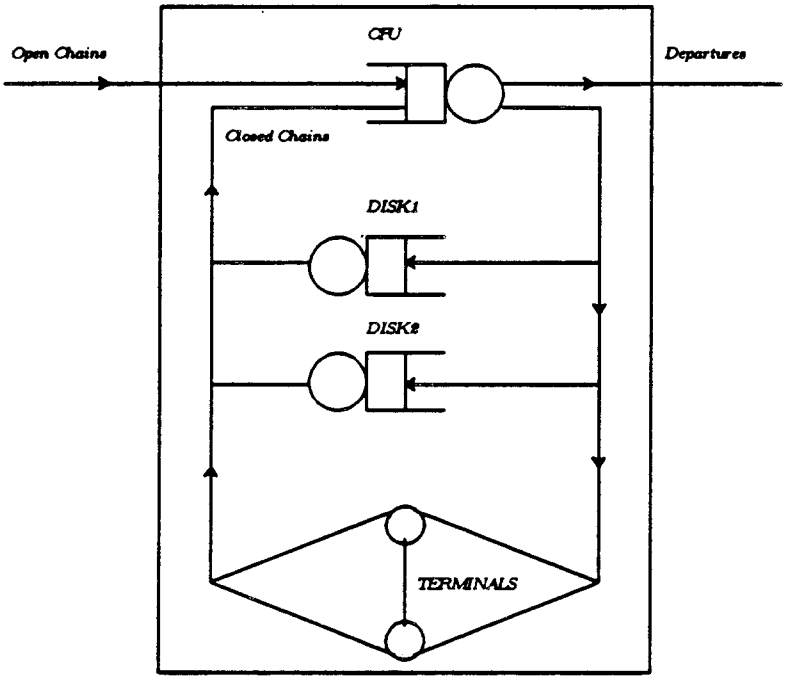


Figure 3 Example Model Used for Experimental Results



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 4. The Parameters of the Network in Figure 3.

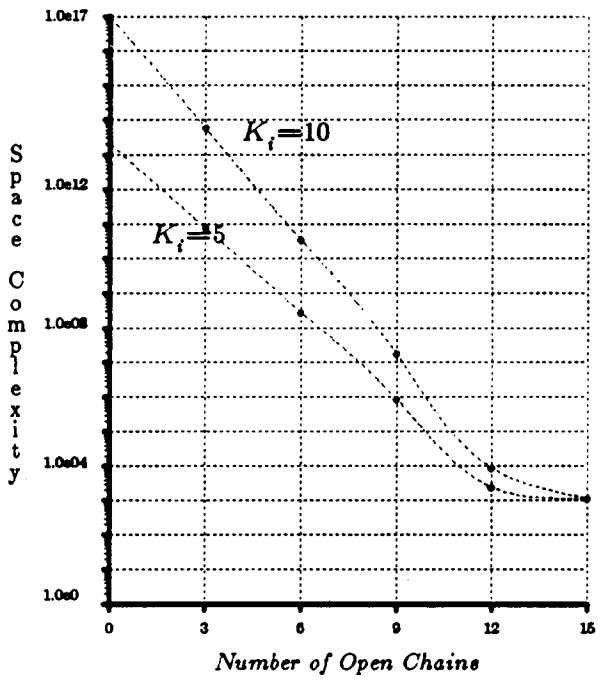


Figure 5

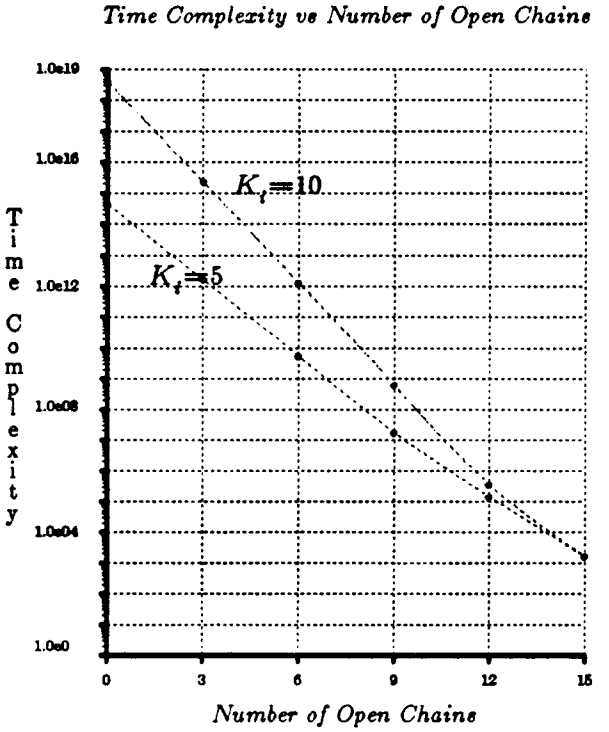


Figure 6

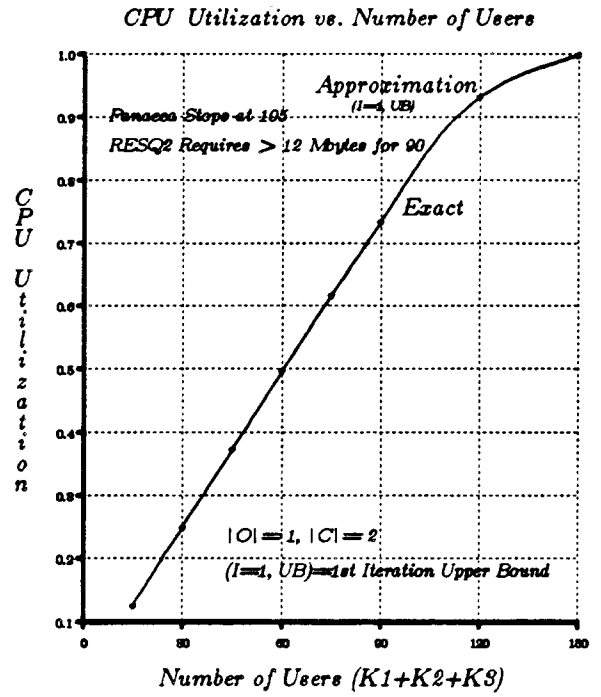


Figure 7

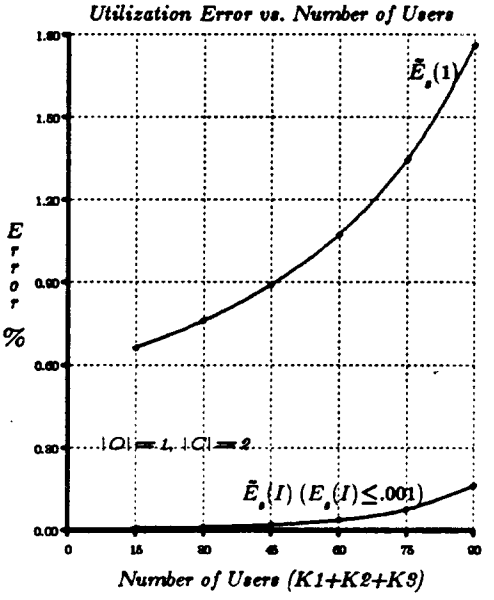


Figure 8

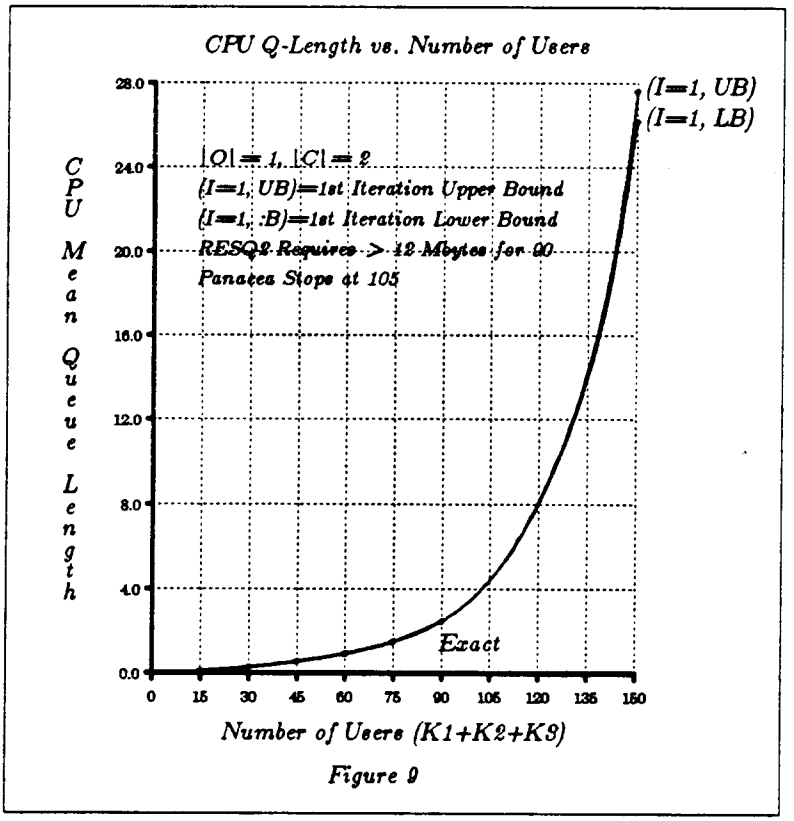


Figure 9

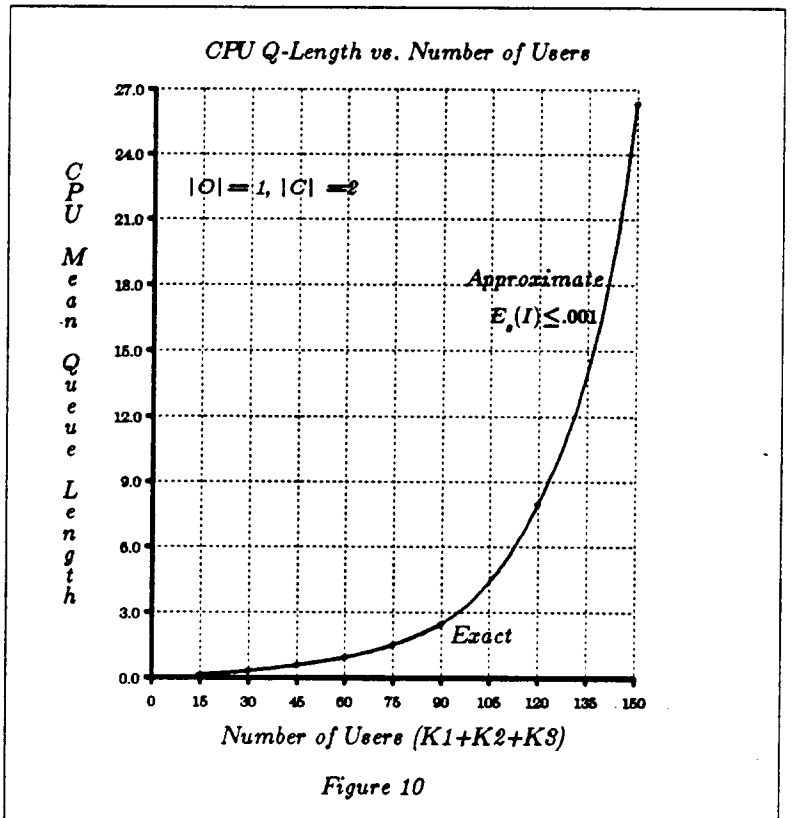


Figure 10

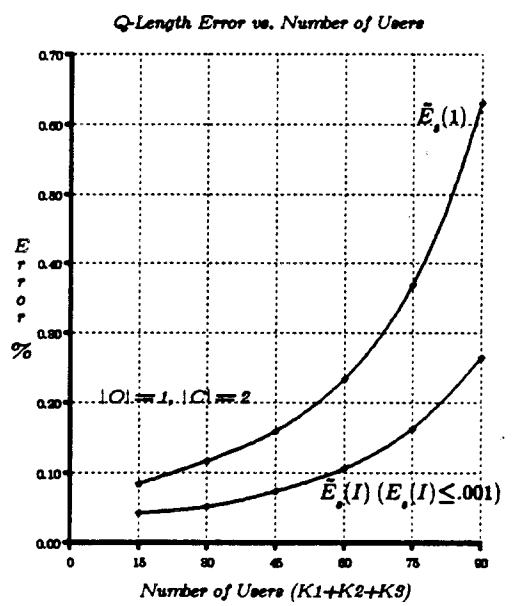


Figure 11

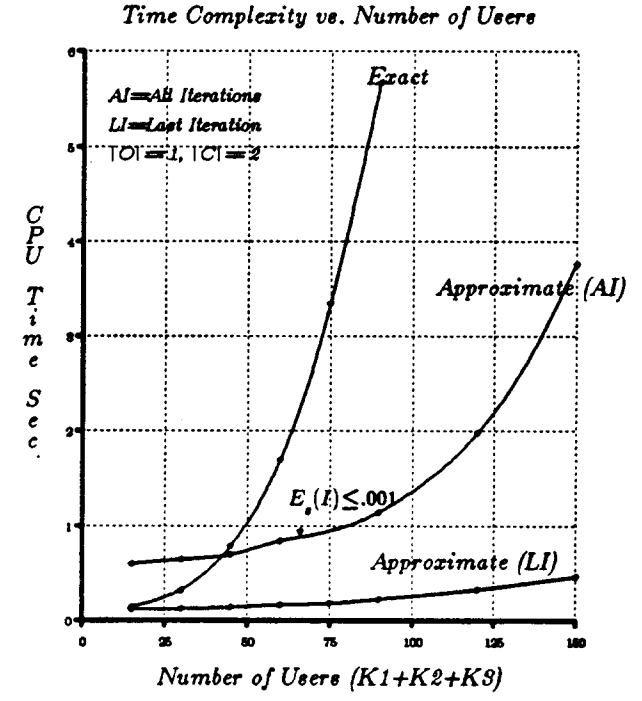


Figure 12

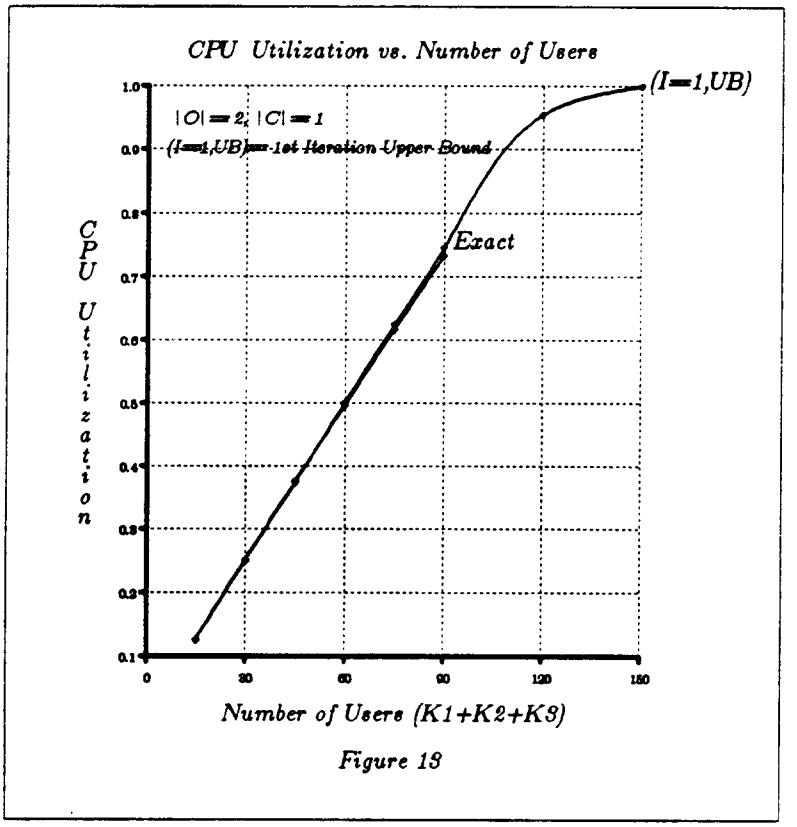


Figure 13

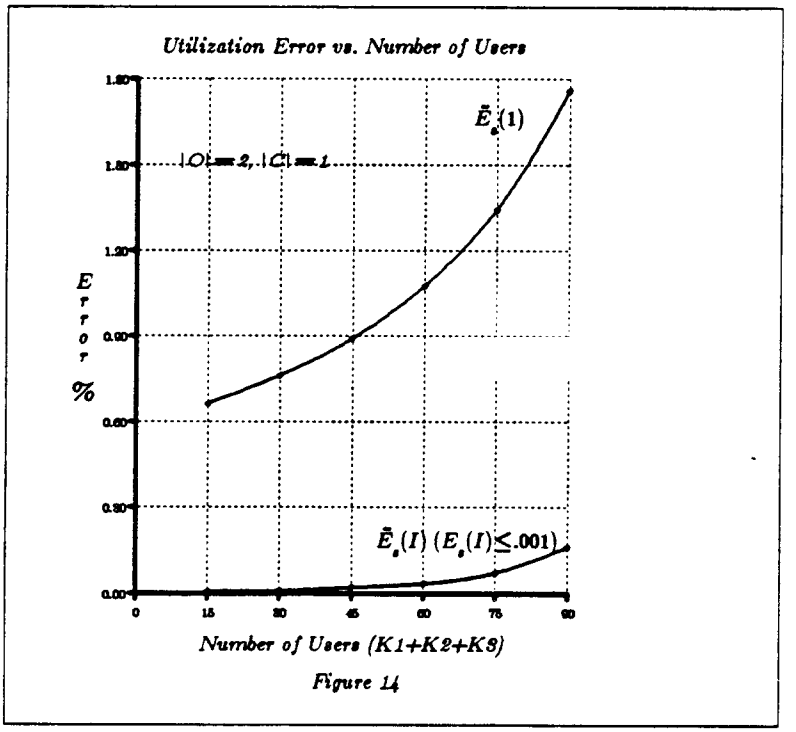


Figure 14

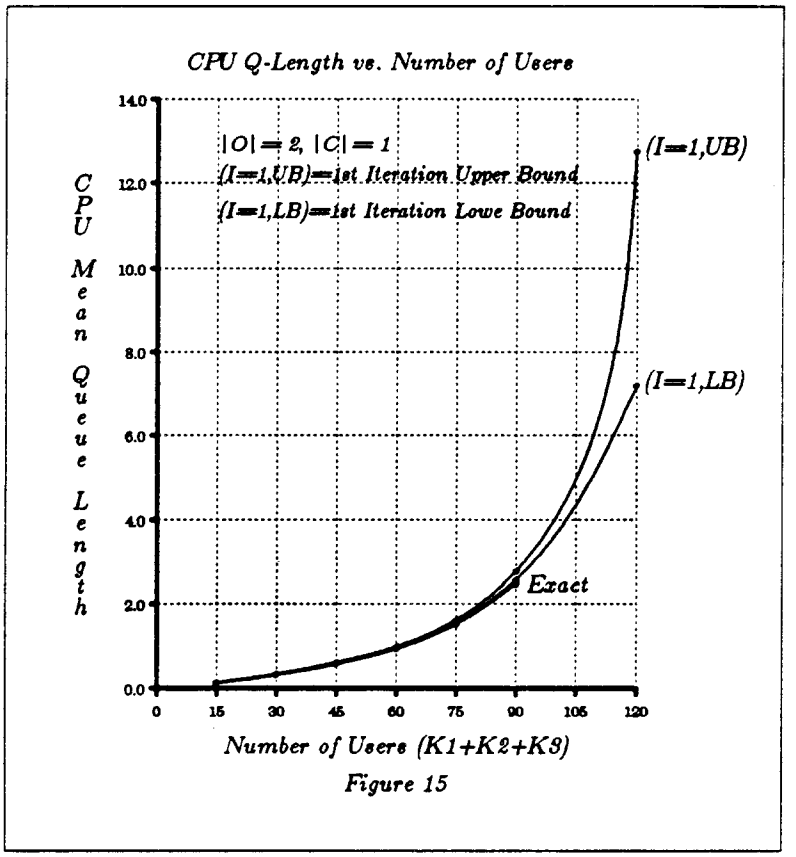


Figure 15

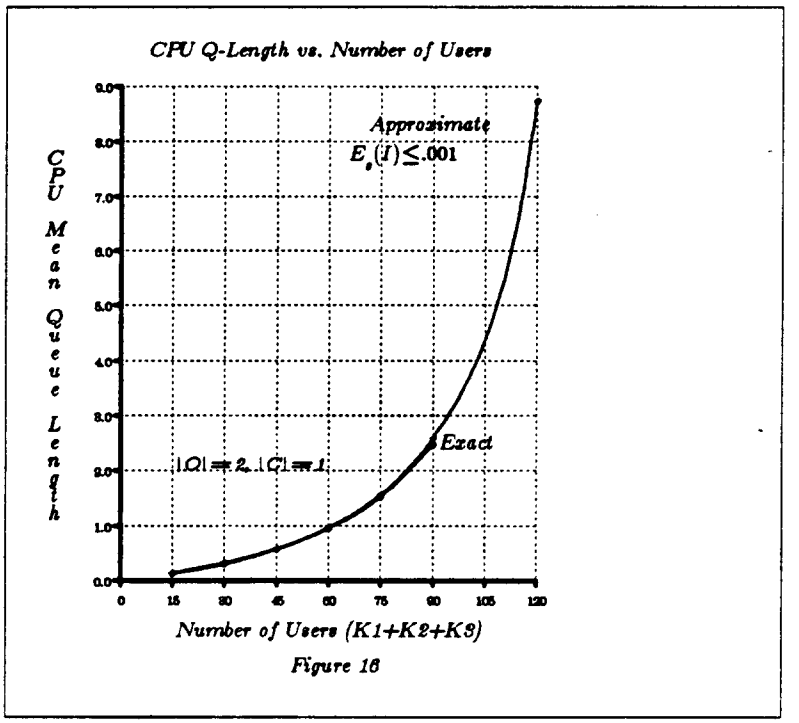


Figure 16

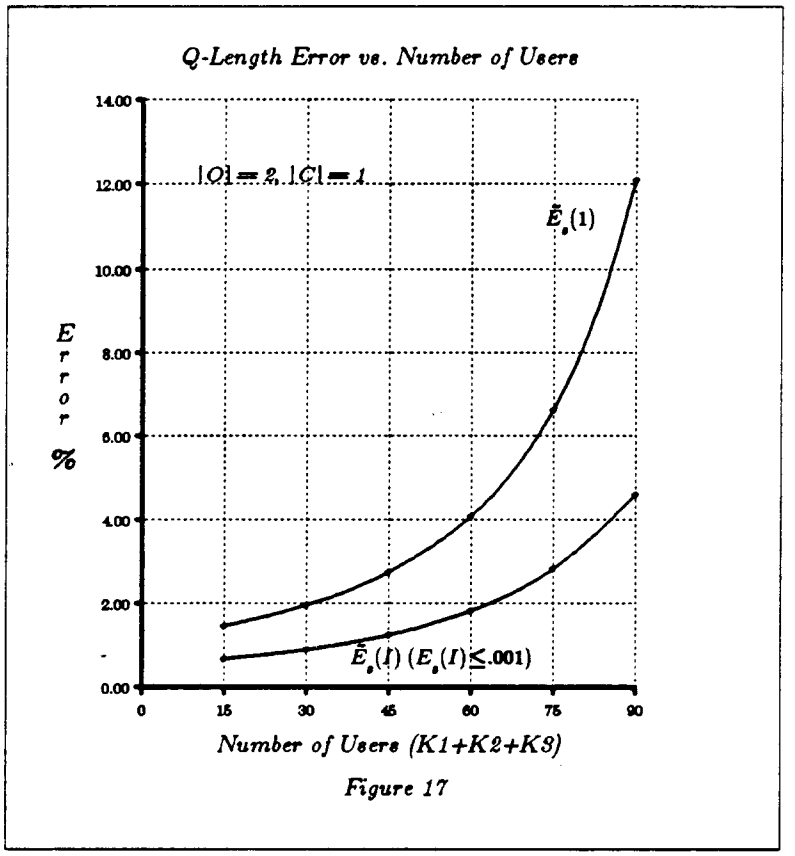


Figure 17

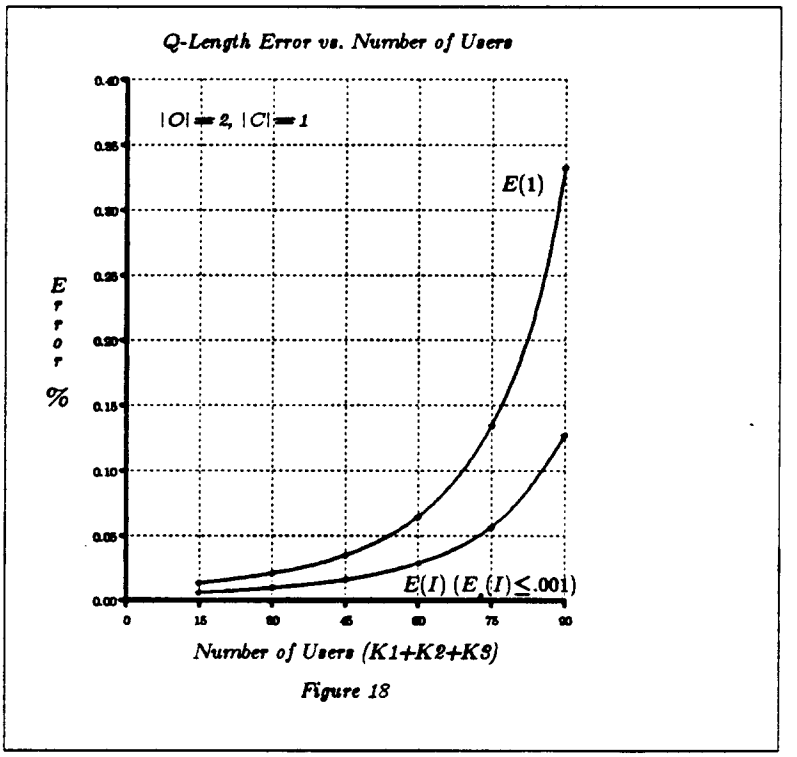


Figure 18

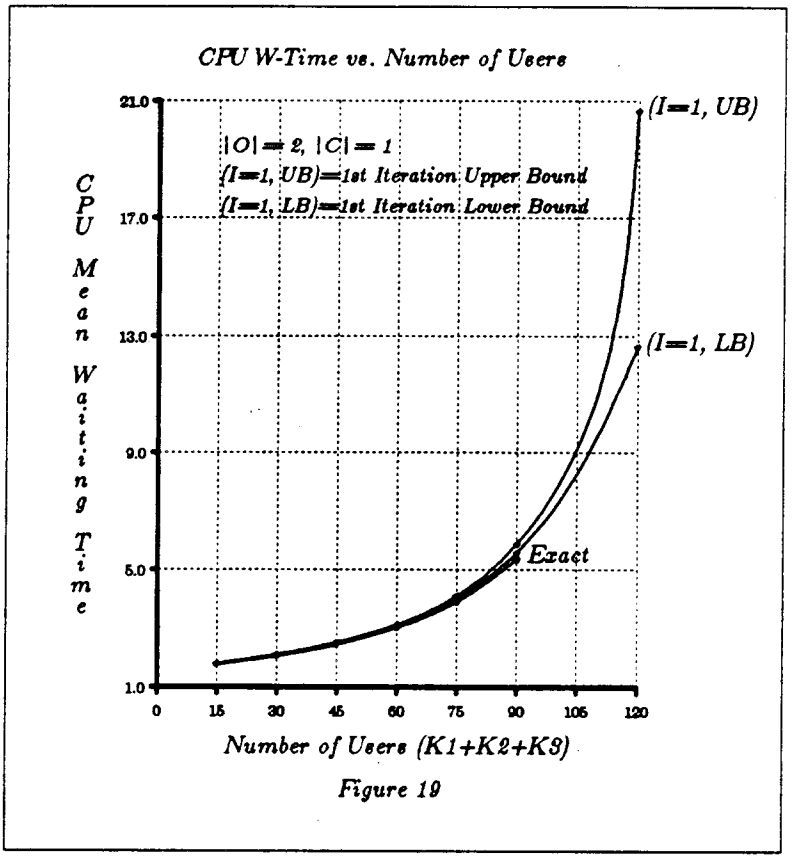


Figure 19

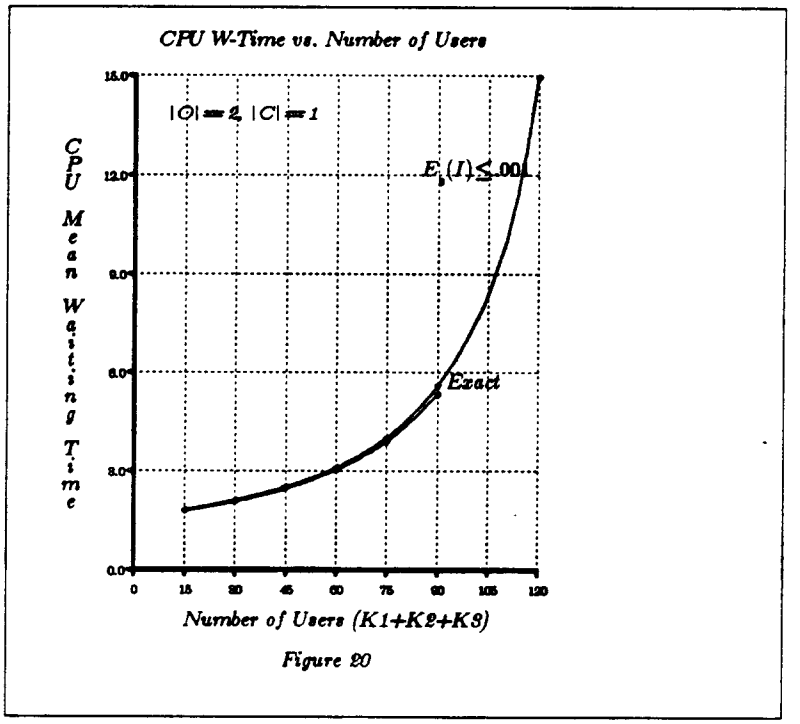


Figure 20

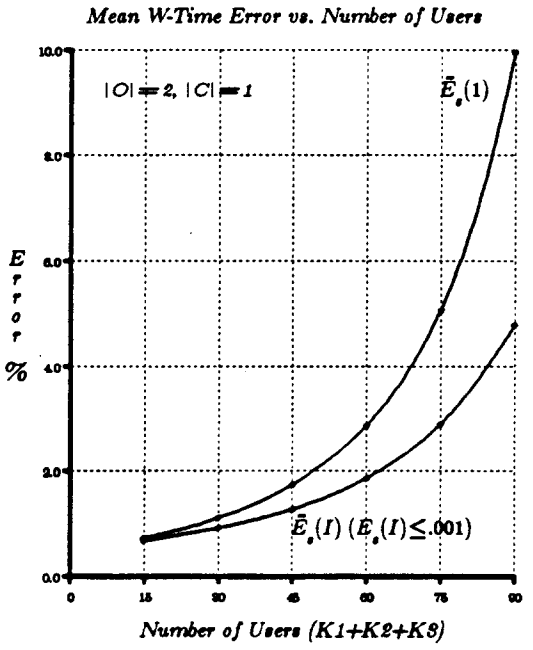


Figure 21

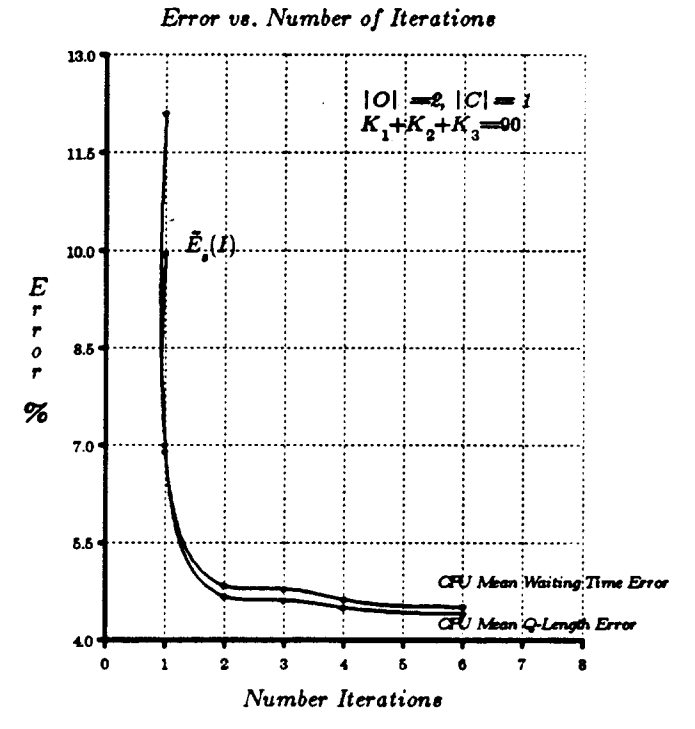


Figure 22

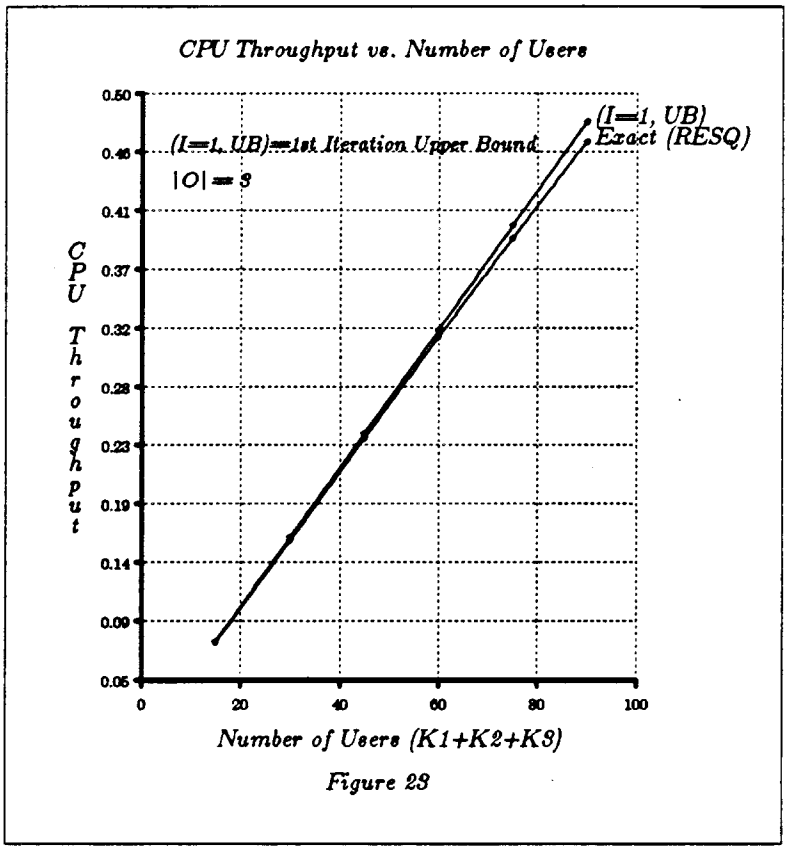


Figure 23

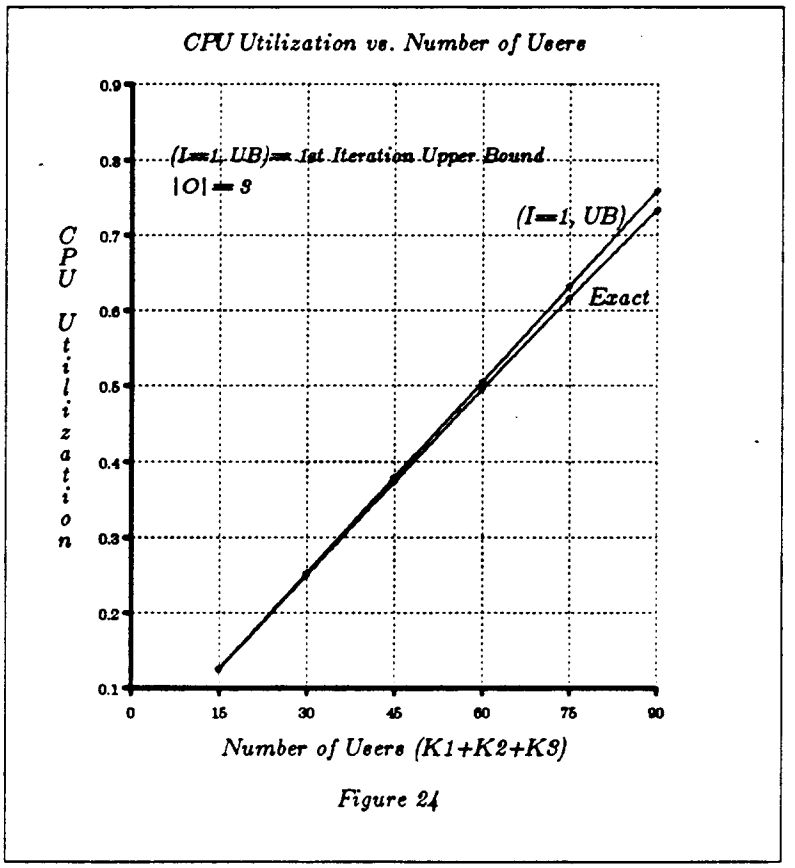


Figure 24

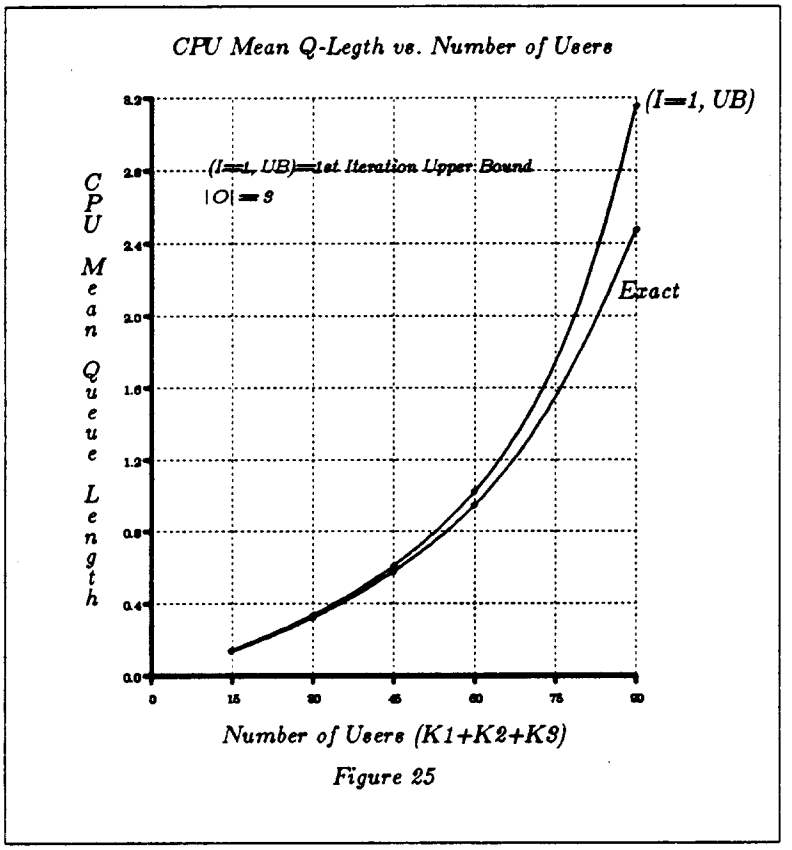


Figure 25

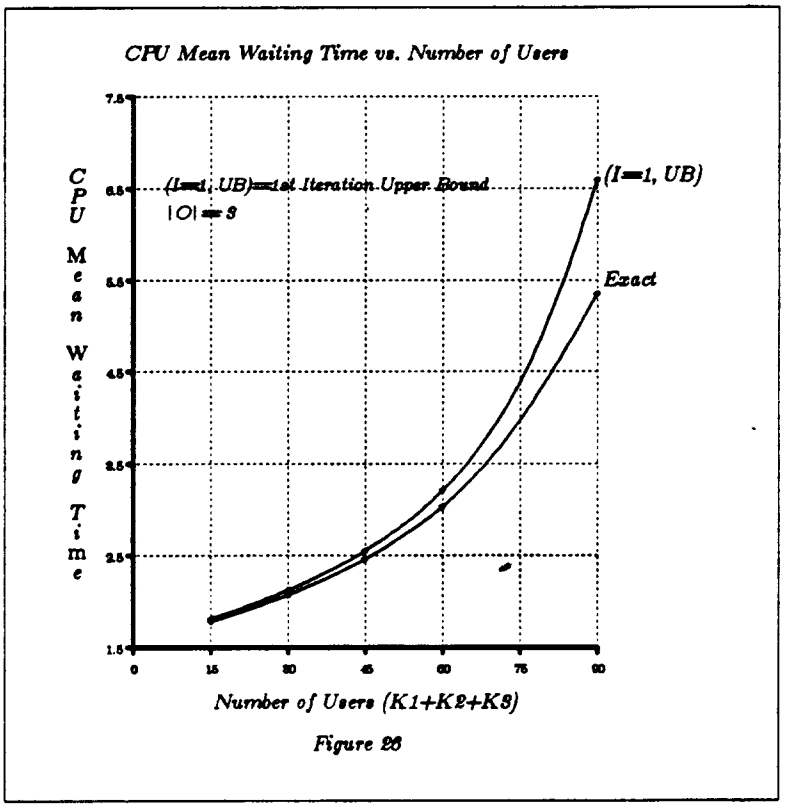


Figure 26

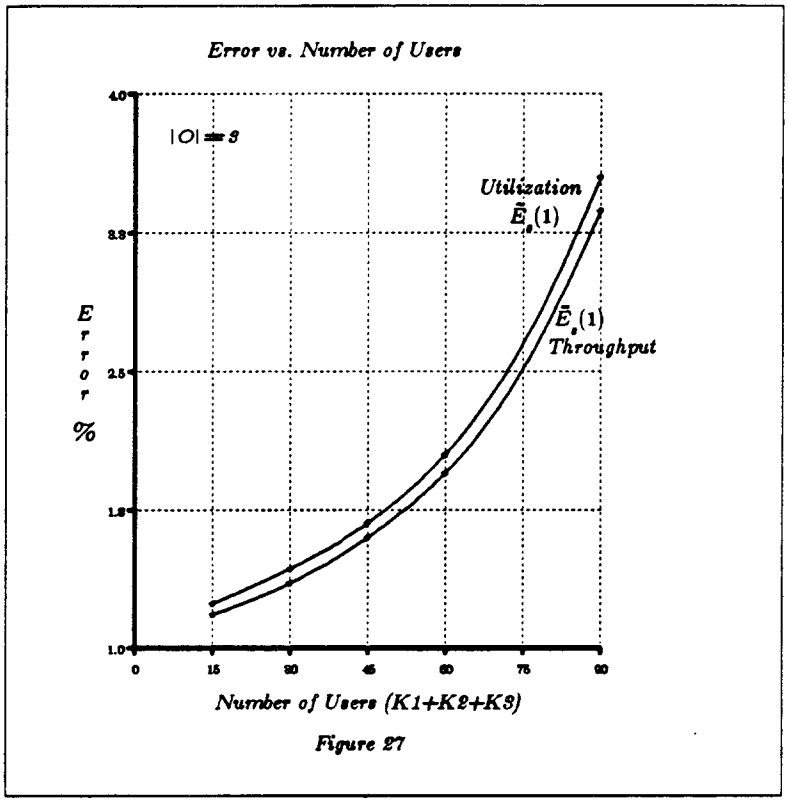


Figure 27

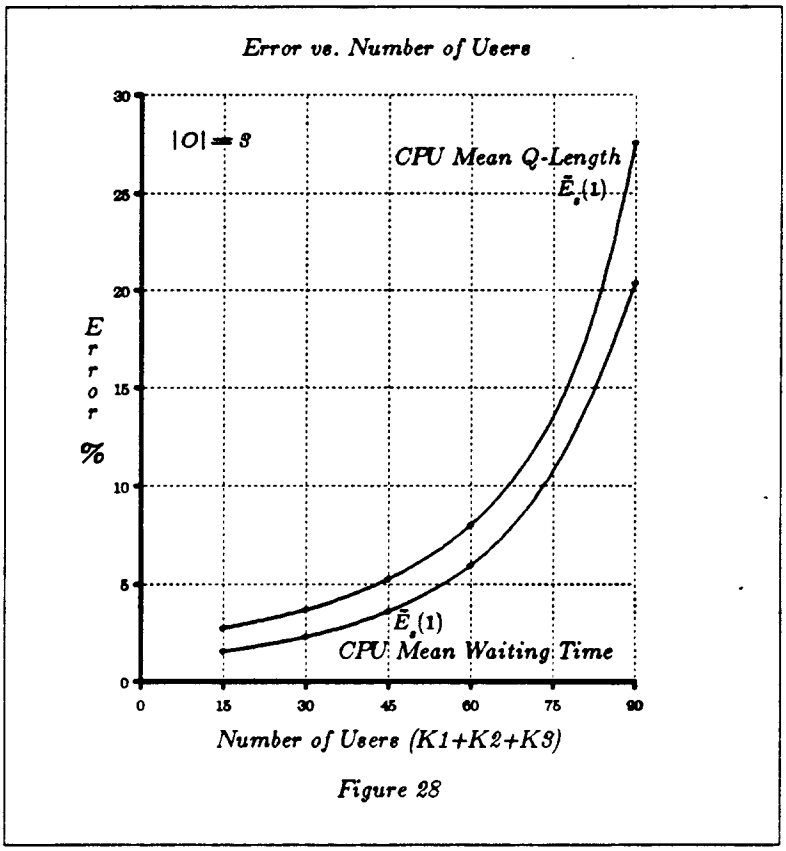


Figure 28

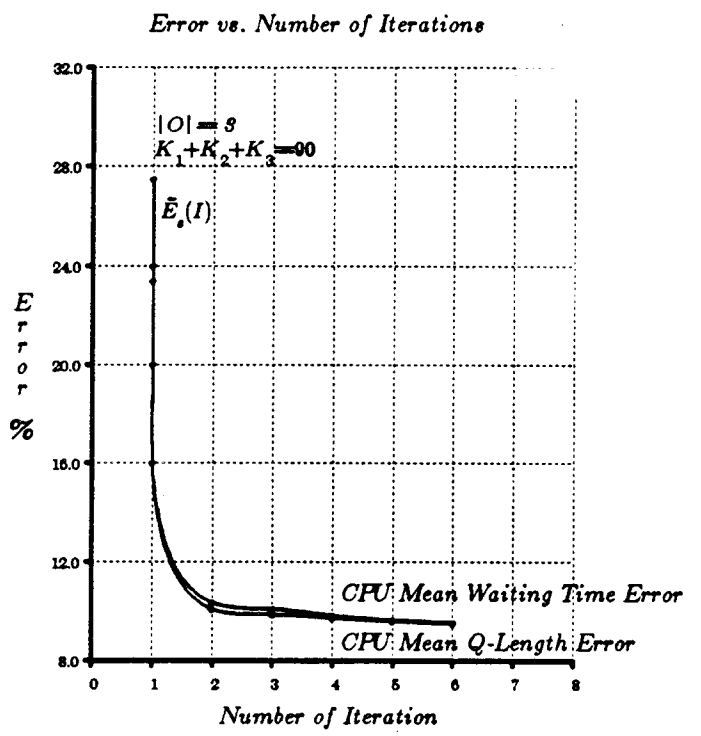


Figure 29

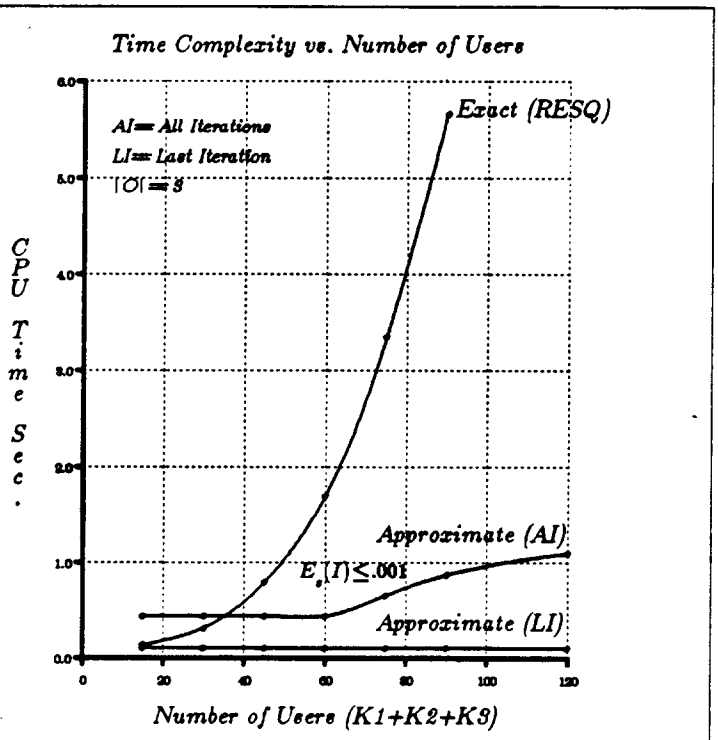


Figure 30