

UNIVERSITY OF CALIFORNIA

Los Angeles

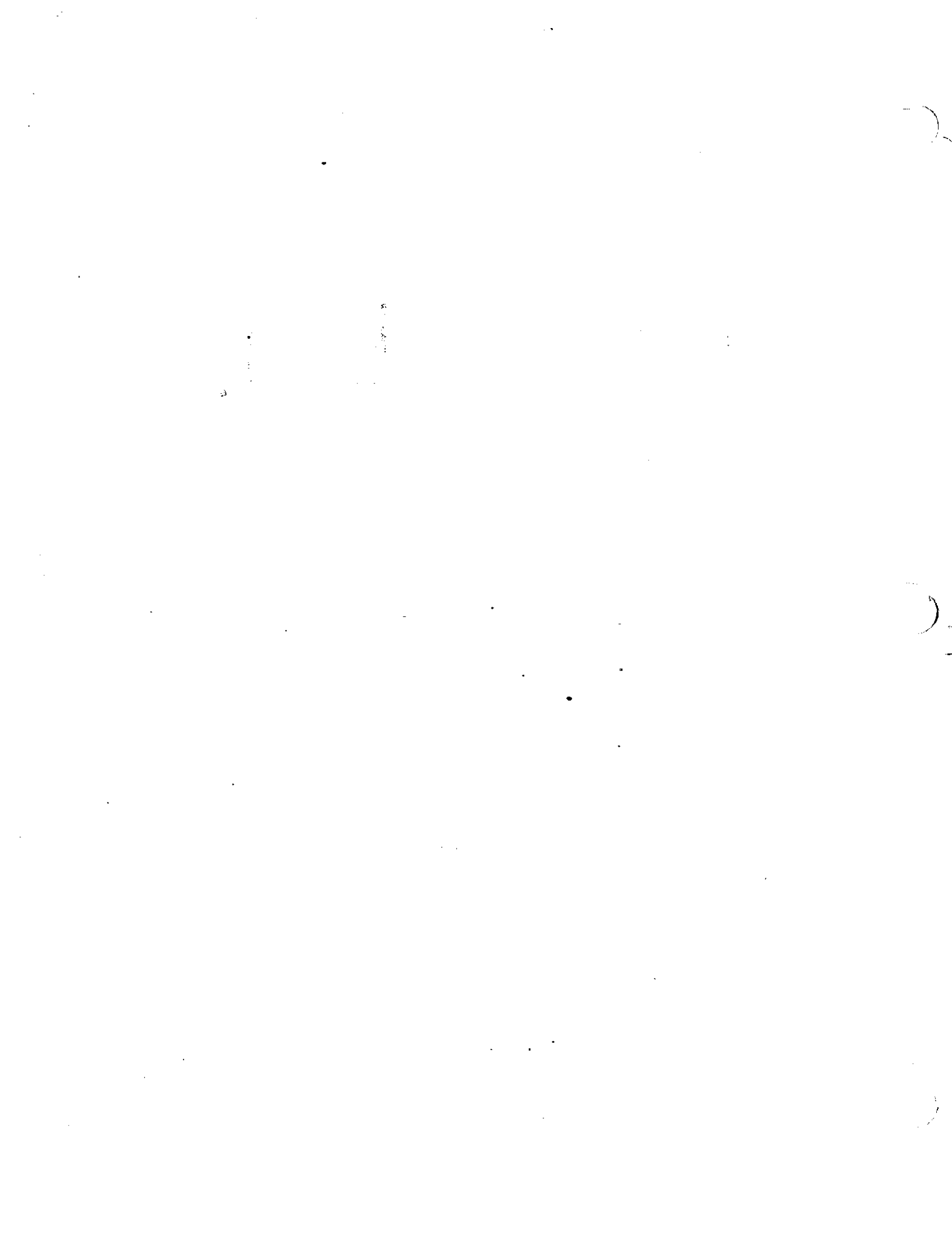
**Algorithms for Queueing Network Analysis
of Distributed Systems**

**A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science**

by

Edmundo Albuquerque de Souza e Silva

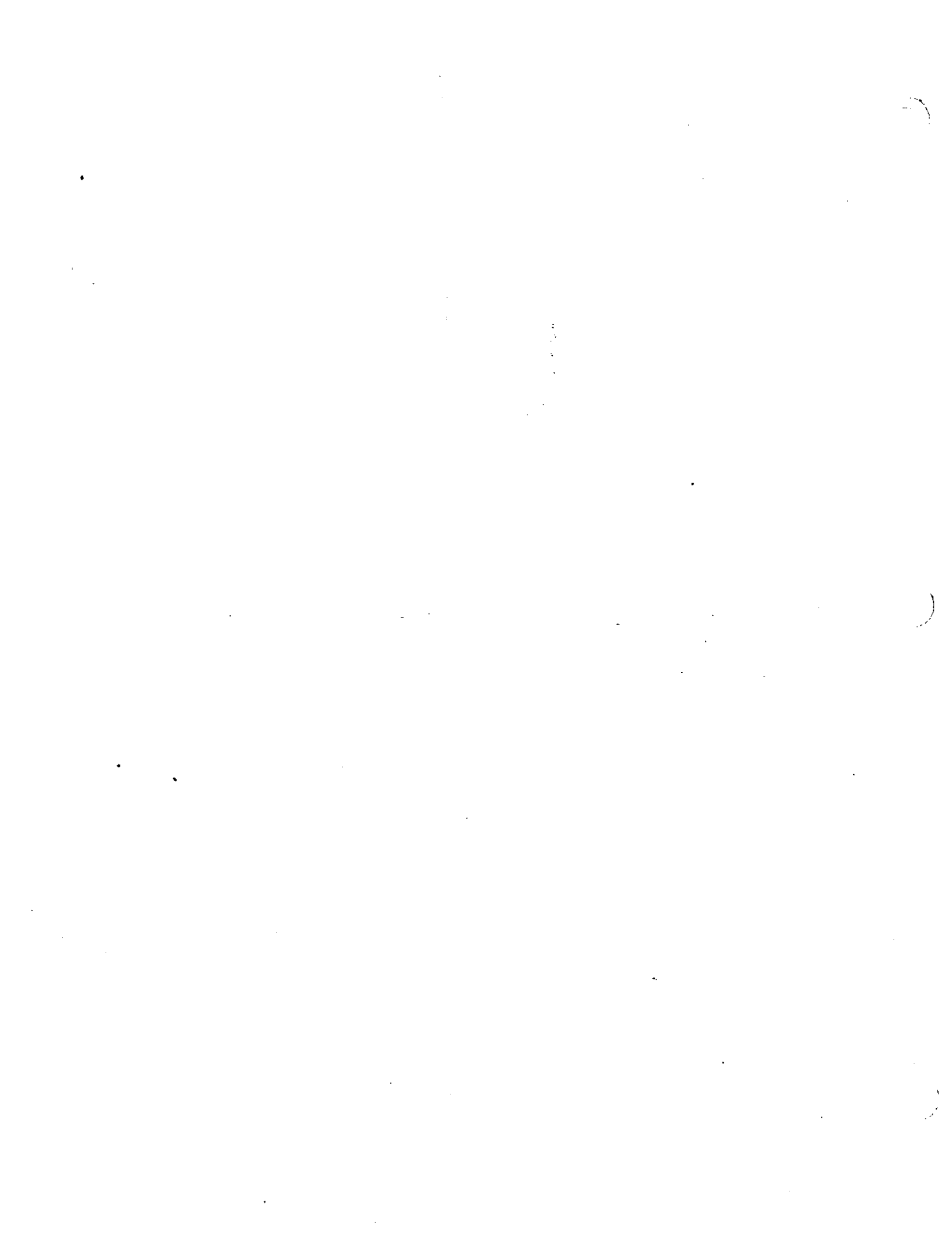
1984



**ALGORITHMS FOR QUEUEING NETWORK ANALYSIS
OF DISTRIBUTED SYSTEMS**

Edmundo Albuquerque de Souza e Silva

**November 1984
Report No. CSD-840040**



© copyright by

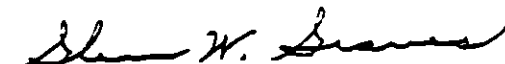
Edmundo Albuquerque de Souza e Silva

1984

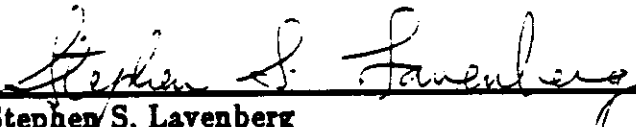
The dissertation of Edmundo Albuquerque de Souza e Silva is approved.




Mario Gerla




Glenn W. Graves



Stephen S. Lavenberg



Steven A. Lippman



Gerald J. Popek



Richard R. Muntz, Committee Chair

University of California, Los Angeles

1984

TABLE OF CONTENTS

	page
1 INTRODUCTION.	1
2 OVERVIEW OF APPROXIMATE METHODS FOR SOLVING QUEUEING NETWORK MODELS.	7
2.1. Introduction.	7
2.2. Non-Iterative Methods.	11
Based on heuristic extensions to the MVA equations.	11
Based on Hierarchical Decomposition and/or Norton's Theorem.	13
Other Non-Iterative Methods.	15
2.3. Iterative Methods - a Unified View.	15
2.3.1. Some Existing Iterative Methods.	18
Bard's Approximation.	18
Schweitzer's Approximation.	19
Linearizer.	19
Marie's Method.	21
Jacobson and Lazowska's Methods for Simultaneous Resource Possession.	22
Modeling Asynchronous Tasks.	26
2.3.2. Discussion.	26
2.4. Conclusions.	30
3 A CLUSTERING APPROXIMATION TECHNIQUE FOR PRODUCT FORM QUEUEING NETWORK MODELS WITH A LARGE NUMBER OF CHAINS.	33
3.1. Introduction.	33
3.2. A Simple Homogeneous Network.	36
3.3. A Clustering Approximation Technique for Product Form Queueing Network Models with Single and Multiple Server Service Centers.	46
3.3.1 Product Form Networks with SSFR and IS Service Centers	49
3.3.2 Product Form Networks with Multiple Server (MS) Service Centers.	55
3.3.3. The Algorithm.	57
3.3.4. The Choice of Subnetworks.	61
3.3.5. Empirical Results	64
3.4. Conclusions.	78
4 A CLUSTERING APPROXIMATION TECHNIQUE FOR NON-PRODUCT FORM QUEUEING NETWORK MODELS.	79
4.1. Introduction.	79
4.2. A Representation of a CSMA-CD Channel in a LOCUS-type Model.	81
4.3. Simultaneous Resource Possession. Dedicated Passive Resources.	84
4.3.1. The Algorithm.	92
4.3.2. Empirical Results.	98
4.4. Simultaneous Resource Possession: Shared Passive Resources.	103
4.4.1. An Approximation for Closed Queueing Network Models	

with FCFS Multiple Server Service Centers and Exponential Service Times.	108
4.4.1.1. Empirical Results for FCFS Multiple Server Centers.	113
4.4.2. Shared Passive Resources. Empirical Results.	116
4.5. Conclusions.	123
5 LOAD BALANCING IN DISTRIBUTED SYSTEMS.	126
5.1. Introduction.	126
5.2. Problem Description.	129
5.3. The Model.	131
5.4. The Solution Approach.	134
5.5. The Algorithm.	141
5.6. Examples.	145
5.7. Conclusions.	149
6 EXTENSIONS TO THE LOAD BALANCING ALGORITHM.	151
6.1. Introduction.	151
6.2. Jobs with Multiple Tasks.	152
6.2.1. Problem Description and Model.	152
6.2.2. The Solution Approach.	153
6.2.3. Example.	157
6.3. An Approximate Algorithm for the Routing Problem in a Queueing Network Model with Multiple Closed Chains.	159
6.3.1. Problem Description.	159
6.3.2. The Approximate Algorithm.	160
The Algorithm.	163
6.3.3. Example.	164
6.4. Other Possible Extensions and Conclusions.	166
7 CONCLUDING REMARKS.	169
Appendix 1 PROOF OF ASSUMPTION 3.2b.	173
Appendix 2 THE CLUSTERING ALGORITHM.	177
Appendix 3 THE CONVEXITY OF THE DELAY FUNCTION OF CHAPTER 5.	180
Appendix 4 MOMENTS OF QUEUE LENGTHS IN QUEUEING NETWORKS	188
A4.1. Introduction.	188
A4.2. Recursive Expressions for the Variance and Covariances in Equation (4.18).	188
A4.3. Moments of Queue Lengths in a Queueing Network with Multiple Closed Chains.	190
A4.4. Conclusions.	200
REFERENCES	205

LIST OF FIGURES

	page
Figure 2.1. A central server model with memory constraint.	14
Figure 2.2. Application of Norton's theorem to solve the network of Figure 2.1.	14
Figure 2.3. Example of an iterative procedure.	17
Figure 2.4. The method of surrogate delays.	23
Figure 2.5. "Population constraint" method.	24
Figure 2.6. A subnetwork and possible representations of its complement.	28
Figure 2.7. Some approximate methods for the analysis of queueing network models.	32
Figure 3.1. The approximate solution of a homogeneous network with many sites.	44
Figure 3.2. Response times versus the number of sites and the limiting result.	46
Figure 3.3. A distributed system with two sites.	48
Figure 3.4. Representation of the effect of the complement for local and foreign chains.	53
Figure 3.5. Subnetwork A and its complement.	54
Figure 3.6. Densities of absolute value of percent errors for LINSUB, LIN and SCH.	68
Figure 3.7. Densities of absolute value of percent errors for MVASUB, for networks with MS centers.	71
Figure 3.8. A queueing model of a distributed system.	72
Figure 3.9. A small packet switching network.	74
Figure 4.1. Average delay in the channel. Message = 1000 bits, MPD = .1 msec.	83
Figure 4.2. Mean response time versus number of sites. Message = 4000 bits, MPD = .1 msec.	84
Figure 4.3. Mean response time versus number of sites. Message = 500 bits, MPD = .05 msec.	85

Figure 4.4. A central server model with memory constraint.	86
Figure 4.5. The CPU/disks subnetwork and the entire network in the saturated case.	87
Figure 4.6. Subnetwork S_1	89
Figure 4.7. Subnetwork S_2	91
Figure 4.8. A single chain visiting 2 passive resources.	92
Figure 4.9. Representation of the effect of the complement for local unconstrained and constrained chains.	97
Figure 4.10. Subnetworks for the first example. Dedicated passive resources.	99
Figure 4.11. Second example (one chain network). Dedicated passive resources.	101
Figure 4.12. Subnetworks for the second example.	101
Figure 4.13. Subnetwork for the first example. Shared passive resources.	118
Figure 4.14. Subnetworks for the second example. Dedicated and shared passive resources.	121
Figure 5.1. A distributed system.	130
Figure 5.2. The model.	131
Figure 5.3. Example. A distributed computer system with 3 sites.	148
Figure 5.4. Percentage of foreign jobs processed at site 2.	148
Figure 5.5. Relative difference (%) of the overall delays.	149
Figure 6.1. Possible paths of chain ν jobs with only one task.	154
Figure 6.2. Possible paths of chain ν jobs with two tasks.	155
Figure 6.3. A multiple chain closed network. Example of section 6.3.3.	165
Figure A3.1. Number of open chain customers at site 1 as a function of open chain throughput.	186
Figure A3.2. Number of open chain customers at site 2 as a function of open chain throughput.	187

ACKNOWLEDGEMENTS

It is virtually impossible for someone to successfully conclude a Ph.D. program without the help and support of others. I am no exception. I would like to express my appreciation to my doctoral committee consisting of Professors Mario Gerla, Glenn W. Graves, Stephen S. Lavenberg, Steven A. Lippman, Richard R. Muntz and Gerald J. Popek. I am deeply grateful to Richard Muntz, my advisor and friend, who was always ready to help me overcome the difficulties throughout the course of this research. I was also fortunate to work closely with Mario Gerla and Stephen Lavenberg. I thank them for all the valuable comments and suggestions.

I am indebted to Gerald Estrin who guided me through the early stages of my work. Personal thanks go to Bertram Bussell who gave me his full support since the beginning of the program and made me feel at home in this foreign country.

I will never forget all my good friends in the Computer Science Department. It was fun to be part of the SARA/IDEAS group and to work with Daniel Berry, Dorab Patel, Dorothy Landis, Rami Razouk, Mary Vernon, Duane Worley and all the other members. My special thanks go to my friends who participated in the "modeling seminar", Joseph Betser, Steven Berson, Hak-Wai Chan, Richard Gail, Paulo Rodrigues, Behrokh Samadi, Frank Schaffa and Hideaki Takagi. They provided an enlightening research environment with many exciting discussions. I also thank Baron Gray for his valuable help during

the editing process of this dissertation.

I am thankful to my parents, Lucullo and Alzira de Souza e Silva, for their constant encouragement, and to my son, Edmundo, for the joyful moments we spent together.

My studies at UCLA were financed by a fellowship from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil. Partial support was provided by PUC/RJ-Brazil. The research documented in this dissertation was also supported by IBM 4-442510-47205, SDC-MICRO 442515-59563 and NCR-MICRO 442515-57158-2.

I dedicate this dissertation to my wife, Ursula, who has been so understanding throughout all these difficult years. I would have never been able to achieve my goal without her.

VITA

- August 10, 1952 -- Born, Belem, Para, Brazil
1975 -- A.B., Pontificia Universidade Catolica do Rio de Janeiro, Brazil
1976-1978 -- M.S., Pontificia Universidade Catolica do Rio de Janeiro, Brazil
1978-1979 -- Assistant Professor, Department of Electrical Engeneering, Pontificia Universidade Catolica do Rio de Janeiro, Brazil
1980-1984 -- Research Assistant, Computer Science Department University of California, Los Angeles

PUBLICATIONS

Vernon, M., E. de Souza e Silva, and G. Estrin, "Performance Evaluation of Asynchronous Concurrent Systems: The UCLA Graph Model of Behavior," Performance 83, 1983, pp. 153-171.

de Souza e Silva, E., S. S. Lavenberg, and R. R. Muntz, "A Perspective on Iterative Methods for the Approximate Analysis of Closed Queueing Networks," Proceedings of the International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems, 1983, pp. 191-210.

de Souza e Silva, E., and M. Gerla, "Load Balancing in Distributed Systems with Multiple Classes and Site Constraints," Performance 84, 1984.

ABSTRACT OF THE DISSERTATION

Algorithms for Queueing Network Analysis of Distributed Systems

by

Edmundo Albuquerque de Souza e Silva
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 1984
Professor Richard R. Muntz, Chair

Recently there has been an increasing number of large distributed computer system implementations based on local area networks. In these systems a number of resources (CPU's, file servers, disks, etc) are shared among jobs originating at different sites. Evaluating the performance of such large systems typically requires the solution of a queueing network model with a large number of closed chains, precluding the use of exact solution techniques. Therefore, it is important to develop accurate and cost effective methods for the approximate analysis of closed queueing networks with many chains. We present an approach based on the clustering of chains and service centers. The method is applicable to queueing networks with single server fixed rate, infinite server and multiple server service centers. We present the results obtained when the method is used to solve large queueing network models. Extensive comparison of this method with existing approximation techniques indicates that the approach has better accuracy/cost characteristics.

We study the problem of solving queueing network models with simultaneous resource possession. These models violate product form requirements and cannot be solved exactly unless the network has a very small state space. We developed a technique to approximately solve this class of models. We evaluate the accuracy of the approximation by comparing the analytical results with those obtained from simulation of several models found in the literature.

We investigate the load balancing problem in distributed systems formed by heterogeneous computer sites connected by a communication network and supporting multiple job classes. Jobs arriving at a site may be processed locally or sent to a remote site for execution. Restrictions may apply in the remote dispatcher, so that a class of jobs may be allowed to run only in a subset of all sites in the network. We assume that the communication network as well as each of the computer sites can be modeled as product form queueing networks. Our goal is to find the optimum assignment of jobs to sites so that a weighted sum of response times over classes is minimized. We propose an efficient algorithm for finding the optimum load balancing strategy.

CHAPTER 1

INTRODUCTION.

The past few years have witnessed an increasing number of distributed systems implementations. These systems consist of several processor-memory pair units to entire computers connected to form a cooperating system under a decentralized control scheme. In the past decade we have seen the evolution of multiple processor architectures [Baer76, Ensl77, Saty80]. In these systems, several processor-memory units cooperate under "integrated control". In a typical configuration a multiprocessor system has a single operating system, the processor units share a global memory, they are not very specialized and can perform significant computation individually. The continuous decrease in the cost of processing power and the advances in VLSI design make cost effective the development of powerful computers for individual use. In the late seventies and in the eighties we have seen an increasing use of personal computers, workstations and medium size computers designed to satisfy most of the needs of a single or a small group of users. In parallel, the existence of inexpensive transmission media capable of supporting high data transfer rates (e.g., coaxial cables which can support point to point or broadcast communication in the range of 1 to 10Mbit/sec over a distance of 1 to 10 km without repeaters, and more recent, fiber optic technology) make viable the interconnection of such work-stations and/or more powerful computers forming a local area network. There is a number of applications for local area networks. One such application

with an enormous market is in the area of office automation.

The main goals to be obtained with a distributed system configuration are: (a) Performance improvement: which can be achieved, for example, by efficiently partitioning the work to be done into tasks which can be executed in parallel; by balancing the load in the system; by sharing programs and costly hardware equipment among users, etc. (b) Increase flexibility: usually a distributed system can be easily reconfigured by increasing or decreasing the number of processing units to increase or decrease the capacity of the entire system according to the demand. (c) Reliability and availability: which is achieved by the inherent redundancy of the configuration. The major issues facing the designers of these systems are: task decomposition, interconnection structures, addressing mechanisms, software system structure, interprocessor interface, deadlock avoidance, fault tolerance.

Distributed processing is an area of current strong research interest as we can observe by the numerous articles in the field (e.g., [Thur79, Mari79a, Walk83b]). The potential advantages of distributed processing (high reliability and availability, easy system growth, performance improvement, etc) by far justify the interest in the area. However, as larger and more complex systems are being developed, in general there is also an increase in the complexity of the problems to be solved, and it becomes more difficult to understand and evaluate the performance of the overall system. As a consequence, there is an increasing need for tools and techniques which can help in understanding the behavior of such systems. One such technique is the modeling of the system. A model is an abstraction of a system and attempts to capture in a simplified way the essential attributes of the system being studied. Modeling is in several cases a much

cheaper alternative than performing measures in the real system, and the only alternative in the design stage. Among the modeling techniques, queueing network modeling is a particular approach in which the system is represented as a network of queues which is evaluated by simulation or analytically. It has been largely used with success to model computer systems. In these cases, the network of queues is a collection of service centers which represents the system resources and customers or jobs which represent users or transactions.

The most widely used tool for computer system analysis is still discrete event simulation. The main reason for that is the capability of representing details of the system being studied. Recently simulation packages were implemented specifically to handle queueing network models [Saue81c, Saue84, Berr82]. These packages have been widely used with success. A more recent implementation incorporates queueing network analysis capabilities in a simulation package which uses a formal graph model to describe the behavior of the system [Vern83]. As computer time gets cheaper we should expect that simulation will continue to be a widely used tool. However, the analytical solution of a model is always more attractive whenever the assumptions used to make the model mathematically tractable do not oversimplify the final model to the point of jeopardizing its accuracy. This is true since an analytical solution is usually several orders of magnitude cheaper than simulation. For large systems, it may be the only alternative.

One of the most used class of analytical models is product form queueing networks. Unfortunately, the existing solution techniques for solving closed chain product form networks exactly are impractical for very large systems, since the computational requirements of these techniques grows combinatorically with

the number of closed routing chains and customers in the network, unless the network has a special sparseness structure [Lam83]. Therefore, it is important to develop cost effective and accurate techniques for the approximate solution of such large models. Furthermore, there is no exact solution technique available to handle queueing networks which violate product form requirements except for Markov chain approaches (which are impractical even for moderate size networks). It is therefore equally important to develop accurate and cost effective approximate methods to handle those cases as well.

The work presented in this dissertation was motivated by our interest in analyzing queueing network models of large local area networks, in particular the LOCUS local area distributed system [Pope81, Walk83b, Walk83a]. Locus is a distributed operating system developed and currently in use at UCLA on a network of VAX11/750 and 780 computers connected by an Ethernet channel. It provides a high degree of network transparency so that, for the user, the system appears as a single computer. Knowledge of the network and the interaction among the several sites is completely hidden above the operating system level. Files and processes can be transferred dynamically between machines in the network. Processes originating at one site may run in another site and process interaction is the same, independent of the location.

Most of the modeling of local area networks has concentrated on the performance of the communication network. In [Gold83] a model was proposed that focused more in the individual sites and their interaction in the network. This model was simply a composition of several "central server models with terminals [Lave83], which have been successfully used in the past to represent the behavior of a single computer system. These "central server models" were inter-

connected through a simplified representation of the communication channel, in order to maintain product form characteristics. Limited comparison with measurements in the LOCUS system for a four site network showed the validity of the model as a first approximation. The current LOCUS configuration running at UCLA is composed of sixteen VAX computers and there are plans to add many more sites. The large number of sites preclude the use of any exact solution technique to solve a model of the system as proposed above. Furthermore, the introduction of more details in the model are likely to violate product form requirements.

In this dissertation, we focus our study on the development of accurate and cost effective algorithms to approximately solve large queueing network models and in particular, the one proposed to model the LOCUS distributed system. We also investigate the problem of balancing the load in a distributed system environment and propose an exact algorithm which finds the optimum load balancing strategy. All the algorithms developed were implemented as interactive tools and currently run on our LOCUS environment.

In chapter 2 we overview some of the approximate methods proposed to solve product and non-product form queueing network models. We try to present the techniques in an unified view by identifying a common ground in which most of the methods are based, hopefully simplifying the understanding of several methods found in the literature.

In chapter 3 we propose an approximation technique to solve product form queueing network models with single server fixed rate, infinite server and multiple server service centers. This technique was initially developed having in mind the solution of large distributed systems models, in particular LOCUS type

of models. We have extended the approach to handle general product form queueing models as well. Extensive empirical studies demonstrate that the approach has better accuracy/cost characteristics when compared to other methods.

Chapter 4 is devoted to non-product form networks. We first address the problem of introducing a more detailed representation of the Ethernet channel in a LOCUS type of model. Furthermore we address the problem of simultaneous resource possession, i.e., networks in which customers contend for resources while holding others acquired in a different stage of execution. We developed an approximation technique to handle non-product form networks where each simultaneous held resource is dedicated to a single type of user or shared among different types of users.

In chapter 5 we investigate the load balancing problem in distributed systems. We again use a LOCUS type of model to represent the system. We develop an efficient and exact algorithm for the optimum solution of a very general class of load balancing problems supporting multiple class of jobs and site constraints. The algorithm is based on a downhill procedure to search for the global minima.

Chapter 6 extends the algorithm of chapter 5 in order to handle cases where jobs are composed of multiple tasks which can only be executed in a subset of all the sites in the network. We also propose an efficient approximation algorithm to solve the routing problem in a multiple closed chain network model, which reduces the computational complexity of the original exact algorithm proposed in [Koba83].

CHAPTER 2

OVERVIEW OF APPROXIMATE METHODS FOR SOLVING QUEUEING NETWORK MODELS.

2.1. Introduction.

The exact analysis of general closed queueing networks is typically too costly due to the large size of the state space, even if the usual Markovian assumptions are made. The most general method is to determine the Markov process representation of the network, which is feasible only if the network has a small number of service centers and jobs. A clever way to attack the problem is described in [Moll81]. However, if the network has product form solution [Jack63, Bask75] efficient algorithms were developed to exactly solve for these models [Reis76, Reis80, Reis81]. Unfortunately, these algorithms have computational requirements that grow combinatorially with the number of closed chains in the network. Therefore, for multiple chain networks with say five or more closed chains and a few customers per chain (say ten) the analysis of closed product form networks can be prohibitive, unless the network has a special sparseness structure, e.g. Lam and Lien [Lam83]. The high cost of existing algorithms for product form networks and the lack of efficient exact algorithms for non-product form networks are the main reasons why it is important to develop accurate and cost effective methods for the approximate analysis of closed queueing networks.

Most of the methods that have been proposed for the approximate analysis of closed queueing networks fall into one of the following three classes: (1) non-iterative methods based on heuristic extensions to the equations of Mean Value Analysis (MVA); (2) non-iterative methods based on hierarchical decomposition and/or Norton's theorem and (3) iterative methods. Non-iterative MVA based methods require a recursive solution over all network population vectors ranging from the all zero vector up to the population vector of interest and, therefore, have the same multiple chain computational limits as exact MVA. Non-iterative methods based on hierarchical decomposition/Norton's theorem typically involve solving a subnetwork for all possible population vectors in order to obtain a "flow equivalent" server representation and then solving an aggregated network consisting of the flow equivalent server and the rest of the network. These methods are more suited to single chain than to multiple chain networks due to the cost of solving the subnetwork and aggregated network.

The majority of the approximate methods described in the literature does not give error bounds for the results obtained. In general the accuracy of the method is evaluated empirically by comparing the results of the approximate algorithms, obtained after analyzing a large number of networks, with the results obtained by analyzing the same networks with an exact algorithm, if the network is product form, or by simulation or Markov analysis, otherwise.

In this chapter we mainly concentrate on iterative methods, since the solution technique we will propose in future chapters is also iterative. We present an unified view of many of the iterative methods proposed in the literature, which hopefully will simplify the understanding of the approaches.

Many of the approximate methods for solving closed queueing networks are based on the MVA equations due to the simple nature of these equations and the physical interpretation they provide. Below we present the equations for closed product form networks for reference.

We define the following notation that will be used throughout the next three chapters:

J	=	total number of service centers.
K	=	total number of chains.
N_k	=	population of chain k .
\bar{N}	=	population vector = (N_1, \dots, N_K) .
$\lambda_k(\bar{N})$	=	mean throughput of chain k with population \bar{N} .
θ_{kj}	=	visit ratio of a chain k customer to center j .
$L_j(\bar{N})$	=	mean number of customers at center j with population \bar{N} .
$L_{kj}(\bar{N})$	=	mean number of customers of chain k at center j with population \bar{N} .
$q_j(\bar{N})$	=	mean number of customers in the queue of center j (excluding customers in service) with population \bar{N} .
$q_{kj}(\bar{N})$	=	mean number of customers of chain k in the queue of center j (excluding customers in service), with population \bar{N} .
$W_{kj}(\bar{N})$	=	mean waiting time (queueing time + service time) of a chain k customer at center j with population \bar{N} .
x_{kj}	=	mean service time of a chain k customer at center j .
μ_{kj}	=	service rate of chain k customers at center j .
u_{kj}	=	relative utilization of a chain k customer at center j , = $\theta_{kj} x_{kj}$.
M_j	=	number of servers at center j .
$P_j(i \bar{N})$	=	probability that there are i customers at center j given that the population vector is \bar{N} .
$\lambda_k^{(-j)}(\bar{N})$	=	throughput of chain k in a network with center j removed when the population is \bar{N} .
$PB_j(\bar{N})$	=	probability that all servers of center j are busy, when the population is \bar{N} .
\vec{e}_k	=	k -dimensional vector whose k -th element is one and whose other elements are zero.

Let J denote the number of service centers, the first J_1 of which are single server fixed rate (SSFR), centers J_1+1 to J_2 are load dependent and the remainder are infinite server (IS) service centers. For $k = 1, \dots, K$.

$$\lambda_k(\bar{N}) = \frac{N_k}{\sum_{j=1}^J \theta_j W_j(\bar{N})} \quad (2.1)$$

$$L_j(\bar{N}) = \lambda_k(\bar{N}) \theta_j W_j(\bar{N}) \quad j = 1, \dots, J \quad (2.2)$$

$$W_j(\bar{N}) = \begin{cases} x_j [1 + \sum_{i=1}^K L_j(\bar{N} - \bar{e}_i)] & j = 1, \dots, J_1 \\ x_j(\bar{N}) \sum_{i=1}^N \frac{i}{\mu_j(i)} P_j(i-1 | \bar{N} - \bar{e}_i) & j = 1, \dots, J_2 \\ x_j & j = J_2 + 1, \dots, J \end{cases} \quad (2.3a)$$

$$W_j(\bar{N}) = x_j(\bar{N}) \sum_{i=1}^N \frac{i}{\mu_j(i)} P_j(i-1 | \bar{N} - \bar{e}_i) \quad j = 1, \dots, J_2 \quad (2.3b)$$

$$W_j(\bar{N}) = x_j \quad j = J_2 + 1, \dots, J \quad (2.3c)$$

$$P_j(i | \bar{N}) = \frac{1}{\mu_j(i)} \sum_k T_{kj} \lambda_k(\bar{N}) \theta_j P_j(i-1 | \bar{N} - \bar{e}_k) \quad (2.4a)$$

$$P_j(0 | \bar{N}) = \frac{\lambda_k(\bar{N})}{\lambda_k(\bar{N})} P_j(0 | \bar{N} - \bar{e}_k) \quad (2.4b)$$

Equation (2.3b) reduces to:

$$W_j(\bar{N}) = x_j \left[1 + \frac{q_j(\bar{N} - \bar{e}_i) + PB_j(\bar{N} - \bar{e}_i)}{M_j} \right] \quad j = J_1 + 1, \dots, J_2 \quad (2.3d)$$

for multiple server (MS) service centers, where $PB_j(\bar{N})$ is calculated from (2.4a)

which, after simplification, gives:

$$PB_j(\bar{N}) = \begin{cases} \frac{1}{M_j} \sum_{i=1}^K \lambda_i(\bar{N}) \theta_i [PB_j(\bar{N} - \bar{e}_i) + P_j(M_j - 1 | \bar{N} - \bar{e}_i)] & |\bar{N}| > M_j \\ 0 & |\bar{N}| \leq M_j \end{cases} \quad (2.4c)$$

where $|\bar{N}| = N_1 + \dots + N_K$.

These equations express an exact relationship between the performance parameters and can be used recursively for an exact solution of the network. For networks with SSFR and IS service centers only, the computational require-

ments to solve the above equations are $O(\prod_{i=1}^K (N_i + 1))$. To solve networks with multiple server service centers or general load dependent service centers, the computational requirements increase considerably due to equation (2.4b). In this case ~~2nd load dependent servers~~ networks have to be solved, each with some of the load dependent service centers removed from the network [Tucc82]. It is easy to see then that for networks with a few closed chains, this recursive computation can be prohibitively expensive. Our experience indicates that networks with more than five chains and ten customers per chain can be impractical to solve in a medium size computer, (say, VAX11/750).

2.2. Non-Iterative Methods.

Most of the non-iterative methods proposed in the literature are based on heuristic extensions to the MVA equations and hierarchical decomposition and/or Norton's theorem.

Based on heuristic extensions to the MVA equations.

It is known that for first come first serve (FCFS) service centers, the product form solution holds only if all jobs belonging to different chains have the same exponential distribution. Reiser [Reis79] proposed a heuristic modification to the MVA equation for the mean waiting times (i.e., equation (2.3a)) for FCFS service centers with different service demand distributions for different chains.

Observing equation (2.3a), we see that the waiting time observed by a job of a chain k at center i has three components, namely: (1) the mean service time of the arriving job at service center i , s_{ki} ; (2) the mean backlog of jobs waiting in queue; (3) the residual service time of the job in service on arrival of a chain

k job. Reiser assumed that: (1) the arrival theorem still holds for non-product form networks with FCFS service center with general service demand distributions; (2) the flows in all chains are sufficiently randomized.

Using the above assumptions and standard renewal theory to calculate (3) above, the following equation was obtained

$$W_n(\bar{N}) = z_n + \sum_{i=1}^K z_n q_{in}(\bar{N}-e_i) + PB_n(\bar{N}-e_i) \times \sum_{i=1}^K \frac{\lambda_i(\bar{N}-e_i) e_n}{\sum_{i=1}^K \lambda_i(\bar{N}-e_i) e_n} \frac{\bar{x}_n^2}{2z_n} \quad (2.5a)$$

which simplifies to

$$W_n(\bar{N}) = z_n + \sum_{i=1}^K z_n L_{in}(\bar{N}-e_i) \quad (2.5b)$$

for exponential service time distribution.

Reiser did limited comparison with simulation, with good results. Equation (2.5a) is applicable to single server service centers only. In chapter 5 we extend this result to handle multiple server service centers as well.

Another approximation that falls into this first category is the recent proposed approximation for pre-empt resume priority queues with exponential service demands [Brya83, Chan82a], (in [Brya83] it is also proposed and approximation for pre-emptive head of line discipline, but no error analysis was done). Like in Reiser's approximation for FCFS queues it was assumed that the arrival theorem holds for non-product form networks with priority queues. Again, under the light of equation (2.3a), the expected time of a tagged job in a priority queue is given by three components: (1) the mean service time of the arriving job at the priority center; (2) the mean backlog of jobs in the priority queue with equal or higher priority than the tagged job, (3) the work brought by the higher

priority jobs which arrive while the tagged job is in queue. In [Brya83] extensive studies were done to evaluate the accuracy of the approximation.

Based on Hierarchical Decomposition and/or Norton's Theorem.

In these methods the network is decomposed into subnetworks which are solved independently. The results are then aggregated to obtain the solution of the original network [Cour75, Cour77].

Most of the decomposition methods are based on the Chandy-Herzog-Woo theorem [Chan75] which is also called Norton's theorem. This theorem states that a subnetwork of queues can be replaced by a single center which is the "flow equivalent" of this subnetwork. This replacement is exact for product form networks and is used heuristically for non-product form networks. In [Lave83] and [Saue81a] we find a brief discussion on the justifications for this heuristic decomposition and aggregation. One of the justifications is based on weakly coupled subnetworks [Cour75] which does not depend on the product form conditions.

This approach has been successfully used in modeling some queueing networks with simultaneous resource possession [Saue81b]. A typical example is shown in Figure 2.1 where a memory partition has to be allocated to a job before being allowed to compete for the CPU and DISKS. In this Figure the allocation of memory is represented by an "allocate node" [Lave83, Saue84]. Figure 2.2 shows how Norton's theorem can be used to approximately solve the model of Figure 2.1. In Figure 2.2a, the subsystem with memory constraint is "isolated" and solved by "shortening" its paths to the remainder of the network (Norton's theorem). This subnetwork is solved for all possible customer popula-

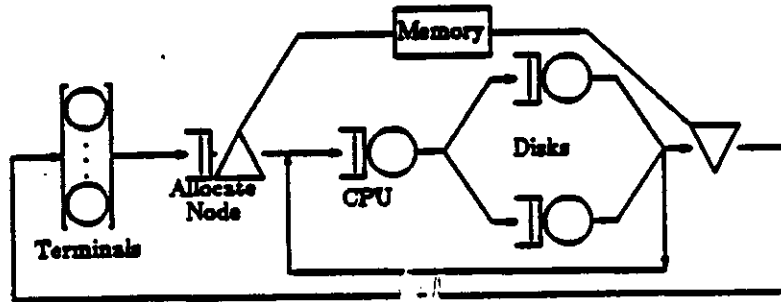


Figure 2.1. A central server model with memory constraint.

tions, in this case, the number of memory partitions. The throughputs $\lambda(n)$ for all n obtained are then used as the service rates ($\mu(n) = \lambda(n)$) for the flow equivalent server in Figure 2.2b. The solution of this last model gives the final

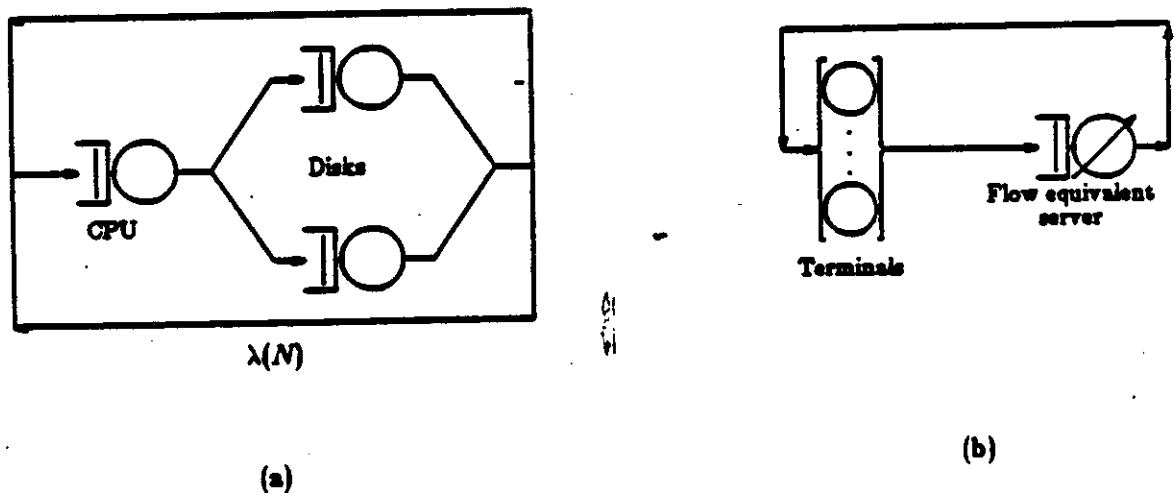


Figure 2.2. Application of Norton's theorem to solve the network of Figure 2.1.

result.

Sauer [Sauer81b] presents a study of the accuracy of this decomposition approach. Unfortunately the method presents difficulties for multiple chain networks with shared memory constraints [Sauer81b, Lave83]. There are other approaches based on Norton's theorem. The iterative ones will be mentioned in

the next section.

Other Non-Iterative Methods.

Recently, a new approach has been proposed [McKe81, McKe82, Rama82, McKe83], to handle large closed queueing networks. The general idea behind the method is to substitute Euler's integral for a factorial term in the equation for the normalization constant [Lave83] for a network. Then Laplace's method is applied to make asymptotic expansions of the integrals obtained.

The major features of this approach are: (1) the algorithm is cheap, $O(JK^t)$ where t , the number of terms in the asymptotic expansion of the first and second moments of the quantities of interest (utilizations, etc), is experimentally chosen to be four for accurate results; (2) bounds are obtained for the results; (3) second moments can be obtained for queue lengths.

However, the method presents some drawbacks: (1) as reported in [Rama82], slow convergence occurs for "small" problems; (2) up to the date this dissertation was written, the method is restricted to networks in which all chains visit an IS center and (3) all other service centers have utilizations that are not too large, e.g., less than .85. These limitations may be severe for a large number of applications. Furthermore, no extensions were published to handle networks with load dependent service centers.

2.3. Iterative Methods - a Unified View.

Iterative methods in general have the property that a set of non-linear equations is developed that relates, in an approximate way, unknown parameters of the network under investigation and iteration is used to solve the equations.

Often there are two sets of equations and two corresponding set of parameters, say X and Y . The two sets of equations take the form:

$$Y = F(X) \text{ and } X = G(Y)$$

These equations can be reduced to a single set of fixed point equations, e.g.,

$$X = G(F(X))$$

In general it is difficult to establish existence and uniqueness of solutions of simultaneous non-linear equations and to establish that a particular iterative solution method converges. In [DeSo83] we find a brief discussion concerning this issue.

A variety of iterative methods have been described in the literature. Many of these methods can be viewed in the following unified way. The methods involve analyzing an interrelated collection of subnetworks. Each subnetwork is analyzed with the remainder of the network (the subnetwork's complement) represented in a simplified way. The methods are iterative in that the parameters of the simplified complement of a subnetwork are modified after the other subnetworks are analyzed. These modifications reflect the latest solution of the subnetworks in the complement. The subnetwork is then reanalyzed with the modified parameter values for its complement. This is continued until some convergence criterion is satisfied. As an illustration, consider Figure 2.3. In this Figure a queueing model is broken into three subnetworks (Figure 2.3a). An approximate solution to the network may be obtained by analyzing "independently" the three subnetworks, with a simplified representation for its complement, as shown in Figure 2.3b. For instance, subnetwork i is analyzed after the parameters of its complement are updated with the previous solutions of subnetworks $(i - 1)_{\text{modulo } 3}$ and $(i - 2)_{\text{modulo } 3}$.

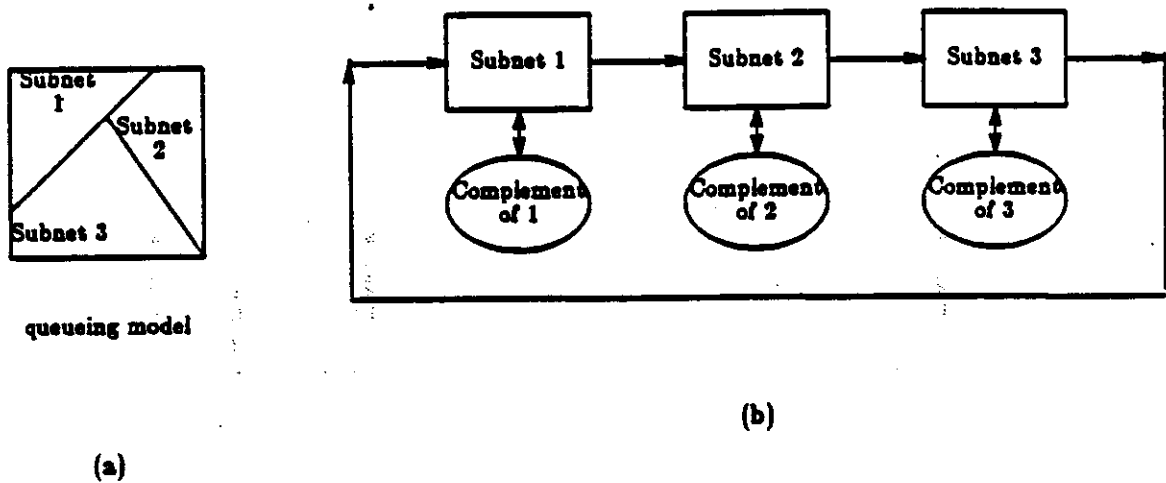


Figure 2.3. Example of an iterative procedure.

The main reasons to partition a queueing network into subnetworks are: (1) for product form networks, to obtain subnetworks which are much less costly to analyze than the original networks. Hopefully, the sum of costs of solving each subnetwork times the number of iterations is significantly less than solving the whole network. (2) for non-product form networks, to obtain subnetworks with product form characteristics, or subnetworks that are otherwise simple enough to be analyzed. In addition to the simplified nature of the subnetworks, it is desirable that the parameters of the complement be efficiently calculatable from the analysis of the other subnetworks.

In this section we present an overview of iterative methods for product and non-product form networks emphasizing the above unified view where appropriate. We present some examples from the literature and discuss some general issues such as accuracy and cost. Finally we discuss issues related to the partition of a queueing network and the possible representations of the complement of a subnetwork.

2.3.1. Some Existing Iterative Methods.

Bard's Approximation.

Bard [Bard81] presented an approach that is closely related to MVA and includes several examples from the literature as special cases. The approach is based on the first two MVA equations ((2.1) and (2.2)) and the following equation which replaces (2.3):

$$W_{ij}(\bar{N}) = F_{ij}(\bar{\lambda}_j, \bar{L}_j) \quad (2.6)$$

where $\bar{\lambda}_j$ is the vector of throughputs for each chain at center j and \bar{L}_j is the vector of mean queue sizes for each chain at center j . Equation (2.6) gives an approximate functional relationship among the performance parameters that have to be specified based on the nature of the j -th service center. Bard considered several non-product features that appear in performance modeling. He used several of his approximations in a model of VM/370 and found errors within 20% for mean response times when compared to system measurements. However, the accuracy of the above approximation has not been adequately assessed.

One way to view Bard's approach is that (2.6) represents the effect of the rest of the network on the mean waiting time of service center j . The effect is a function of the throughputs (arrival rates) and the mean queue sizes (population). Equations (2.1) and (2.2) give constraints that these parameters must satisfy.

In the above, the approximation for a service center, or more generally a subnetwork, depends only on the arrival rates and mean populations at that center, rather than on any more detailed representation of the effect of the

remainder of the network. Presumably, more detailed representations could lead to more accurate approximations.

Schweitzer's Approximation.

A widely used approximation for product form networks with SSFR and IS service centers, which falls into the category of the above approximation, proposed by Schweitzer is:

$$L_{ij}(\bar{N}-\bar{e}_k) = \begin{cases} L_{ij}(\bar{N}) & l \neq k \\ \frac{(N_k-1)}{N_k} L_{ij}(\bar{N}) & l = k \end{cases} \quad (2.7)$$

This approximation removes the recursion in equation (2.3a) and greatly reduces the computational complexity necessary to exactly solve a model. As we will see in chapter 3 the errors obtained are typically less than 20% for mean queue lengths and waiting times and less than 10% for throughputs. However, large errors may be found in some cases, which motivated the development of the next algorithm we describe. A proof that the non-linear equations that result from using Schweitzer's approximation in (2.3a) have a solution in the feasible region can be found in [DeSo83].

Linearizer.

Chandy and Neuse [Chan82b] proposed an algorithm called Linearizer, which is based on refinements to Schweitzer's approximation and was also proposed to approximately solve networks with SSFR and IS service centers only. This approximation is very accurate but more costly. In this method it is assumed that the error made with Schweitzer's approximation is known. It starts with the assumption that the error is zero and then successive iterations

refine the value of this error. The estimate value of the error is obtained by approximately solving K networks identical to the original one, but each network with one less job at chain k . Schweitzer's approximation is used for solving each network.

Reducing the equations obtained by this method to a set of simultaneous non-linear equations [Lave83], $I + K(I - 1)$ sets of such equations must be solved, where I was heuristically found to be equal to 3 in [Chan82b]. Therefore, this algorithm is considerably more expensive than Schweitzer's approximation where only one set of simultaneous non-linear equations must be solved. On the other hand, extensive tests reported in [Chan82b] and in chapter 3 of this dissertation have shown that, in the average, Linearizer is one order of magnitude more accurate than Schweitzer's approximation. As we will see in the next chapter, the use of the Linearizer algorithm can be prohibitive for networks with a large number of chains.

Neuse and Chandy [Neus81] proposed an extension to the Linearizer algorithm to approximately solve product form queueing network models with load dependent service centers. The approach uses additional heuristics for load dependent centers. They chose to estimate the probability density of the queue lengths as:

$$P_j([L_j(\bar{N}-\bar{c}_i)]|\bar{N}-\bar{c}_i) = [L_j(\bar{N}-\bar{c}_i)] + 1 - L_j(\bar{N}-\bar{c}_i) \quad (2.8a)$$

$$P_j([L_j(\bar{N}-\bar{c}_i)]+1|\bar{N}-\bar{c}_i) = 1 - P_j([L_j(\bar{N}-\bar{c}_i)]|\bar{N}-\bar{c}_i) \quad (2.8b)$$

$$P_j(i|\bar{N}-\bar{c}_i) = 0 \quad \text{for } i < [L_j(\bar{N}-\bar{c}_i)] \text{ or } i > [L_j(\bar{N}-\bar{c}_i)] + 1 \quad (2.8c)$$

(where $[L]$ is the greatest integer less or equal to L).

The above equations indicate that the probability density of queue lengths at a load dependent service center for population $(\bar{N} - \bar{c}_i)$ will be different than zero at most in two points around the average queue length. However, there is no theoretical support for this heuristic choice.

Marie's Method.

Marie's method [Mari79b] is an iterative algorithm for single chain networks of FCFS service centers with general service time distributions. It has been extended for multiple chains by Neuse and Chandy [Neus82]. Each step of the iteration involves J substeps where J is the number of service centers violating product form requirements.

The method can be easily explained in terms of our "unified view". Each iteration involves analyzing a set of subnetworks and its complement. In this method, a subnetwork is simply formed by one of the FCFS queues in the original network which violates product form requirements (say queue j). Marie assumed that Norton's theorem was valid for non-product form networks. Then, the complement of center j 's subnetwork was simply formed by the flow equivalent of the remaining of the network, obtained after (1) replacing each of the remaining FCFS centers with general distributions, by an "equivalent" center with exponential service times and state dependent service rates (thus, the remaining of the network has product form) and (2) applying Norton's theorem to the remaining of the network. The model formed by a subnetwork and its simplified complement is not product form, but it has the solution of an M/G/1 queueing model with state dependent arrival rates (hence the solution is costly, particularly for multiple chain networks). After solving for the subnetwork containing center j , the parameters of the "equivalent" exponential center j

are determined and used in subsequent substeps.

As pointed out in [Lave83] we have a number of ways to determine the performance measures for a center, because the results for each substep are likely to be inconsistent. In [Mari79b] the performance measures for a FCFS center are obtained when the center is in the subnetwork.

Jacobson and Lazowska's Methods for Simultaneous Resource Possession.

There are several modeling cases where we must impose restrictions on the number of jobs in some subnetworks, because these jobs can only wait for some resources if they have already obtained other ones. This is the case, for instance, when we want to represent memory contention in a central server model, as previously illustrated in Figure 2.1. We have already mentioned one method to solve the example of Figure 2.1 which uses Norton's theorem.

Recently, Jacobson and Lazowska proposed two different methods to deal with simultaneous resource possession: [Jaco82, Jaco83]. Although the methods can be used to solve more sophisticated queueing models than the one presented in Figure 2.1, we will use this networks for illustration purposes.

The first of the methods proposed by Jacobson and Lazowska is called the "Method of Surrogate Delays" [Jaco82]. It is applicable to models which can be framed in terms of a set of primary resources and a secondary subsystem. The approach, applied to the network of Figure 2.1, requires that two interrelated product form subnetworks, shown in Figures 2.4a and 2.4b, be solved. The subnetwork in Figure 2.4a explicitly represents the terminals and memory partitions. The server representing the memory partitions is a multiple server

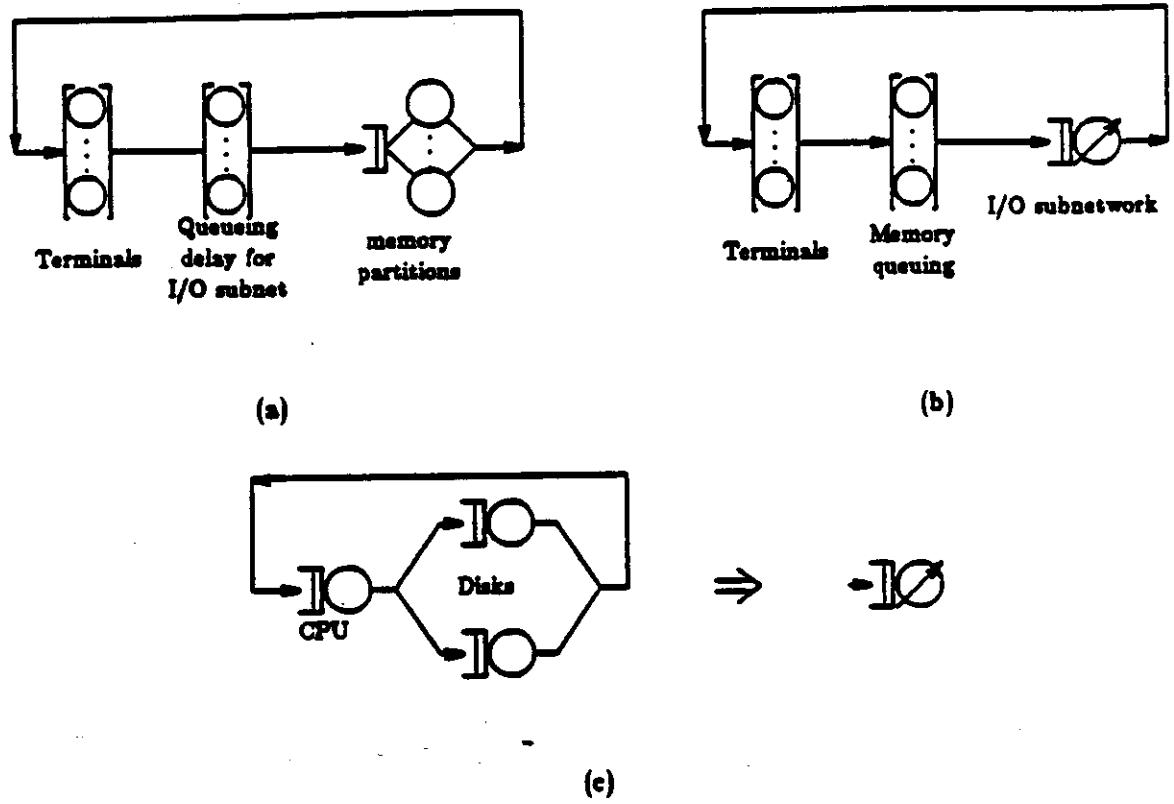


Figure 2.4. The method of surrogate delays.

(MS) service center with M servers, where M is the number of memory partitions, and service time equal to the response time a job would experience in the I/O subsystem if no contention for CPU and disks occurs. The complement is formed by a pure delay or infinite server, which represents the total queueing delay (excluding service) in the I/O subsystem (call this delay X). The subnetwork in Figure 2.4b explicitly represents the terminals and I/O subsystem. However, the I/O subsystem is represented by a "flow equivalent" center. The service rates for this center are obtained after applying Norton's theorem to the I/O subsystem, and limiting the maximum service rate to the rate obtained when M jobs are present. The complement of the subnetwork of Figure 2.4b is

formed by an infinite server center which represents the total queueing delay (excluding service) due to memory contention (call this delay Y). Finally, delay X is obtained after solving the subnetwork of Figure 2.4b and delay Y is obtained after solving the subnetwork of Figure 2.4a.

The surrogate delay method is not readily extendible to multiple chain networks since it involves introducing a FCFS center with queue size dependent rates [Jaco82].

The second (and newer) approximation for dealing with simultaneous resource possession was presented in [Jaco83]. In this method simultaneous resource possession is represented as a population constraint in a subnetwork. The approach, applied to the network of Figure 2.1, requires that two interre-

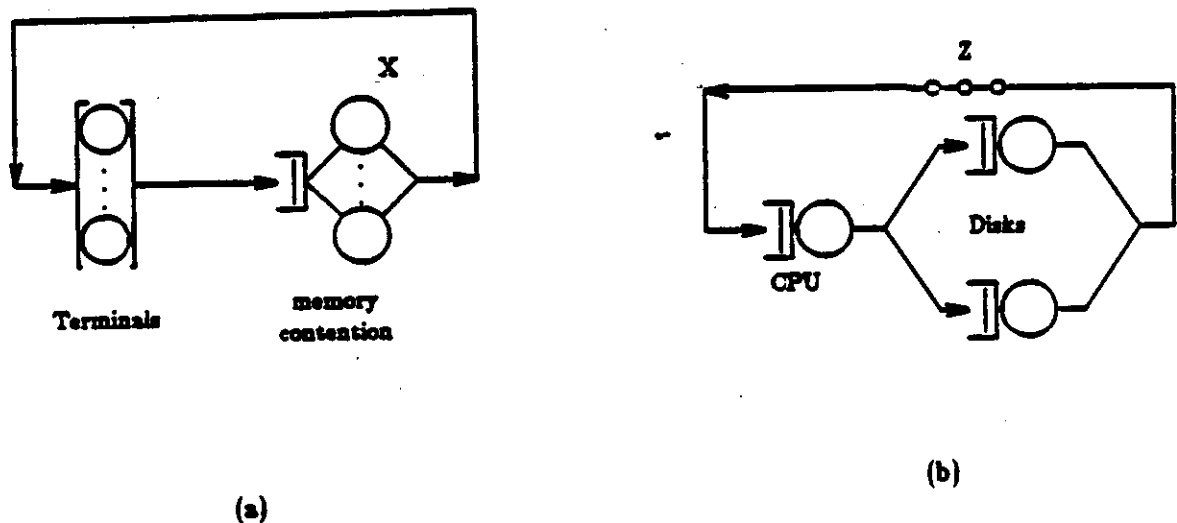


Figure 2.5. "Population constraint" method.

lated product form subnetworks; shown in Figure 2.5a and 2.5b, be solved. The subnetwork in Figure 2.5a is formed by the terminals and memory partitions which are represented by a MS service center with M servers. The complement

is represented by the average delay a job experiences in the I/O subsystem. This delay is used as the service time for the MS center (call this delay X). The subnetwork in Figure 2.5b explicitly represents the I/O subsystem. The complement is represented by the population size which is chosen equal to the mean number of used memory partitions (call this number Z). The mean number of used memory partitions Z is obtained as a function of the mean delay in the I/O subsystem by solving the network of Figure 2.5a. The delay, or service time X , is obtained as a function of Z by solving the network of Figure 2.5b. Since Z needs not to be an integer, an approximation is needed to solve the subnetwork in Figure 2.5b for population Z . In general, Schweitzer or Linearizer approximations can be used to approximately solve product form networks with non-integral population.

This second approximation method is intended for multiple chain networks. However, in this case, an additional approximation is required to handle FCFS centers with multiple servers and chain dependent mean service times, even if the original network does not have such centers. For example, if the network of Figure 2.1 has more than one chain with different population or different service requirements in the I/O subsystem, then the mean memory service times in Figure 2.5a will be chain dependent. For single server FCFS centers there is Reiser's approximation presented in section 2.2 (see also [Lave83]), but, to our knowledge, there is no such approximation in the literature for FCFS with multiple servers. In chapter 4 we present a new approximation to handle these cases.

Modeling Asynchronous Tasks.

Heidelberger and Trivedi [Heid82] proposed an iterative approach to solve queueing network models with "split nodes" [Lave83], i.e., service centers where a task is split into another independent one. In this method, a subnetwork is formed by the original network with the split nodes replaced by an equivalent service center without the "splitting" facility. The complement is formed by a Poisson source representing the effect of the secondary task. The arrival rate of this Poisson source is equal to the throughput of the chain which split the job at the split node.

2.3.2. Discussion.

The following discussion is based partly on [Zaho83] and partly on our own research. We have presented a number of methods that involve analyzing a collection of subnetworks. Informally a subnetwork is a detailed representation of a set of resources. The complement is typically represented by an approximation of the effect of the remainder of the network. Each subnetwork is analyzed in isolation with a simplified representation of the effect of its complement. In general, the parameters of a complement are function of the performance parameters of the other subnetworks and are determined by iteration (in [Heid82] however, the parameters of the complement is a function of the only existing subnetwork). As pointed out in the beginning of this section, the main reasons to partition a queueing network into subnetworks are fundamentally related to the cost of solving exactly the original network. Therefore, it is desirable that the subnetworks obtained be simple and the parameters of the complement be efficiently calculatable from the analysis of the other subnetworks.

Throughout this chapter we have seen several ways to represent the effect of the complement of a subnetwork. For example, Marie [Mari79b] used queue size dependent arrival rates; Jacobson and Lazowska used IS centers [Jaco82] and reduced customer population [Jaco83]; Heidelberger and Trivedi [Heid82] used Poisson arrivals. The accuracy of these methods must be evaluated empirically since so far error bounds have not been obtained.

For multiple chain networks, the effect of a complement can be represented differently for different chains. For example, with regard to one chain the effect of the complement can be represented as Poisson arrivals, i.e., the chain is open; with regard to another chain, the complement can be represented as an infinite server and with regard to another chain as a reduced population. This general framework permits numerous possibilities and little is known about how to utilize these possibilities. Figure 2.6 illustrates a subnetwork and its complement. The following are some of the choices that are available:

- 1) The choice of subnetworks. Each subnetwork may contain a disjoint set of resources or they may overlap. (see for example, Marie's method [Lave83], Jacobson and Lazowska [Jaco80], and chapter 3 of this dissertation.

- 2) The choice of the way in which the final estimates of the performance parameters of the original network are obtained. This choice arises because the same performance parameters, e.g., a chain throughput or a mean queue size, may be represented in more than one subnetwork. Thus, when the iteration is terminated, there may be more than one value for the same parameter. The final estimates could be a function of these values.

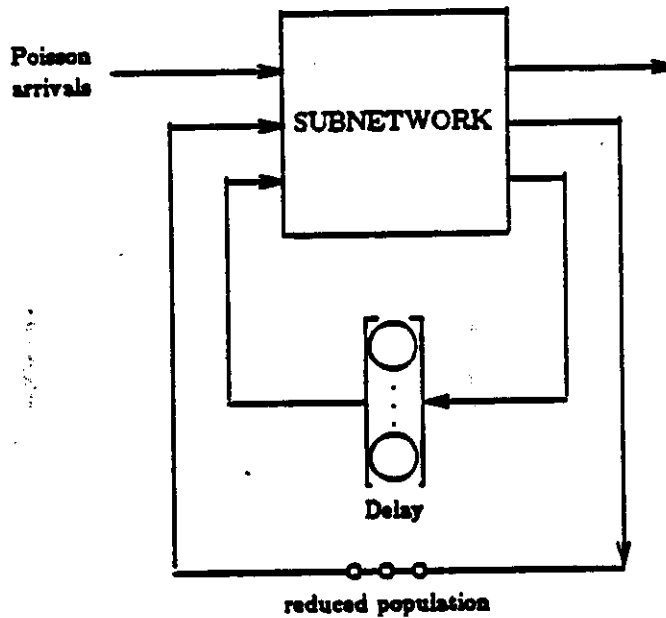


Figure 2.6. A subnetwork and possible representations of its complement.

3) The choice of the complement of a subnetwork for each chain. Only some of the possibilities have been mentioned above.

4) For a given chain and representation of the complement for this chain, the choice of how the parameters of the complement are determined. For example, for a Poisson arrivals representation the arrival rate must be determined. Some of the choices are: (1) set the arrival rate equal to the throughput for that chain that was computed on the previous iteration; (2) set the arrival rate equal to the value that yields the same mean population in the subnetwork for that chain as was calculated on the previous iteration; (3) set the arrival rate for a chain in a center according to some characteristics of that center, and/or to yield the same mean population of that chain in that center as calculated on the previous iteration. Note that the second method above requires an additional

iterative step while the first does not.

Zahorjan [Zaho83] considered a few representations for the complement of a subnetwork. In his paper subnetworks are disjoint and the representation is the same for each chain. The representations he considered were Poisson arrivals, infinite server, first come first serve single server and reduced population. He considered two alternatives with regard to how the parameters of a complement are determined: matching the throughput and matching the mean population in the subnetwork (choices (1) and (2) mentioned above in item 4). For example, when matching the mean population is used with open representation, the arrival rate of the complement of a chain in a subnetwork is chosen so that the resulting mean number of customers from that chain in the subnetwork is the same as calculated in the subnetwork where that chain is explicitly represented. As Zahorjan pointed out, the open representation with population match requires an iterative step when solving a subnetwork, in order to determine the right throughput that matches the population. The reduced population, on the other hand, requires iteration when throughput matching is used and FCFS and IS representations requires iteration if any of the above two matches is used.

If only one match is performed in a subnetwork, and if the subnetworks are very cheap to solve, the extra iteration mentioned above may not increase the overall cost significantly, but we will be limited to the representation of only one chain per submodel. However, if more than one representation is used, the submodel iteration is not trivial and the cost may increase considerably.

Zahorjan conclusions based on analytical results for single chain networks and on empirical results for small multiple chain networks are that the IS and FCFS single server representations with parameters determined from either throughput matching or population matching were the most accurate. These methods however require additional iterative steps. He found the Poisson arrival representations with throughput matching the least accurate with extremely large errors possible.

We argue that for solving large networks we should search for simple parametrizations and avoid extra iterative steps in solving for a subnetwork, since for large networks the subnetworks may be large or there may be a large number of these subnetworks.

2.4. Conclusions.

We presented an overview of some approximate methods for solving product and non-product form queueing network models. With the exception of the method of the integral representation and asymptotic expansions, which has some limited applicability for product form networks, all the other methods do not give any error bounds, and so their accuracy must be empirically evaluated.

For product form networks, Schweitzer and Linearizer approximations have been largely used and are by now standard approximations for networks consisting of only SSFR and IS service centers. Schweitzer's approximation is very cheap, but may present large error in some cases. Linearizer approximation on the other hand is more accurate, but much more expensive as well. There are no generally accepted MVA based approximations available for product form networks having queue size dependent rates, although one such approximation

was proposed in [Neus81]; as discussed in section 2.3.

For non-product form networks we overviewed several methods proposed to solve particular cases that violate product form requirements, some of them with some overlap in the set of problems they can solve. For example, both Marie's method and Feiser's approximation are proposed to solve networks with FCFS centers with general service times. The second method proposed by Jacobson and Lazowska for queueing networks with simultaneous resource possession seems to be able to solve all cases solved by the method of surrogate delays.

We have presented a unified view of several proposed iterative methods, with the purpose of highlighting some common ground most of these methods are based on. It is interesting to observe that, in the majority of these methods, subnetworks with product form characteristics are obtained, which simplifies their individual analysis. Some of them, however, still obtain non-product form subnetworks like Marie's method.

Finally, in Figure 2.7 we summarize in a diagram the methods overviewed in this chapter, indicating their applicability.

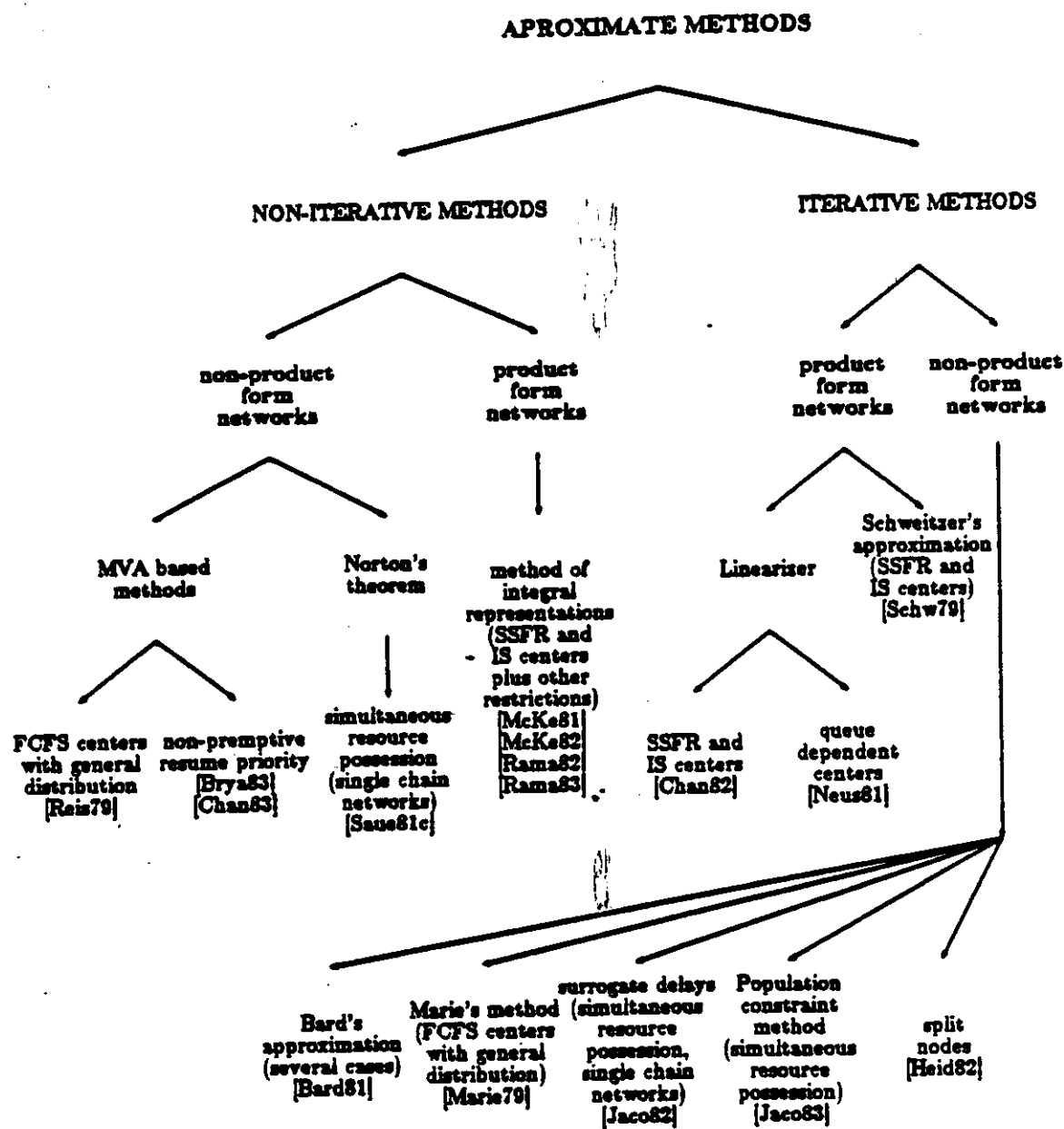


Figure 2.7. Some approximate methods for the analysis of queuing network models.

CHAPTER 3

A CLUSTERING APPROXIMATION TECHNIQUE FOR PRODUCT FORM QUEUEING NETWORK MODELS WITH A LARGE NUMBER OF CHAINS.

3.1. Introduction.

There are many applications which require the analysis of closed queueing networks models with a large number of chains. For example, in modeling a distributed computer system different chains might be used to model workloads generated at different sites [Gold83] and in modeling a packet switching networks different chains might be used to model traffic in different virtual channels [Reis79]. However, the exact analysis of such closed queueing networks is usually too costly.

The two major algorithms that have been widely used to solve exactly product form queueing network models are the convolution algorithm [Lave83, Reis76] and the mean value analysis algorithm (MVA) [Reis80, Reis81]. If only mean values are computed and there are no load dependent service centers present in the network, both algorithms have approximately the same cost requirements. Nevertheless, MVA has two significant advantages over the convolution algorithm: it presents no numerical problems; the equations obtained are extremely simple and have a physical interpretation. Unfortunately, the cost requirements of these algorithms are extremely high for large number of chains.

MVA for example requires $O(\prod_{i=1}^K (N_i + 1))$ operations and the memory require-

ments are in the same order of magnitude if no load dependent centers are present in the network. As pointed out in chapter 2 this high cost imposes a severe limitation in the size of the networks that can be solved. As an example, in a VAX11/750 computer, networks with more than a few customers per chain (say ten) and a few chains (say five) may be too costly to solve. When load dependent service centers are present and the MVA algorithm is used, the marginal queue length distributions appear in the equations for these centers. Although the simple interpretation is still present, the computational complexity grows combinatorially with the number of load dependent service centers. The very high cost of the algorithms to solve exactly large product form queueing networks and the growing need to solve larger and complex models are the motivation toward the development of accurate approximate techniques for these queueing models.

The work presented in this chapter was motivated by our interest in analyzing queueing network models of large local area networks, i.e., networks with dozens or even hundred of sites. As an example we mention the model of the LOCUS local area distributed system presented in [Gold83]. In that model each site was represented by a "central server model" [Lave83] with terminals linked by a representation of the communication channel. The behavior of customers from different sites was represented by associating one or more closed chains with each site. A large number of chains can result from this model. The LOCUS network at UCLA has currently 16 sites and more sites are planned to be added, which means that a model of this network would require at least 16 closed chains. Therefore, an approximate solution is required, even if the resulting model is product form. The problem gets considerably more complex if we introduce multiple server service centers in the model (for instance, to model a

multiprocessor computer), by the reason we have already outlined above.

We mention three approximate techniques that have been proposed to handle large closed queueing networks, which are described in chapter 2: the method of integral representations [McKe82, Rama82], Schweitzer's approximation [Schw79] and Linearizer [Chan82]. The method of integral representations, unlike the other two methods, obtains lower and upper bounds on performance measures. However, its applicability is restricted to networks in which all chains visit an IS service center, all other service centers have utilizations that are not too high, e.g., less than .85, and there are no load dependent service centers. In general, a CPU center in a computer works with high utilization, thus the restriction on utilization values imposes a severe limitation concerning the use of this method to solve the models we are interested in. Schweitzer's approximation is very cheap, but as we will show in the next sections large errors can be found (see also [Chan82]). Linearizer, on the other hand, is a very accurate approximation, but much more expensive than Schweitzer's approximation. We will present an example which shows that this approximation may be too expensive to solve a large local area network. Finally, there is no widely accepted approximation technique to handle multiple server service centers.

In this chapter we present a new approximate technique for closed product form queueing networks with SSFR and MS service centers. This technique extends previous methods and may prove to have better accuracy/cost characteristics.

3.2. A Simple Homogeneous Network.

We consider a single LOCUS type distributed local network. In this network, the behavior of customers from one site is represented by associating a closed chain to that site. This closed chain will be referred as the local chain of that site. All the other chains are referred as foreign chains to that site. This notation is adequate to a LOCUS type of model since customers logged on to a particular site (i.e., the local customers of that site) can request service at other sites (i.e., a foreign site). When these customers are executing in a foreign site they become foreign customers to the current foreign executing site.

Let S denote a particular site. $LC(S)$ is the local chain for S and $FC(S)$ the set of foreign chains for S .

Definition 3.1

We define a homogeneous network as the one in which:

- a. All sites are identical and the behavior of all customers executing at their respective local sites is the same, as well the behavior of customers in any foreign site. We use the notation z_h, θ_h for the mean service time and mean visit ratios respectively, of a customer of chain $l \in LC(S)$ at center $i \in S$. z_h and θ_h are identical for all sites S in the network. Similarly, we use z_f, θ_f for chain $f \in FC(S), i \in S$ or $f \in LC(S), i \notin S$. z_f and θ_f are identical for all sites S .
- b. All sites have the same number of customers, say N .
- c. All visit ratios of a customer from a particular site are relative to a designated service center in its local site.

- d. Customers from one site may request service from any foreign site with equal probability. We use the notation $\theta_j = \Theta_j / (M - 1)$ where M is the number of sites in the network. Note that $\Theta_j = \sum_i \theta_{ij}$, $i \in LC(S)$, $j \notin S$ and the sum is over all correspondent centers in sites different than S .

The queueing network resulting from the definition above possesses a strong symmetry. In fact, each chain is an exact replica of each other chain. Therefore, the following statements are true:

1. For any population vector, if two or more chains have the same number of customers, the mean queue lengths of corresponding service centers at the sites local to these customers are identical:

$$L_i(N_1, \dots, N_t, \dots, N_v, \dots, N_M) = L_j(N_1, \dots, N_t, \dots, N_v, \dots, N_M)$$

$$i \in T, j \in V, t \in LC(T), v \in LC(V), \text{ for } N_t = N_v$$

2. For any population vector, if we permute the number of customers of two chains, the mean queue lengths of corresponding service centers at the sites which are local to these two chains are also permuted. The mean queue lengths of the service centers in the remaining sites remain the same:

$$L_i(N_1, \dots, N_t, \dots, N_v, \dots, N_M) = L_j(N_1, \dots, N_v, \dots, N_t, \dots, N_M)$$

$$i \in T, j \in V, t \in LC(T), v \in LC(V).$$

These two statements follow from the fact that customers from chain t behave exactly as customers from chain v (by a, c and d in definition 3.1). Therefore, if the number of customers in these two chains are the same, the chains are indistinguishable from each other, which implies statement 1. Statement 2 is true since, because of the symmetry, all sites different from sites T and

V will not be affected by the permutation of the number of customers of chains t and v . Again, because of the symmetry, site T will "see" the remaining of the network that site V was "seeing" before the permutation and vice-versa. In other words, site T will behave as site V before the permutation and vice-versa. We can take advantage of the symmetry to reduce the computational requirements necessary to calculate the results of a network defined in 3.1, using MVA. In fact, we can show that these computational requirements are $O\left(\binom{N+M}{N}\right)$. To show this we first note that, by statements 1 and 2 above, once the results of a homogeneous network with population $\bar{N} = (N_1, \dots, N_M)$ are calculated, all the results for the same network with population \bar{N}' , obtained by permuting the elements of vector \bar{N} , are immediately known. Therefore, only the results for populations that can not be obtained by permuting the elements of \bar{N} of a previous calculated permutation vector needed to be obtained. This implies that the computational requirements are $O(f_M(N))$ where:

$$f_M(N) = \sum_{c_1=1}^N \sum_{c_2=1}^{c_1} \cdots \sum_{c_{M-1}=1}^{c_{M-2}} c_{M-1} \quad M \geq 1$$

$$f_0(N) = 1 \quad f_0(0) = 0$$

The above summation can be easily evaluated by the recursive relation,

$$f_M(n) = \sum_{i=1}^n f_{M-1}(i)$$

Now, taking the z-transform of $f_M(n)$

$$F_M(z) = \sum_{n=0}^{\infty} f_M(n)z^n = \sum_{n=0}^{\infty} \sum_{i=1}^n f_{M-1}(i)z^n = \frac{F_{M-1}(z)}{1-z}$$

which by induction gives

$$F_M(z) = \frac{F_0(z)}{(1-z)^M}$$

where

$$F_0(z) = \sum_{n=0}^{\infty} f_0(n)z^n = \frac{1}{1-z}$$

Therefore,

$$F_M(z) = \frac{1}{(1-z)^{M+1}}$$

and so

$$f_M(z) = \binom{M+N}{N}$$

As we would expect, the above result indicates that considerable savings in computation can be obtained for networks with a large number of sites and a few jobs per sites. For instance, if $M = 50$ and $N = 5$ the original MVA algorithm would have computational requirements on the order of 10^{55} but, by taking advantage of the homogeneity of the network, the computational requirements are in the order of 10^6 . However, it is still not feasible to find the exact solution for very large problems.

Throughout the remaining part of this section we will develop an approximation technique suitable for networks with properties defined by 3.1 and with a very large number of sites (and, so, chains). Let $\bar{N}(S, n)$ be the population vector where all chains foreign to site S have the full population and the local chain of S has population n . The approximation we will develop is based on the three assumptions stated below:

Assumptions 3.2

- a. The mean number of customers in a service center j of site T is not altered if the number of customers of a chain $k \in LC(S)$ is decreased to n , $S \neq T$. In our notation:

$$L_j(\bar{N}(S,n)) = L_j(\bar{N}) \quad j \in T$$

- b. The mean number of customers in a service center j of site S , when the population vector is $\bar{N}(S,n)$, is not altered when the number of customers of a chain $k \in FC(S)$ is decreased by one. In our notation:

$$L_j(\bar{N}(S,n) - \bar{e}_k) = L_j(\bar{N}(S,n)) \quad j \in S, k \in FC(S)$$

- c. The mean throughput of a chain $k \in FC(S)$ is not altered if the number of customers of a site S is decreased to n . In our notation:

$$\lambda_k(\bar{N}(S,n)) = \lambda_k(\bar{N}) \quad k \in FC(S)$$

We now show that these three assumptions are exact in the limit as the number of sites grows to infinite. Without loss of generality, let us assume that there is only one service center per site. Therefore, by the symmetry of the problem:

$$L_j(\bar{N}) = N \quad \forall j$$

Now, letting the number of customers of chain $k \in LC(S)$ decrease to n , we have:

$$L_j(\bar{N}(S,n)) = N - a(n) \quad j \notin S$$

$$L_j(\bar{N}(S,n)) = b(n) \quad j \in S$$

where $a(n)$ and $b(n)$ are functions of n , $a(n) \leq N$. The above is true since, by the symmetry, the average number of customers in all service centers $j \notin S$ will suffer the same perturbation $a(n)$ when the number of customers of chain $k \in LC(S)$

decreases to n . The total number of customers in the network must sum to $(M-1)N + n$, so:

$$(N - a(n))(M-1) + b(n) = (M-1)N + n$$

$$a(n) = \frac{b(n) - n}{M-1}$$

If $b(n) \leq C$ (where C is a constant that does not increase with M), i.e., if the average number of customers in the service center of site S does not increase with M without bound when the number of customers of chain $k \in LC(S)$ decreases, then $a(n) \leq C/(M-1)$. Therefore, if $b(n) \leq C$, $a \rightarrow 0$ as $M \rightarrow \infty$, i.e., the average number of customers of a site $T \neq S$ is not affected when the population of chain $k \in LC(S)$ decreases.

To show that $b(n)$ is indeed $< 2N$, we note that, by the symmetry of the network, the maximum perturbation will occur when $n=0$, i.e., $(a(n))_{\max} = a(0) = b(0)/(M-1)$. This implies that $b(n) \leq b(0) + N$. We now show that $b(0) < N$. The average utilization of center $i \in S$ when the population is $\bar{N}(S,0)$ is given by:

$$\begin{aligned} U_i(\bar{N}(S,0)) &= \sum_{k \neq i} \lambda_k(\bar{N}(S,0)) a_{ik} & i \in S, k \in LC(S) \\ &= \lambda_f(\bar{N}(S,0)) a_{if}(M-1) & f \in FC(S) \\ &= \lambda_0 a_f \end{aligned}$$

where $a_f = \Theta_{f,S}$, $\lambda_0 = \lambda_f(\bar{N}(S,0))$ for any $f \in FC(S)$. For center $j \in T \neq S$

$$U_j(\bar{N}(S,0)) = \sum_{k \neq j} \lambda_k(\bar{N}(S,0)) a_{jk} \quad j \in T, k \in LC(S)$$

$$= \lambda_j(\bar{N}(S,0))e_{ij}(M-2) + \lambda_j(\bar{N}(S,0))e_{ij} \quad j \in T, i \in LC(T), f \in FC(T)$$

$$= \lambda_0 e_{ij} \frac{M-2}{M-1} + \lambda_0 e_{ij}$$

Therefore,

$$\begin{aligned} U_j(\bar{N}(S,0)) - U_i(\bar{N}(S,0)) &= \lambda_0 e_{ij} - \lambda_0 e_{ij} \left(1 - \frac{M-2}{M-1}\right) \\ &= \lambda_0 \left(e_{ij} - \frac{e_{ij}}{M-1}\right) \end{aligned}$$

As $M \rightarrow \infty$ $U_j(\bar{N}(S,0)) > U_i(\bar{N}(S,0))$ $j \in T \neq S, i \in S$ which, by the symmetry of the network, implies that $L_j(\bar{N}(S,0)) > L_i(\bar{N}(S,0)) = \mu(0)$. Therefore, $\mu(0) < N$.

We have just shown that assumption 3.2a is exact as $M \rightarrow \infty$. Assumption 3.2b can also be proven to be exact in the limit as $M \rightarrow \infty$ by showing that a unity variation in the number of customers of a chain $i \in LC(T)$ produces a $O(M)$ variation in the mean queue length of a center $j \in T$. We leave this proof to Appendix 1 (*). Assumption 3.2c follows from equation (2.1) and assumptions 3.2a and 3.2b, i.e.,

$$\sum_i W_{in}(\bar{N}(S,n)) = \sum_i e_{in} [1 + L_i(\bar{N} - \bar{e}_i)] + \frac{e_{if}}{M-1} [1 + L_i(\bar{N}(S,n) - \bar{e}_i)] \quad i \in S,$$

by assumptions 3.2a and 3.2b

$$= \sum_i W_{in}(\bar{N}) + O(M)$$

Using equation (2.1), $\lambda_k(\bar{N}(S,n)) = \lambda_k(\bar{N})$ as $M \rightarrow \infty$ for $k \in FC(S)$.

(*) An alternative proof of assumption 3.2a follows from assumption 3.2b. From assumption 3.2b for $n = N$, we have (see proof in appendix 1):
 $L_j(\bar{N}(S, N-1)) = L_j(N) + O(M)$, $L_j(\bar{N}(S, N-2)) = L_j(N) + 2O(M)$, ...
 $L_j(\bar{N}(S, n)) = L_j(N) + (N-n)O(M) = L_j(N) + O(M)$, if N is finite.

We now apply assumptions 3.2 to the MVA equations. We further assume that all the service centers in the network are single server fixed rate (SSFR) or IS. From equation (2.2) and (2.3a), for a site S :

$$L_j(\bar{N}(S,n)) = \sum_i \lambda_i(\bar{N}(S,n)) a_{ij} [1 + L_j(\bar{N}(S,n) - \bar{e}_i)] \quad j \in S$$

$$= \sum_{i \neq k} \lambda_i(\bar{N}) a_{ij} [1 + L_j(\bar{N}(S,n))] + \lambda_k(\bar{N}(S,n)) \theta_{kj} W_{kj}(\bar{N}(S,n)) \quad j \in S, k \in LC(S),$$

by assumptions 3.2.b and and 3.2.c

$$L_j(\bar{N}(S,n)) = \frac{\Lambda_j x_{ij} + \lambda_k(\bar{N}(S,n)) \theta_{kj} W_{kj}(\bar{N}(S,n))}{1 - \Lambda_j x_{ij}} \quad j \neq IS, j \in S \quad (3.2.1a)$$

and similarly, using (2.3c) instead of (2.3a)

$$L_j(\bar{N}(S,n)) = \Lambda_j x_{ij} + \lambda_k(\bar{N}(S,n)) \theta_{kj} W_{kj}(\bar{N}(S,n)) \quad j = IS, j \in S \quad (3.2.1b)$$

where

$$\Lambda_j = \lambda_j(\bar{N}) \theta_{jj} (M - 1) \quad j \in FC(S), j \in S$$

$$= \lambda_k(\bar{N}) \theta_{kj}, \quad k \in LC(S), j \in S \quad (3.2.2)$$

by the symmetry of our problem. From equation (2.1)

$$\lambda_k(\bar{N}(S,n)) = \frac{n}{\sum_{i \in S} \theta_{ik} W_{ik}(\bar{N}(S,n)) + \sum_{\substack{i \in S \\ i \neq IS}} a_{ik} [1 + L_i(\bar{N}(S,n) - \bar{e}_i)] + \sum_{\substack{i \in S \\ i = IS}} a_{ik}}$$

and by assumption 3.2 and the symmetry of the network

$$\lambda_k(\bar{N}(S,n)) = \frac{n}{\sum_{i \in S} \theta_{ik} W_{ik}(\bar{N}(S,n)) + D_k} \quad (3.2.3)$$

where

$$D_k = \sum_{\substack{j \in S \\ j \neq IS}} a_{jk} [1 + L_j(\bar{N})] + \sum_{\substack{j \in S \\ j = IS}} a_{jk}$$

(3.2.4)

Equations (3.2.2) and (3.2.3) have an intuitive explanation. When equations (3.2.1) are compared to the exact MVA equations for mixed networks [Zabo81, Lave83], we can conclude that the combined effect of all foreign arrivals to a service center j of site S is Poisson with rate λ_j . Furthermore, equations (3.2.3) and (3.2.4) indicates that a customer of a site S "sees" the complement as an infinite server with mean delay D_j . Therefore, the whole homogeneous network can be reduced to the solution of a single site S as indicated in Figure 3.1. In this Figure, all foreign arrivals to site S are represented by a Poisson arrival with rate λ . A customer from site S which goes foreign visits an infinite server

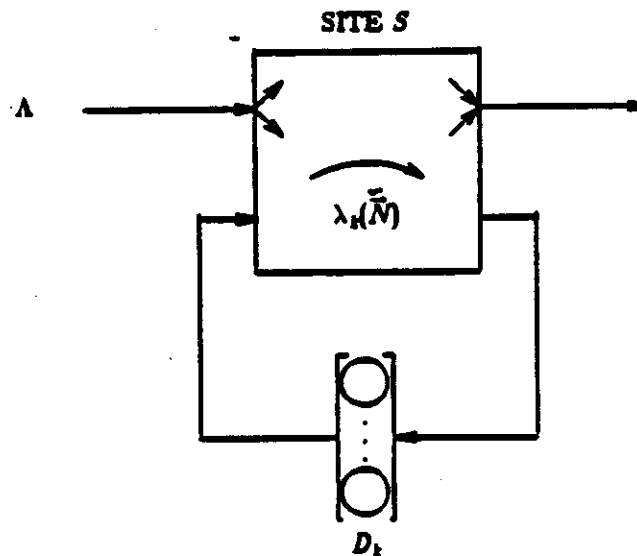


Figure 3.1. The approximate solution of a homogeneous network with many sites.

delay before returning to its local site.

The solution of a homogeneous network as defined in 3.1 with a very large number of chains can be easily obtained by solving recursively the equations obtained above. Below we briefly outline the iterative algorithm. More details will be given in the next section where the algorithm is extended to general queueing networks.

Step 1. Estimate the value for the mean throughput $\lambda_N(\bar{N})$ of site S and delay D_s .

Step 2. Use equations (3.2.1) through (3.2.4) and (2.3), obtaining the results for site S using MVA.

Step 3. Compare the current estimates of queue lengths with previous estimates using the equation below and terminate if this result is less than a specified threshold. Otherwise go to Step 2.

$$\frac{|L_s^{(t)} - L_s^{(t-1)}|}{N} < \text{threshold}$$

(where the superscript indicates the number of iterations.)

The computational costs of a single iteration is $O(JN)$ where J is the number of service centers in a site and N is the number of customers per site. The convergence of this iterative procedure is very fast (in the order of ten iterations when the threshold is chosen to be 10^{-4}), but gets worse when there is a service center whose utilization approaches one (say greater than .99).

Figure 3.2 shows the response time obtained after solving a homogeneous network exactly using MVA, versus the number of sites in the network. In this Figure the limiting result obtained by using the approximation developed in this section is also shown. The parameters for the example are shown in Table 3.1.

From the Figure we note that the response time quickly approaches its limiting value. This behavior was typical for several tested cases. In fact, the limiting behavior seems to be a very good approximation for five or more sites. These results suggest that we may be able to use similar development to solve non-symmetric networks. In the next section we present such development.

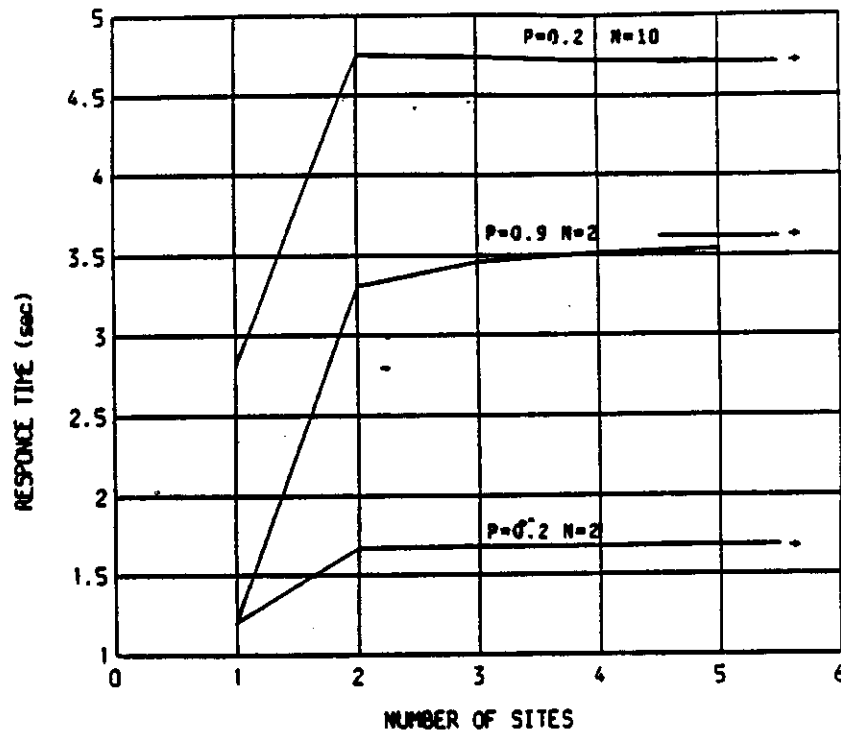


Figure 3.2. Response times versus the number of sites and the limiting result.

3.3. A Clustering Approximation Technique for Product Form Queuing Network Models with Single and Multiple Server Service Centers.

In the previous section we developed an approximation technique suitable for large homogeneous LOCUS type of networks, given that a job that requests

Service Times (sec)		
	job executing in a local site	job executing in a foreign site
CPU	0.05	0.1
disks (2)	0.06	0.06
terminals	3	
probability of executing at disk 1:		0.5
average number of local CPU/disk cycles:		10
probability of requesting a service at a foreign site after a disk service:		P
number of customers local to a site:		N

Table 3.1. Parameters for an homogeneous network.

service in a foreign site randomly chooses the processing site among all foreign sites. In that method the whole network is collapsed to a single subnetwork which represents the resources of a site (all sites are identical). The complement of this subnetwork is represented by Poisson arrivals and by an infinite server service center. Although the method is efficient it can solve only a very limited class of queueing networks. In this section we extend the approach used in the previous section and develop an approximation technique for general queueing network models, suitable for networks with a large number of chains.

We start our development by considering again LOCUS type queueing network models. We divide the entire network into subnetworks, each one representing the resources of a particular site. Consider a particular subnetwork S . The chains representing the behavior of different classes of customers fall naturally into two categories. One category consists of the chains that represent customers logged on at site represented by subnetwork S . These customers tend to have much higher resource utilizations at the site than do customers logged

on at other sites. The other category consists of the chains that represent customers logged on at other sites. As in the previous section, we refer to the two categories of chains by calling the former local chains and the latter foreign chains.

For illustration purpose, let us consider a LOCUS distributed system consisting of only two sites (referred as site A and B respectively). A queueing model of this system is shown in Figure 3.3. In this example, the behavior of customers logged on site A (B) is represented by chain A (B). Therefore, we refer to chain A (B) as the local chain of site A (B) and chain B (A) as the foreign chain of site B (A). To apply our approximation technique, we divide the network in subnetworks. In this example a subnetwork will be formed by "clustering" the resources of a site and the local chains of this site. Therefore,

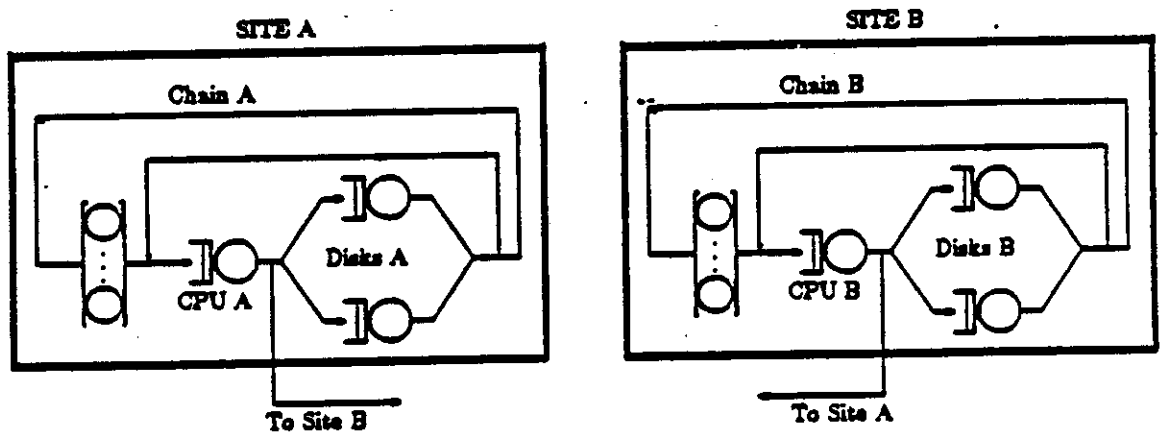


Figure 3.3. A distributed system with two sites.

the network of Figure 3.3 will be divided into two subnetworks. It remains to be found a representation of the complement of these subnetworks. This is the subject of the development shown in the next paragraphs. As we will show, a representation will be found from the MVA equations after simple algebraic

manipulations.

The clustering of service centers and local chains into subnetworks is appropriate to LOCUS-type of queueing networks. However, we can extend this approach to be a general method for large multiple chain product form queueing networks. In the extension a network is still partitioned into subnetworks representing a subset of the resources of the entire network. Furthermore, we still have two categories of chains for each subnetwork. Although the adjectives "local" and "foreign" have lost some of their original meaning, it is convenient to retain the terminology. We wish to choose the subnetworks and their local and foreign chains so that it remains true that the set of local chains with respect to a subnetwork contributes more to the total utilization at the service centers in the subnetwork than those chains that are foreign with respect to the subnetwork. In other words, chains and service centers are clustered into subnetworks according to the contribution of the chains to the total utilization of the centers.

3.3.1 Product Form Networks with SSFR and IS Service Centers

We consider product form queueing networks consisting of only SSFR and IS service centers. (In the next section we extend the analysis to account for multiple server service centers.) We divide the network into subnetworks which need not be disjoint, but whose union includes all service centers. Therefore, a subnetwork may represent resources already represented in any other subnetwork. For each subnetwork we designate a subset of the chains that visit the subnetwork as local and the remaining chains that visit the subnetwork as foreign. The subset of local chains of a subnetwork may overlap with the subset of local chains of another subnetwork. We allow a set of local chains of a subnetwork to be empty. Furthermore, the union of all sets of local chains may not

include all chains. We refer to the set of chains that do not belong to any set of local chains as the local chains of an "empty subnetwork".

Our approach consists of:

- (1) preserving the recursion in the MVA equations involving the effect on the mean queue sizes in a subnetwork of removing a local chain customer.
- (2) approximating the effect on the mean queue sizes in a subnetwork and its complement of removing foreign chain and local chain customers, respectively.

We base our approximation in the following assumptions:

- (1) the throughputs of all foreign chains are not affected if an arbitrary number of local chain customers are removed from the network.
- (2) the mean waiting times of a local chain at any service center in the complement is not affected if an arbitrary number of local chain customers are removed from the network.

Note that these assumptions are identical to the ones used in the previous section, when a homogeneous LOCUS type network was considered. They are reasonable if local chain customers contribute little to the utilization of service centers in the complement and if foreign chain customers contribute little to the utilization of service centers in the subnetwork.

In analogy to the previous section, let S denote a particular subnetwork, $LC(S)$ the set of local chains for S and $FC(S)$ the set of foreign chains for S . $\bar{N}(S, \bar{n})$ is the population vector where all foreign chains have the full population

and the local chains have population \bar{n} . Now, using equations (2.2) and (2.3a) for subnetwork S:

$$L_{cj}(\bar{N}(S, \bar{n})) = \lambda_c(\bar{N}(S, \bar{n}))a_{cj} \left[1 + \sum_{i=1}^K L_{ij}(\bar{N}(S, \bar{n}) - \bar{e}_c) \right] \quad j = 1, \dots, J_1, j \in S, c \in FC(S)$$

Applying assumption (1) about the throughputs of foreign chains and Schweitzer's approximation yields:

$$L_{cj}(\bar{N}(S, \bar{n})) = \lambda_c(\bar{N})a_{cj} \left[1 + L_j(\bar{N}(S, \bar{n})) - \frac{L_{cj}(\bar{N}(S, \bar{n}))}{N_c} \right]$$

from which

$$L_{cj}(\bar{N}(S, \bar{n})) = \frac{\lambda_c(\bar{N})a_{cj}}{1 + \lambda_c(\bar{N})a_{cj}/N_c} [1 + L_j(\bar{N}(S, \bar{n}))] \quad j = 1, \dots, J_1, j \in S, c \in FC(S) \quad (3.3.1a)$$

Similarly we can obtain:

$$L_{kj}(\bar{N}) = \frac{\lambda_k(\bar{N})a_{kj}}{1 + \lambda_k(\bar{N})a_{kj}/N_k} [1 + L_j(\bar{N})] \quad j = 1, \dots, J_1, j \notin S, k \in LC(S) \quad (3.3.1b)$$

We write

$$L_j(\bar{N}(S, \bar{n})) = \sum_{c \in FC(S)} L_{cj}(\bar{N}(S, \bar{n})) + \sum_{k \in LC(S)} L_{kj}(\bar{N}(S, \bar{n}))$$

Substituting (3.3.1a) into the first sum and (2.2) into the second sum yields

$$L_j(\bar{N}(S, \bar{n})) = U_j [1 + L_j(\bar{N}(S, \bar{n}))] + \sum_{k \in LC(S)} \lambda_k(\bar{N}(S, \bar{n})) \theta_{kj} W_{kj}(\bar{N}(S, \bar{n}))$$

from which it follows that

$$L_j(\bar{N}(S, \bar{n})) = \frac{U_j + \sum_{k \in LC(S)} \lambda_k(\bar{N}(S, \bar{n})) \theta_{kj} W_{kj}(\bar{N}(S, \bar{n}))}{1 - U_j} \quad j = 1, \dots, J_1, j \in S \quad (3.3.2)$$

where:

$$U_j = \sum_{i \in FC(S)} \frac{\lambda_i(\bar{N}) a_{ij}}{1 + \lambda_i(\bar{N}) a_{ij} / N_i} \quad j = 1, \dots, J_1, j \in S \quad (3.3.3)$$

Using assumption (2) about the mean waiting times of a local chain in the complement and equation (2.1) yields

$$\lambda_k(\bar{N}(S, \bar{n})) = \frac{n_k}{\sum_{j \in S} \theta_{kj} W_{kj}(\bar{N}(S, \bar{n})) + \sum_{j=1, \dots, J_1}^{j \notin S} \theta_{kj} W_{kj}(\bar{N}) + \sum_{j=1, \dots, J}^{j \notin S} a_{kj}} \quad k \in LC(S)$$

Using Schweitzer's approximation and equation (2.3a) yields

$$\theta_{kj} W_{kj}(\bar{N}) = a_{kj} \left[1 + L_j(\bar{N}) - \frac{L_{kj}(\bar{N})}{N_k} \right] \quad j = 1, \dots, J_1, j \notin S, k \in LC(S)$$

Substituting equation (3.3.1b) into the above equation yields

$$\theta_{kj} W_{kj}(\bar{N}) = \frac{a_{kj}}{1 + \lambda_k(\bar{N}) a_{kj} / N_k} [1 + L_j(\bar{N})]$$

Therefore,

$$\lambda_k(\bar{N}(S, \bar{n})) = \frac{n_k}{\sum_{j \in S} \theta_{kj} W_{kj}(\bar{N}(S, \bar{n})) + D_k} \quad k \in LC(S) \quad (3.3.4)$$

where

$$D_k = \sum_{j=1, \dots, J_1}^{j \notin S} \frac{a_{kj}}{1 + \lambda_k(\bar{N}) a_{kj} / N_k} [1 + L_j(\bar{N})] + \sum_{j=1, \dots, J}^{j \notin S} a_{kj} \quad k \in LC(S) \quad (3.3.5)$$

Equations (3.3.2) - (3.3.5) and for $j \in S$ equations (2.3a), (2.3c) constitute the set of equations for subnetwork S that we wanted to obtain. When comparing these equations with the exact MVA equations for mixed networks [Zaho81] it is easy to see that in terms of mean values (1) a foreign chain l effects service center $j \in S$ as if it were an open chain, i.e., Poisson arrivals, with arrival rate to center j given by

$$\lambda_i(\bar{N})\theta_{ij}/(1 + (\lambda_i(\bar{N})a_{ij}/N_i))$$

and (2) a local chain k customer that leaves the subnetwork "sees" the complement as an infinite server with mean delay D_j . Thus when we solve a particular subnetwork we reduce the network to that shown in Figure 3.4. In that Figure all foreign chains are represented by open chains and the delay when a local

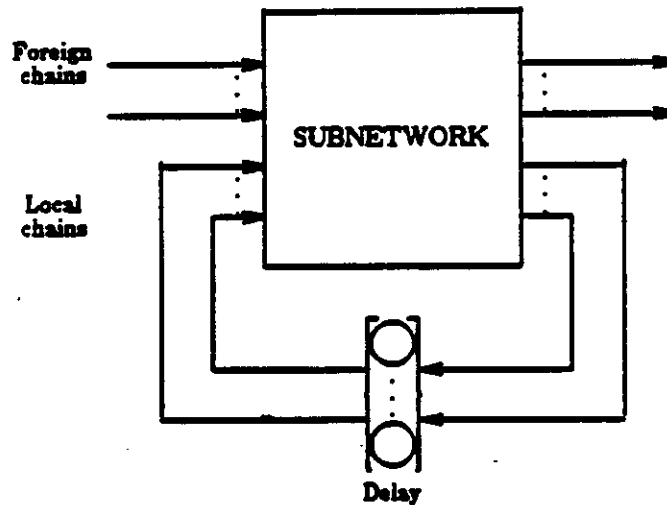


Figure 3.4. Representation of the effect of the complement for local and foreign chains.

chain customer leaves the subnetwork is represented by an infinite server. The effect of the complement of a subnetwork is then represented differently for the local and foreign chains. For local chains the complement is represented as an infinite server while for foreign chains the complement is represented by Poisson arrivals. It is worthwhile to observe that, for homogeneous networks defined in definition 3.1, the equations obtained above reduce to the ones obtained in section 3.2, when the number of sites grows to infinite. It should also be noted that the open chain arrival rate for foreign chain i at service center j in the subnetwork is less than the throughput $\lambda_i(\bar{N})\theta_{ij}$ of the corresponding closed foreign chain. As proved in [Zaho83] if in a single chain closed network the closed chain

is replaced by an open chain leaving the same throughput then the mean queue sizes increase, possibly substantially. (This result also appears to hold for multiple chain networks). The reduced arrival rate we use tends to compensate for the larger mean queue sizes that result from replacing foreign closed chains by open chains. Later we will present results of extensive empirical tests that demonstrate the accuracy that is achieved.

Let us return to the example of Figure 3.3. In Figure 3.5 we show subnetwork *A* (representing the resources from site *A*) and its complement. The IS service center added to the subnetwork represents the complement of the local chain *A*. The complement of the foreign chain is represented by a set of open

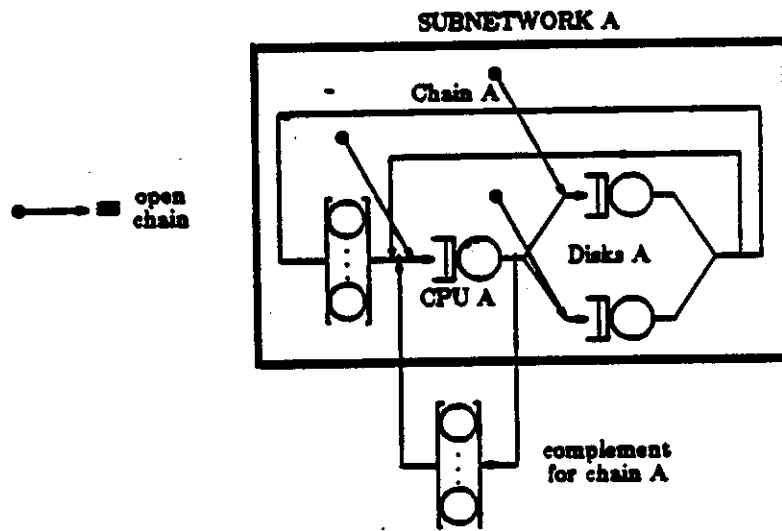


Figure 3.5. Subnetwork *A* and its complement.

chains. Each open chain in this set represents the effect of foreign chain *B* to a particular center in subnetwork *A*. Note that the chains have different arrival rates.

3.3.2 Product Form Networks with Multiple Server (MS) Service Centers.

For networks with MS service centers, we apply the same assumptions used in the previous section. The development is similar, but for MS centers equation (2.3d) is used instead of equation (2.3a). We also assume that for MS service centers:

$$q_j(\bar{N}(S, \bar{n}) - \bar{e}_i) + PB_j(\bar{N}(S, \bar{n}) - \bar{e}_i) = q_j(\bar{N}(S, \bar{n})) + PB_j(\bar{N}(S, \bar{n})) - \frac{q_j(\bar{N}(S, \bar{n}))}{N_i} \quad j \in S, i \in FC(S) \quad (3.3.6)$$

This assumption is similar to Schweitzer's approximation, since $q_b(\bar{N}) = L_b(\bar{N}) - \lambda_b(\bar{N})a_b$, and was found to work well in our empirical tests.

Using equations (2.2) (2.3d) and (3.3.6) and applying assumption (1) about foreign chain throughputs we obtain:

$$q_c(\bar{N}(S, \bar{n})) = \frac{\lambda_c(\bar{N})a_c}{1 + \lambda_c(\bar{N})a_c / (N_c M_c)} \left[\frac{q_j(\bar{N}(S, \bar{n})) + PB_j(\bar{N}(S, \bar{n}))}{M_j} \right] \quad j \in S, c \in FC(S) \quad (3.3.7)$$

which is similar to equation (3.3.1a).

Following the same steps leading to equations (3.3.2) and (3.3.3) we obtain:

$$q_j(\bar{N}(S, \bar{n})) = \frac{UM_j PB_j(\bar{N}(S, \bar{n})) / M_j + \sum_{i \in FC(S)} \lambda_i(\bar{N}) a_i |W_{ij}(\bar{N}(S, \bar{n})) - z_{ij}|}{1 - UM_j / M_j} \quad j = J_1 + 1, \dots, J_2, j \in S \quad (3.3.8)$$

where:

$$UM_j = \sum_{i \in FC(S)} \frac{\lambda_i(\bar{N}) a_i}{1 + \lambda_i(\bar{N}) a_i / M_i N_i} \quad j = J_1 + 1, \dots, J_2, j \in S \quad (3.3.9)$$

Similarly, we obtain the expression for the throughput of local chains:

$$\lambda_k(\bar{N}(S, \bar{n})) = \frac{n_k}{\sum_{j \in S} \theta_{kj} W_{kj}(\bar{N}(S, \bar{n})) + D_k + DM_k} \quad k \in LC(S) \quad (3.3.10)$$

where

$$DM_k = \sum_{j = \substack{j \in S \\ j_1+1, \dots, j_2}} \frac{a_{kj}}{1 + \lambda_k(\bar{N}) a_{kj} / N_k M_j} \left[\frac{g_j(\bar{N}) + PB_j(\bar{N})}{M_j} \right] \quad (3.3.11)$$

As in the previous section, equations (3.3.8) - (3.3.11) have the same form as the exact equations for a mixed network with MS service centers. Therefore, the same interpretation holds, i.e., a foreign chain l effects multiple server center j as an open chain with arrival rate

$$\lambda_l(\bar{N}) \theta_{lj} / [1 + (\lambda_l(\bar{N}) a_{lj} / M_j N_l)]$$

and a local chain customer that leaves the subnetwork "sees" the complement as an infinite server with mean delay $D_k + DM_k$.

In order to complete the set of equations for subnetwork S we need an expression for $PB_j(\bar{N}(S, \bar{n}))$, $j \in S$, which is the probability that all servers are busy at MS center $j \in S$ considering all chains in the network. Rather than derive an approximate expression for these probabilities we will solve the above mixed network. These subnetworks are the ones obtained after the clustering of chains and service centers. The Poisson arrivals represent the complement of the foreign chains. It is well known that the solution of a mixed network can be reduced to the solution of a corresponding closed network [Lave83]. The solution is obtained by recursively solving a corresponding closed network with the degraded service rates [Lave83, Saue83]:

$$\mu_j(n) = \mu_j(n) \frac{\alpha_j(n-1)}{\alpha_j(n)}$$

where

$$\alpha_j(n) = \frac{M_j^{M_j-n-1}}{M_j^{M_j-1}} \times \frac{1}{(1 - UM_j/M_j)^{n+1}} + \sum_{i=0}^{n-2} \frac{(i+n)!}{i! n!} UM_j \left\{ \frac{1}{\prod_{h=i+1}^{n+1} \mu_j(h)} - \frac{1}{\prod_{h=i+1}^{M_j-1} \mu_j(h) M_j^{n-M_j+i+1}} \right\} \quad \text{for } n \geq M_j - 1 \quad (3.12a)$$

$$\alpha_j(n) = \frac{1}{(1 - UM_j/M_j)^{n+1}} \quad \text{for } n < M_j - 1 \quad (3.12b)$$

and

$$\mu_j(n) = \begin{cases} n & \text{if } n \leq M_j \\ M_j & \text{otherwise} \end{cases}$$

Note that the average queue lengths (not including the customers in service) of foreign chain customers at center $j \in S$ are obtained from [Lave83]:

$$q_{c_j}(\bar{N}) = \frac{\lambda_{c_j}(\bar{N}) a_{c_j}}{1 + \frac{\lambda_{c_j}(\bar{N}) a_{c_j}}{N_c M_j}} \left(\sum_{i=0}^{|\bar{N}_S|} \frac{(i+1) P_{c_j}(i|\bar{N})}{\mu'_{c_j}(i+1)} - 1 \right) \quad j \in J_2+1, \dots, J_2, \quad j \in S, \quad c \in S \quad (3.3.13)$$

where $|\bar{N}_S| = N_1 + \dots + N_L$, $l, \dots, L \in LC(S)$.

3.3.3. The Algorithm.

The following algorithm can be used to iteratively solve the equations obtained for all subnetworks.

Step 1. Obtain an initial estimate of the throughput of all chains and the mean queue sizes of all service centers. Obtain an initial estimate for

the equilibrium marginal queue size probabilities of all multiple server centers.

Step 2. Loop through the subnetworks and for each one solve the corresponding mixed queueing network recursively. New estimates for $\{D_j\}$, $\{DM_j\}$, $\{U_j\}$, $\{UM_j\}$ and $\{\alpha_j(n)\}$ are obtained just before calculating the performance measures for a subnetwork, using (3.3.5), (3.3.11), (3.3.3), (3.3.9), (3.3.12). (Note that for a particular subnetwork this recursion is only over the population of local chains. The populations of all foreign chains are fixed. Furthermore the only MS service centers involved are those belonging to this subnetwork).

Step 3. Compare the current estimates of mean queue lengths with the previous estimates using (3.3.a1) below, and terminate if this result is less than a specified threshold. Otherwise go to Step 2).

$$\frac{L_j^{(l)} - L_j^{(l-1)}}{N_j} < \text{threshold} \quad j = 1, \dots, J, \quad l = 1, \dots, K \quad (3.3.a1)$$

(where the superscript indicates the number of the current iteration.)

As initial estimates in step 1 we used:

$$L_{hj}(\bar{N}) = \frac{a_{hj} N_k}{\sum_{i=1}^I a_{hi}} \quad j = 1, \dots, J; \quad k = 1, \dots, K \quad (3.3.a2)$$

$$\lambda_{hk}(\bar{N}) = \frac{N_k}{\sum_{j=1}^{I_1} a_{hj} [1 + L_j(\bar{N}) - L_{hj}(\bar{N}) / N_j] + \sum_{j=I_1+1}^{I_2} a_{hj} [1 + (L_j(\bar{N}) - L_{hj}(\bar{N}) / N_j) / M_j] + \sum_{j=I_2+1}^I a_{hj}} \quad (3.3.a3)$$

$$k = 1, \dots, K$$

$$P_j(i|\bar{N}) = \begin{cases} 1 & \text{if } i = \lfloor \sum_{l \in LC(S)} L_l(\bar{N}) \rfloor \quad j \in S, \quad j = J_1+1, \dots, J_2 \\ 0 & \text{otherwise} \end{cases} \quad (3.3.a4)$$

Instead of using exact MVA to recursively solve a subnetwork, we can use any existing approximation for product form networks. For example, Schweitzer's and the Linearizer approximations can be used when there are no MS service centers in the subnetwork. It is easy to show that if Schweitzer's approximation is used in all subnetworks, the results are identical to using Schweitzer's approximation on the entire network.

As pointed out in section 2.3.2, when there is overlap among subnetworks (i.e., the same service center is represented in different subnetworks) and/or a chain is included in more than one set of local chains of different subnetworks (e.g., $l \in LC(S_i)$ and $l \in LC(S_j)$, $S_i \neq S_j$) the performance values of the elements represented more than once may differ. For example, if a chain l belongs to the set of local chains of subnetworks S_i and S_j , the throughput of that chain calculated when subnetwork S_i is solved may differ from the throughput of the same chain calculated when subnetwork S_j is solved (*). Thus, the algorithm has to decide which value to use. In our empirical tests we observed that the difference among the values is small and the choice is not critical. The following heuristic choice is used by the algorithm: (a) whenever a service center j is represented in more than one subnetwork, the value used for the total queue length of j is the one calculated when subnetwork S_M is solved so that, among all subnetworks S_i , $j \in S_i$, the local chains of S_M are the ones who give the highest contribution to the total utilization of $j \in S_M$; (b) whenever a chain l is in more than one set of

(*) These multiple values also appear in other methods where overlaps among subnetwork occur (e.g., Marie's method [Lave83], the method of surrogate delays [Jaco82]).

local chains, the value of the throughput used is the one calculated when subnetwork S_M is solved so that, among all subnetworks $S_i, i \in LC(S)$ chain i gives the highest contribution to the utilization of all centers $j \in S_M, j \neq IS$. For the final estimates, the values used for queue lengths of customers of a chain i at service center j are the ones calculated for subnetwork S so that, $i \in LC(S)$ and $j \in S$, i.e., the preference is given for parameters calculated locally to a subnetwork, if possible. The reasoning behind this choice is based on our initial assumptions which imply that the approximation should be reasonable if chains and service centers are clustered so that the local chains of a subnetwork S contribute to most of the utilization of a center in S .

Although the development presented in this section is based on the MVA algorithm, any solution technique can be used for the approximation algorithm we described above. Furthermore, different solution techniques can be used to solve different subnetworks. This is true since each subnetwork is solved in isolation and its complement has a very simple representation.

The cost of the algorithm depends on the solution technique used to solve each subnetwork and the way the subnetworks are chosen. The cost per iteration is equal to the sum of the costs to solve each subnetwork. For example, if MVA is used in all subnetworks and there are no MS centers, the cost per iteration is $O(\sum_S [J_S \prod_{i \in LC(S)} (N_i + 1)])$, where J_S is the total number of centers of subnetwork S . The total number of iterations is small, typically around 10 iterations for the tested cases. The memory requirements of the algorithm are on the order of the memory requirements of the subnetwork which used the largest amount of memory. For example, if MVA is used in all subnetworks, the memory requirements are $O(\max_S [J_S \prod_{i \in LC(S)} (N_i + 1)])$.

3.3.4. The Choice of Subnetworks.

There are many ways to choose the subnetworks and the local chains for each subnetwork. For each choice the final results and the total cost of each iteration will differ. In general, larger subnetworks with more local chains yield more accurate but more costly results. As we have mentioned our approximation should be reasonable if the main contributions to the utilization of service centers in a subnetwork are from local chains. However, it was determined after extensive tests that the approximation can be used even if the foreign chains are responsible for up to 50% of the total utilization of a service center. Furthermore, this constraint can be relaxed for service centers with low to moderate total estimated utilization, say less than .6.

We developed a heuristic search algorithm to divide a network into subnetworks and choose the local chains for each subnetwork. The choice of subnetworks and local chains is based on the estimated utilizations of each chain at a service center, obtained from (3.3.a2) and (3.3.a3). The goal of the algorithm can be described as follows:

Given:

1. The estimated utilizations of each chain at each service center.
2. The maximum cost per iteration of a subnetwork.

Find:

A minimum covering set of subnetworks (the subnetworks need not be disjoint), and for each subnetwork a subset of the chains (the local chains).

So that:

1. For each subnetwork, the sum of utilizations of local chains at each service center of that subnetwork is greater than 50% of the total utilization of that center.
2. The cost of each subnetwork is less than the given cost.
3. For a subset of chains, a subnetwork is formed by adding all possible service centers so that the constraints 1 and 2 above are not violated.
4. Constraint 1 may be relaxed if the total utilization of a service center is less than .6.

The user is prompted for the maximum total cost (defined below) per iteration of a subnetwork and an initial number of local chains (INC) per subnetwork. This number will be used as the starting point for the search procedure, i.e., the algorithm starts searching for subnetworks with INC local chains. This number can be decreased or increased automatically to adjust the total cost of a subnetwork up to the maximum allowed cost. The objective of each step of the algorithm is to find a subnetwork that contains at least one service center with moderate to high estimated utilization, which does not violate the cost constraint. (Note that a service center can belong to a subnetwork if its local utilization is higher than .5). The algorithm takes into account that overlap can occur among subnetworks and that there can be subnetworks with no local chains or no service centers (*). The cost function of a subnetwork used by

(*) Chains that do not belong to any set of local chains are clustered in a subnetwork with no service centers. These chains will be the local chains of an empty subnetwork. Therefore, they visit only an IS center representing their complement. This case results from networks which contain chains that do not contribute much to the total utilization of any center. From the set of equations

the algorithm, when MVA is used to solve a subnetwork, is $\prod_{i \in C(S)} (N_i + 1)2^{MS}$ ~~service centers~~, since this function provides an estimate of the computational costs to solve each subnetwork exactly using MVA. Other functions may be used as appropriate.

We classify the centers in the network into critical and non-critical. Critical service centers are SSFR or MS service centers that have total estimated utilization greater than or equal to .6. We also define two functions:

1) "Usage value" (UV) of a chain:

$$UV_k(S) = \sum_{j \in C(S)} U_j \frac{U_{kj}}{(1 - U_{kj})}$$

2) "Local usage" (LU) of a center:

$$LU_j(S) = U_j \frac{\sum_{k \in C(S)} U_{kj}}{1 - \sum_{k \in C(S)} U_{kj}}$$

where:

- U_j = total estimated utilization of center j .
- U_{kj} = estimated utilization of chain k at center j .
- $C(S)$ = critical centers in subnetwork S .

Note that $UV_k(S)$ is a measure of the contribution of chain k to the utilization of critical centers in subnetwork S , and $LU_j(S)$ is a measure of the contribution of the local chains of subnetwork S to the utilization of a center $j \in S$. A detailed description of the algorithm is given in appendix 2.

obtained above we see that the effect of these chains is approximated by a reduction in the capacity of all the centers they visit.

In the tests we will report on next, we did not make use of the automated choice of subnetworks in the first series of experiments. Instead, we based our choice on the estimated utilizations. We then repeated all tests with the subnetworks chosen automatically. For the automated clustering we tried to maintain the same total cost as the manual choice. In the majority of the cases the automated choice was different than the manual choice of subnetworks. However, in some cases the automated choice produced slightly better results and in other cases slightly worse results, for approximately the same cost. Overall, the automated choice produced results with accuracy comparable to the manual choice.

It is important to note that the cost of clustering was negligible compared to the actual computation of the performance measures in all cases.

3.3.5. Empirical Results

We have tested our method on more than 180 queueing networks consisting of randomly selected networks, networks of the type proposed to model LOCUS [Gold83] (e.g., see Figure 3.8) and networks of the type proposed to model packet switching networks [Reis79] (e.g. see Figure 3.9). We divided our tests into two sets of experiments with identical numbers of networks.

The first set consists of networks with SSFR and IS service centers only. We varied the number of chains from 3 to 9 and the total number of customers from 6 to 50. In the majority of the cases at least one service center had utilization greater than .8. For the LOCUS type of models we varied the mean think time from 1 to 20 seconds; the mean local CPU service time from 1 to 100 msec and the mean foreign CPU service time from 1.5 to 150 msec. The average

number of local CPU-disk cycles was varied from 1 to 10; the average disk service time was varied from 35 msec to 150 msec; the probability of making a request to a foreign site (after a CPU service) was varied from .1 to .9. Message sizes varied from 1K bytes to 10K bytes and were transmitted over a 1 to 10M bit channel. For the packet switching network models, the packet size varied from 250 to 2K bytes, and channel speeds from 20 kbps to 100 kbps (both half and full duplex). The window size varied from 1 to 15.

The second set of experiments consisted of networks with at least one MS service center. More than half of the networks had the visit ratio of their chains and service times of the centers randomly selected. The number of chains varied from 3 to 6, the maximum total number of customers was 50, the number of MS service centers varied from 2 to 5 and the maximum number of servers varied from 2 to 15. For the rest of the networks we chose, from the first set of tests, several networks that produced some of the least accurate results and replaced one or more of the centers by a multiple server center. For these networks, the number of chains varied from 4 to 8, the number of MS service centers varied from 1 to 3, the number of servers varied from 2 to 3 and the maximum total number of customers was 50. It is worthwhile to comment that in the second set of tests the exact solution using MVA was very costly and several of the tests took a few hours of CPU time in the VAX11/750, even for networks with a small number of chains. Below we present the results of our experiments.

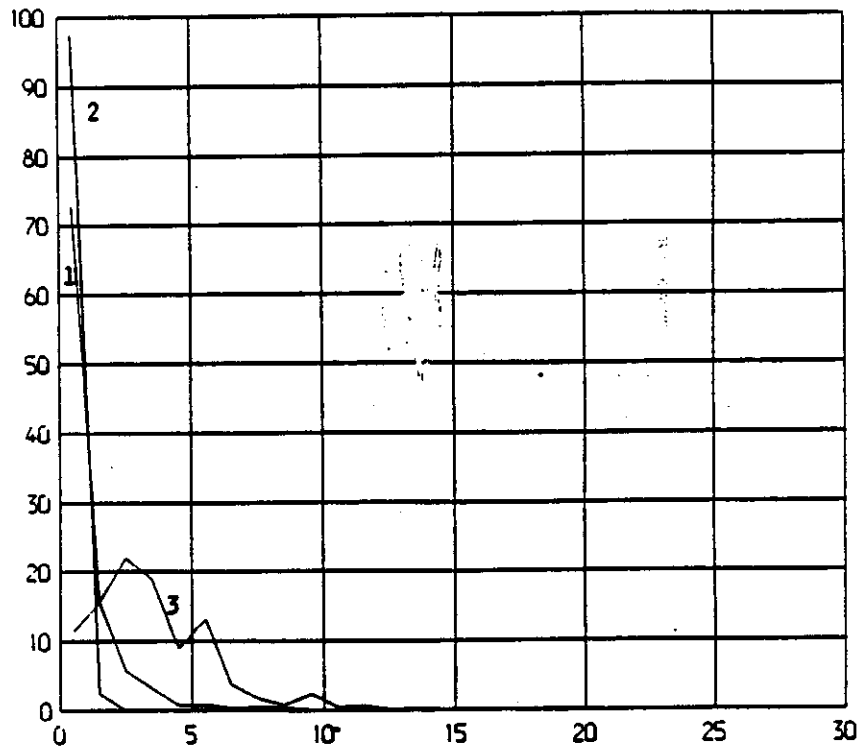
In the first set of experiments (i.e., for networks with SSFR and IS service centers only) each queueing network model was solved exactly using MVA and approximately using exact MVA to solve each subnetwork (MVASUB) and approximately using Linearizer to solve each subnetwork (LINSUB). For com-

parison with existing approximations we also solved the entire network using Schweitzer's approximation (SCH) and Linearizer approximation (LIN). In the second set of experiments (i.e., for networks with at least one MS service center) each queueing network model was solved exactly using MVA and approximately using exact MVA to solve each subnetwork.

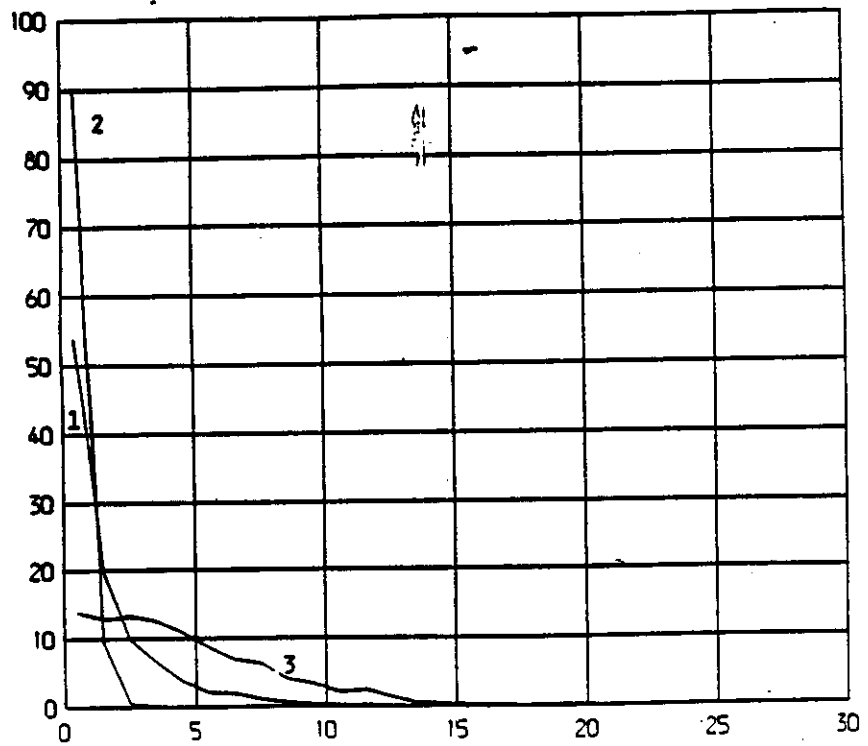
Table 3.2 and Figure 3.6 summarize the errors obtained for the first set of experiments. Table 3.2 gives the average absolute value of the percent errors and the average of the maximum for each network. Separate results are given for mean queue sizes, mean waiting times and throughputs in the columns headed L, W and λ respectively. Figure 3.6 shows densities of the absolute value of percent errors for L, W, and λ . We found virtually the same errors were obtained when we used exact MVA or Linearizer to solve each subnetwork. Since LINSUB is less costly than MVASUB we only plot the results for LINSUB in Figure 3.6. We found that for LINSUB 73.7% (74.5%, 88.7%) of the errors for mean queue sizes (mean waiting times, throughputs) were less than two percent compared with 26.8% (43.8%, 26.9%) for SCH and 99.6% (99.2%, 100%) for LIN.

From Table 3.2 and Figure 3.6 it is clear that Linearizer applied to the entire network is the most accurate approximation. However, as we will illustrate in two large LOCUS examples below, it is by far the most expensive of the approximations and its cost becomes prohibitive for networks with many chains. Schweitzer's approximation applied to the entire network is less costly, but it can yield large errors. It was not uncommon to get maximum errors around 20% and in one network we found a 30% error for a throughput and a 43% error for a mean waiting time. The maximum errors found in our subnetwork approach were 6% for throughputs and 14% for waiting times (with automatic clustering,

(a) λ



(b) L



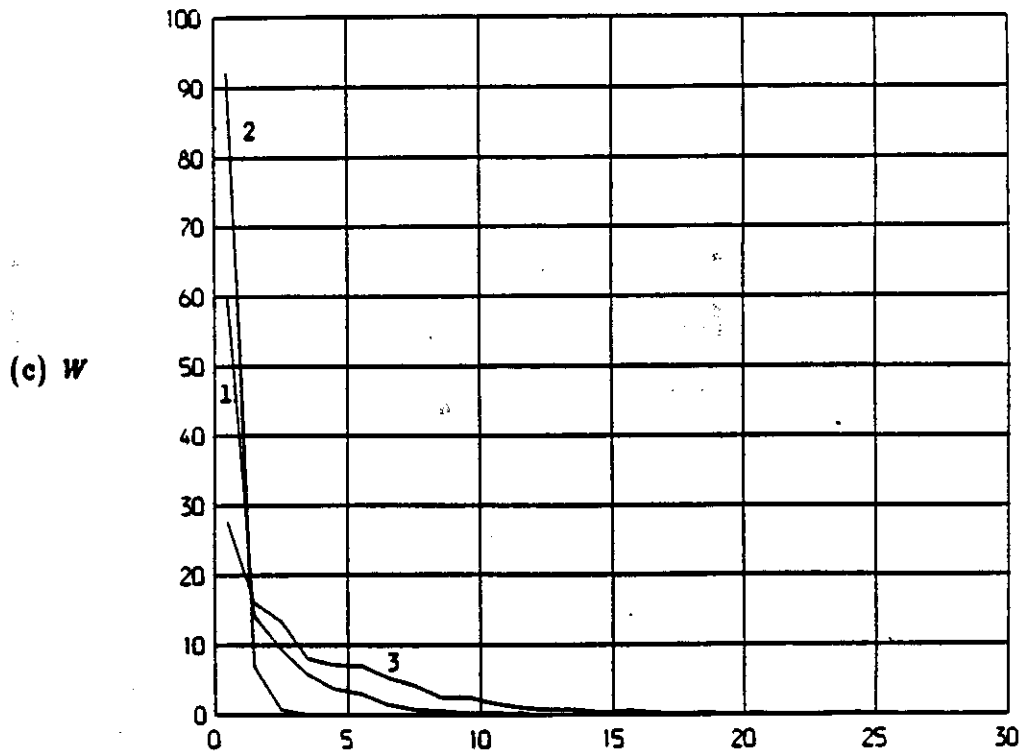


Figure 3.6. Densities of absolute value of percent errors for LINSUB (1), LIN (2) and SCH (3). All-axes in percent.

	Average			Average of Maxima		
	L	W	λ	L	W	λ
MVASUB	1.44	1.38	.81	5.37	5.82	2.10
LINSUB	1.54	1.44	.85	5.67	6.19	2.11
SCH	4.41	3.56	3.48	10.68	13.29	6.56
LIN	.41	.33	.26	1.12	1.34	.49

Table 3.2. Absolute Value of Percent Errors. First Set of Experiments.

with very cheap cost specified and in networks with also contain MS service centers). Note that with larger subnetworks (and more costly solution) the errors may be decreased. Our subnetwork approach provides favorable accuracy/cost characteristics, particularly as the number of chains increases.

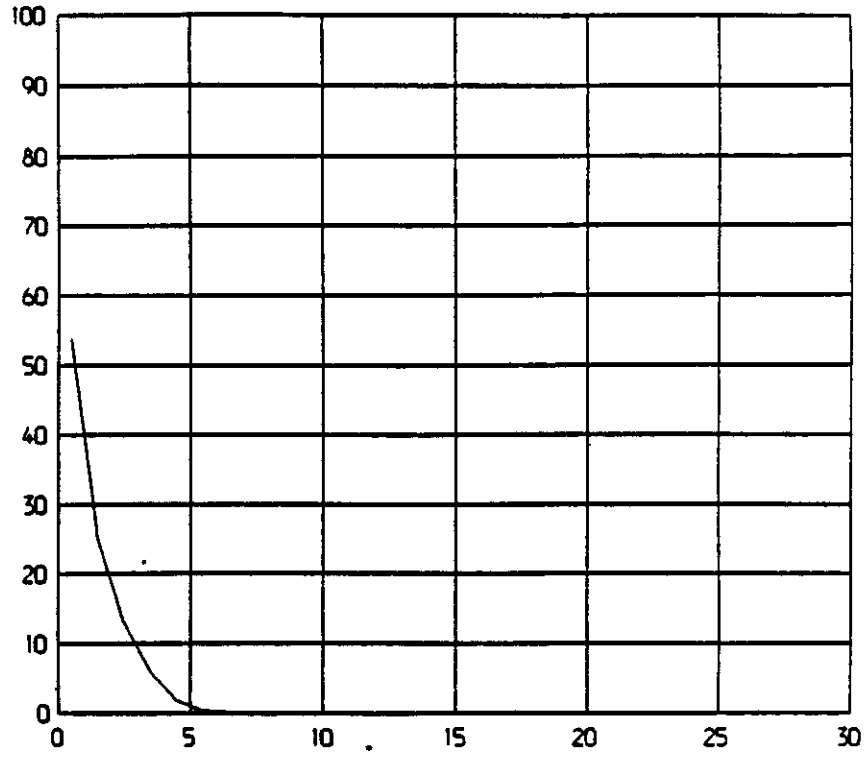
Table 3.3 and Figure 3.7 present results for the second set of experiments. We note that the results are slightly worse than the results obtained for the first set of experiments. This can be explained by the fact that, for this second set, we chose several networks that produced some of the least accurate results in the first set of experiments.

We next present four networks with many chains to illustrate the potential of the approximation technique we have described. Since our initial research goal was motivated by the LOCUS system, we chose as the first two networks LOCUS types of models, as illustrated in Figure 3.8.

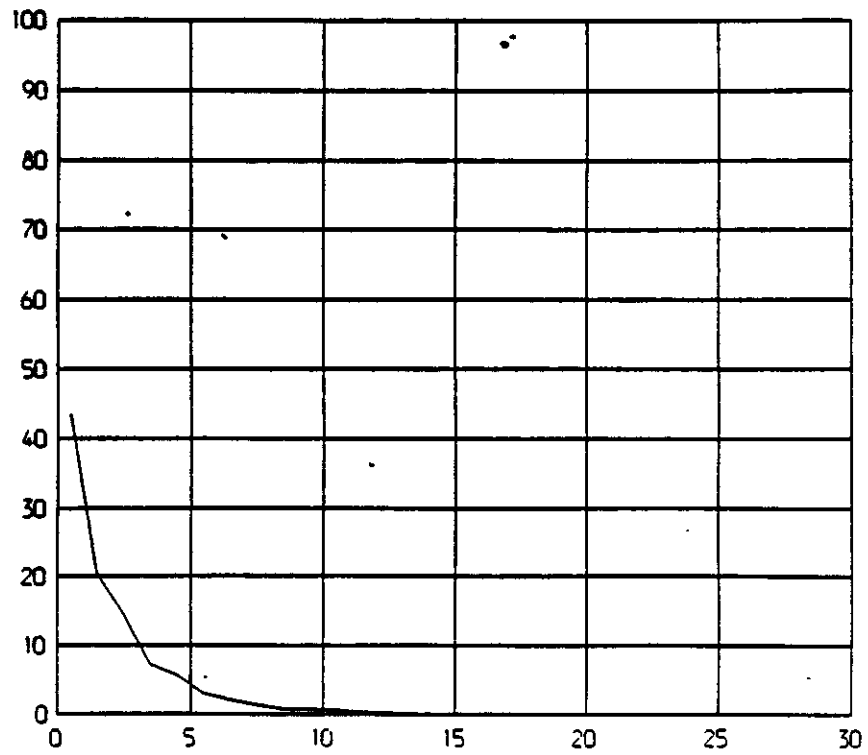
The first network is a model of LOCUS with 10 computer sites connected by a slotted ring communication channel modeled as proposed in [Bux81]. Each site has 1 CPU (modeled as processor sharing queue), 2 disks (modeled as FCFS queues) and only one type of job (thus only 1 chain per site is needed). The slotted ring is modeled by a processor sharing service center and a closed chain visiting only this center. This closed chain has only one job and models the time periods of one cycle during which the slot is empty (see [Bux81]). Therefore we have a network with 11 chains and 41 service centers. For each site we randomly chose from the following parameter values:

- mean think time: 2 to 15 sec.
- mean local CPU service time: 50 to 100 msec.
- mean foreign CPU service time: 7% to 20% greater than local service time.

(a) λ



(b) L



(c) W

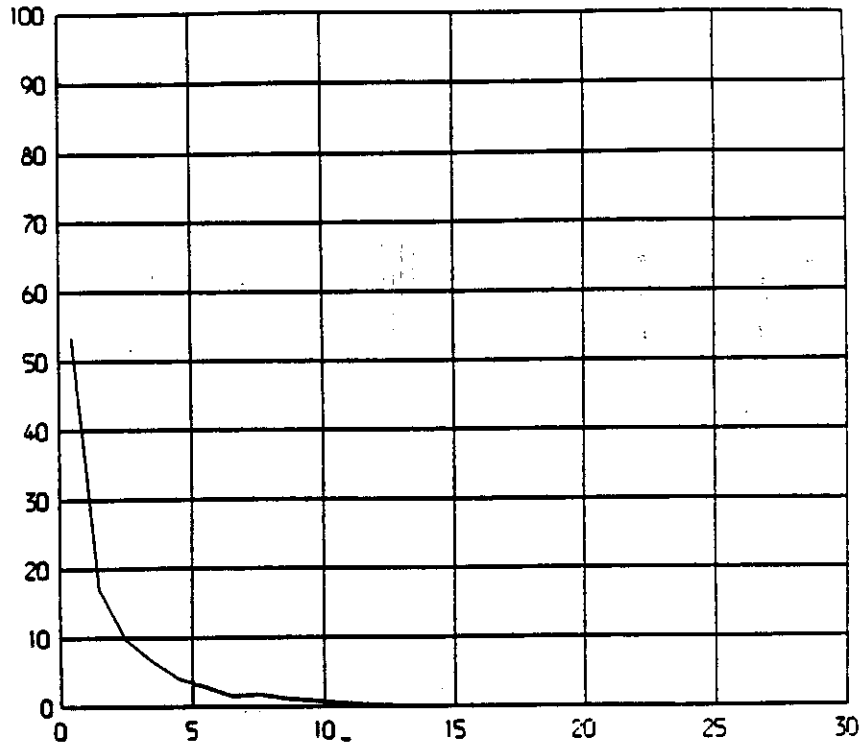


Figure 3.7. Densities of absolute value of percent errors for MVASUB, for networks with MS centers.

	Average			Average of Maxima		
	L	W	λ	L	W	λ
MVASUB	1.96	1.74	1.17	6.76	7.43	2.4

Table 3.3. Absolute Value of Percent Errors. Second Set of Experiments.

- average disk service times: 50 to 80 msec.
- average number of local CPU-disk cycles: 8 - 10.
- probability of making a request to a foreign site after a CPU-disk cycle: 0.1 to 0.4.
- message size: 1000 bytes.
- channel speed: 10 Mbps.
- jobs from one site may be restricted to run at only some of the sites.

- number of jobs per chain: 6 to 12 (except for the chain of the slotted ring model which has 1 job).

The second network is a much larger one and reflects the large number of sites and types of jobs that would be found in a LOCUS network. The model is for a 16 site computer network, again connected by a slotted ring. Each site has 2 or 3 different types of jobs (thus 2 or 3 chains per site). Each site has 1 CPU and 2 disks. The total number of chains and service centers is 41 and 65 respectively.

We randomly chose from the following parameter values for each site:

- mean think time: 1 to 15 sec.
- mean local CPU service time: 10 to 120 msec.
- mean foreign CPU service time: 10 to 75% greater than local CPU service time.
- average disk service time: 40 to 90 msec.
- average number of local CPU-disk cycles: 2 to 10.
- probability of making a request to a foreign site after a CPU-disk cycle: .1 to .8.
- message size: 1000 bytes.
- channel speed: 10 Mbps.
- jobs from one site may visit a subset of sites or all the other sites.
- number of jobs per chain: 1 to 10.

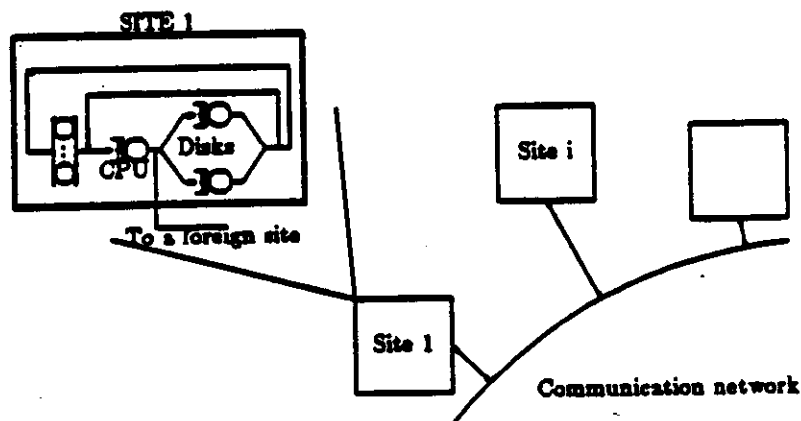


Figure 3.8. A queuing model of a distributed system.

The networks are too large to solve exactly using MVA and we only solved them approximately using LIN, SCH and LINSUB. When LINSUB was used, the first network was manually divided into 11 subnetworks, 10 of them representing the sites and 1 representing the communication channel. Similarly, the second network was manually divided into 17 subnetworks, 16 of them representing the sites and 1 representing the communication channel. The CPU times to approximately solve the networks on a VAX11/750 using LINSUB, SCH and LIN were for network 1: 32 sec, 23 sec and 164 sec respectively and for network 2: 171 sec, 123 sec and 1 hour and 15 min respectively (*). In Table 3.4 we give the absolute value of the percent differences between the LIN solutions and

	Average			Maximum		
	L	W	λ	L	W	λ
11 Chain network						
LINSUB	.19	.18	.04	1.29	1.30	.8
SCH	3.54	2.44	2.09	15.3	20.9	5.75
41 Chain network						
LINSUB	.38	.33	.11	4.61	4.95	.39
SCH	4.90	3.21	3.36	15.8	24.8	10.3

Table 3.4. Comparison with LIN Solutions - Absolute Value of Percent Differences.

the LINSUB and SCH solutions for the two networks. Our LINSUB method provides close to the accuracy of Linearizer applied to the entire network at close to the cost of Schweitzer's approximation applied to the entire network. It

(*) In [DeSo83] we reported higher CPU times to solve these examples with LINSUB and LIN. The reason was that we implemented the Linearizer algorithm as suggested in the original paper [Chan82], i.e., with the core algorithm costing $O(JK^2)$. However, it can be shown that the core algorithm necessary for Linearizer can be implemented with a cost of only $O(JK)$. These new CPU times reflect the new implementation.

appears to be the method of choice when Linearizer is too expensive to apply to the entire network.

The third network is a model of small packet switching network with

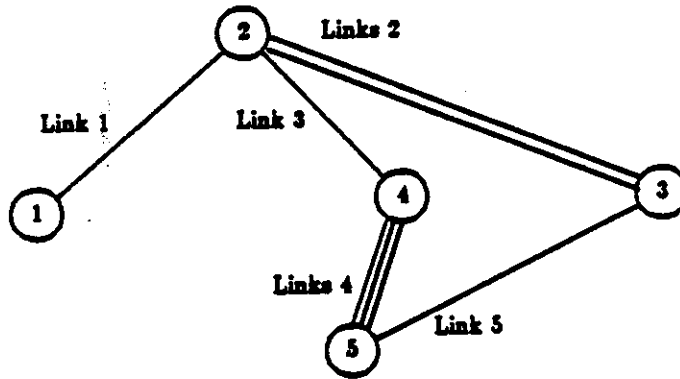


Figure 3.9. A small packet switching network.

window flow control as illustrated in Figure 3.9. There are five virtual circuits, each modeled as a closed chain as proposed in [Reis79]. All the links are half-duplex. Centers 1 to 5 represent groups of links 1 to 5 respectively and centers 6 to 10 represent the sources of virtual routes 1 to 5 respectively. In this example there are two identical links connecting node 2 to node 3 and three identical links connecting node 4 to node 5. We assume that the switching node 2 (4) has only one internal queue to route packets to node 3 (5) and chooses the first available free link for that. Therefore, we model the multiple links 2 and 4 as MS service centers with 2 and 3 servers respectively.

Table 3.5 presents the parameter values for this network. The automated clustering algorithm divided the network into four non-overlapping subnetworks as shown in Table 3.6. Note that subnetwork 4 has no local chains which means that the complement for this subnetwork is represented only by Poisson arrivals.

Table 3.7 gives the absolute percent errors for the solution obtained when

LINK	CAPACITY (Kbps)	Mean Service Time (msec)
1	70	29
2	50	40
3	30	66
4	40	50
5	40	50

VIRTUAL CIRCUIT	WINDOW SIZE	ROUTE (Switching Nodes)	MEAN SERVICE TIME AT THE SOURCES (msec)
1	10	1 → 2 → 3	50
2	6	1 → 2 → 4 → 5	200
3	8	1 → 2 → 3	100
4	5	2 → 4 → 5	100
5	5	4 → 5 → 3	50

Table 3.5. Parameters of Example 3.

SUBNETWORK	LOCAL CHAINS	CENTERS
1	1, 3	1, 6, 8
2	2, 4	3, 7, 9
3	5	5, 10
4		2, 4

Table 3.6. Grouping of Example 3.

MVASUB was used in comparison to MVA used for the whole network. The CPU times to solve this network in a VAX11/750 using MVASUB and MVA were 12.7 sec and 2.86 hours, respectively.

	Average			Maximum		
	L	W	λ	L	W	λ
MVASUB	0.75	0.66	.29	2.13	2.11	.84

Table 3.7. Comparison with MVA. Third Example.

Finally, the fourth network is also a model of small packet switching network taken from an example presented in [Reis79]. We transcribe the parameters for this example in Table 3.8. However, we modify center 1 and 2 so that they have two servers, instead of one in the original network. All other service centers have only one server. We used the automated clustering algorithm with the maximum cost (defined above) per iteration of a subnetwork (MCIS) specified as 100 and 180. The two partitions in subnetworks obtained are shown

Mean Service Times									
center	1	2	3	4	5	6	7	8	window size
chain 1	2	2	2	-	2	-	-	-	6
chain 2	-	0.5	0.5	0.5	-	2	-	-	8
chain 3	4	4	-	-	-	-	3	-	4
chain 4	1	-	-	1	-	-	-	5	8

All centers are processor sharing. Centers 1 and 2 have two servers, all other centers have one server.

Table 3.8. Parameters of example 4.

in Tables 3.9a and 3.9b, respectively. Note that in Table 3.9a there is an overlap between subnetworks 1 and 3. Furthermore, chain 3 belongs to the set of local chains of both subnetworks 1 and 3. Tables 3.10a and 3.10b give the absolute percent errors for the solution obtained when MVASUB was used in comparison to MVA used for the whole network. The CPU times to solve this network in a VAX11/750 using MVASUB were 35 sec for the first clustering and 42

MCIS specified	Subnetwork	Local chains	Centers	MCIS obtained
(a) 100	1	2, 3	2, 6, 7	90
	2	4	4, 8	18
	3	1, 3	1, 3, 5, 7	70
(b) 180	1	2, 4	4, 6, 8	162
	2	1, 3	1, 2, 3, 5, 7	140

Table 3.9. Clustering of example 4.

sec for the second clustering. The CPU time was 33.67 min when MVA was

MVA SUB	Average			Maximum		
	L	W	λ	L	W	λ
(a)	2.05	2.49	0.99	5.69	6.42	1.57
(b)	0.69	0.70	0.32	3.06	3.48	0.44

Table 3.10. Comparison with MVA. Fourth example.

used for the whole network. The maximum space allocated to solve this network with MVASUB and the first and second clustering and using MVA was approximately 1.92K bytes, 4.06K bytes and 356K bytes, respectively. From the results we note that the second clustering involves larger subgroups than the first one. It produces more accurate results, but is also more expensive than the first clustering (*). The first and second approximate solutions are 58 and 48 times faster than the exact one and require 185 and 88 times less storage, respectively.

(*) More expensive clustering does not necessarily produces more accurate results. It depends on how the network is clustered. However, if the more expensive clustering is formed by the union of subnetworks of a less expensive one, the more expensive cluster gives better results in general.

3.4. Conclusions.

We presented an iterative approximation technique applicable to queueing network models with a large number of closed chains. The method applies to product form networks with single server fixed rate service centers, infinite server service centers and multiple server service centers. The approach can be easily extended to support the class of queue dependent servers described in [Heff82]. Extensive empirical results indicate that this method has good accuracy/cost characteristics when compared to existing methods, particularly for networks with many chains. The approximation involves partitioning a queueing network into subnetworks. A critical part of applying the algorithm is the "clustering" of chains and service centers to form the subnetworks. An efficient and effective heuristic was described to automatically perform the clustering based on the computational cost that is specified. The approximation also provides the flexibility to trade off increased cost for increased accuracy by choosing larger subnetworks with more local chains.

The physical interpretation obtained with this technique provides a way of using it in a broader class of problems. In particular, in the next chapter, we described a simple heuristic application for a non-product form network. Furthermore, in chapter 6 we use the Poisson rates implied by equation (3.3.3) in an approximate solution for the load balancing problem with multiple chains.

CHAPTER 4

A CLUSTERING APPROXIMATION TECHNIQUE FOR NON-PRODUCT FORM QUEUEING NETWORK MODELS.

4.1. Introduction.

There are many applications which require the analysis of queueing network models which do not have product form solution. In the previous chapter we solved a LOCUS network model assuming the channel was slotted ring and used Bux's model [Bux81] for the ring, which does not violate product form requirements. Goldberg et al [Gold83] used a FCFS service center to model the channel. However, the present LOCUS network running at UCLA uses an Ethernet channel. A more accurate representation of this channel would probably violate product form requirements. Other examples include FCFS nodes with different service times for different chains in the network and/or non-exponential service times and models with simultaneous resource possession [Saue81c].

Unfortunately, the only exact method available to solve general networks with non-product form characteristics is to determine the Markov process representation of the network, if this is possible. A clever way to attack the problem can be found in [Moll81] where Petri nets are used to simplify the description of the model, and the correspondent Markov chain is automatically obtained. Even if the Markov process representation of the network can be found, the solution is usually too costly unless the network has a small state space. This is not the case for the majority of practical problems.

In chapter 2 we summarized several methods proposed to solve particular models which do not have product form solution. We overviewed Reiser's approximation [Reis79] and Marie's method [Mari79] to solve FCFS service centers with general service times, approximations to solve service centers with non-preemptive resume priority discipline [Brya83, Chan82], decomposition methods [Saue81c] and the Jacobson and Lazowska's methods [Jaco82, Jaco83] for networks with simultaneous resource possession, Heidelberger and Trivedi's method [Heid82] for solving models with the so called "split nodes", and other methods. The methods proposed in the literature for simultaneous resource possession seems to be more suited for closed networks with a single chain [Saue81c, Jaco82, Jaco83, Agra83]. The methods proposed in [Saue81c] and [Jaco83] can be used to solve multiple chain networks if jobs from different chains do not share the same passive resource (for a definition of passive resources and the associated allocate nodes the reader should consult [Lave83] or [Saue81a] or [Saue81b] or [Saue84]). Sauer [Saue81c] proposed a very expensive solution to solve networks with multiple chains where heterogeneous jobs can share the same passive resources. Another solution can be found in [Bard78], but the results are not as accurate [Saue81c].

In this chapter we use the results obtained in the previous chapter to deal with the problem of using a more accurate representation of the Ethernet channel in a LOCUS type of network. The major part of this chapter addresses the general problem of simultaneous resource possession. We propose a clustering approximation technique to solve multiple chain queueing network models with simultaneous resource possession. The method we present is able to handle more general forms of simultaneous resource possession than previous methods. It is applicable when jobs from different chains share the same passive resources.



Furthermore, customers are allowed to contend for resources while holding more than one passive resource acquired during different steps of execution. In other words, the use of passive resources can be nested. The technique requires an additional approximation for FCFS service centers with multiple servers and different service requirements for jobs belonging to different chains. Since no such approximation for this type of server in closed queueing networks has been published, we developed a new approximation to handle this problem.

4.2. A Representation of a CSMA-CD Channel in a LOCUS-type Model.

This section illustrates the application of the clustering technique developed in chapter 3 to an approximate solution for a non-product form network. Many local computer networks use a CSMA-CD protocol to access the bus connecting the different sites in the network. In the examples of the previous chapter, we assumed that the communication channel was a slotted ring and used Bux's model for the ring. Therefore the whole network model was product form. In [Gold83] a FCFS single server center was used to model the effect of the Ethernet communication channel in the LOCUS network. We now address the problem of introducing a more detailed representation of the Ethernet channel in a LOCUS type model.

In the first two examples presented in section 3.3.5, we applied the approximation developed in chapter 3 to solve a model of LOCUS with several sites. The network was divided into several subnetworks, one for each site and one representing the communication channel (subnetwork $S_{channel}$). For subnetwork $S_{channel}$ we assumed that all chains modeling customer behavior were foreign, and so the effect of customers in the rest of the network was represented as

Poisson arrivals. Furthermore, the effect of the communication channel on the different sites was represented as an infinite server delay. In this section, instead of using a FCFS center for the channel or Bux's model, we choose to solve sub-network $S_{channel}$ by using the analytical results obtained by Lam [Lam80] for a CSMA-CD protocol, with the minor heuristic modification introduced in [Bux81] for a non-slotted channel. Lam obtained a formula for the average delay for a packet in a CSMA-CD channel, assuming Poisson arrivals. Since, for sub-network $S_{channel}$, all chains are foreign, we use the arrival rates implied by equation (3.3.3) as input to Lam's formula. The mean delay obtained is used in (3.3.4).

Figures 4.1, 4.2 and 4.3 show the results obtained for a model of LOCUS with identical sites, when a FCFS center was used for modeling the channel and when Lam's average delay formula was used. The parameters for each site are:

- mean think time: 4 sec.
- mean CPU service time: 8 msec.
- average disk service time: 10 msec.
- average number of local CPU-disk cycles: 8.
- probability of making a request to a foreign site after a CPU-disk cycle: .3.
- channel speed: 1 Mbps.
- jobs from one site may choose any foreign site to run, with equal probability.

Figure 4.1 shows the average message delay in the Ethernet channel for a network with five sites when the population of each site increases from 1 to 50, the message size is 2000 bits and the maximum propagation delay (MPD) is .1 msec (thus the ratio ρ of the MPD over the mean message transmission time is .05). Also shown are 90% confidence intervals obtained from a simulation of this model. It is worthwhile to mention that the CPU time to simulate each of the four points indicated in the Figure were 1 hour, 1.24 hours, 2.23 hours and 2.51 hours, respectively using RESQ in an IBM 4341. On the other hand the

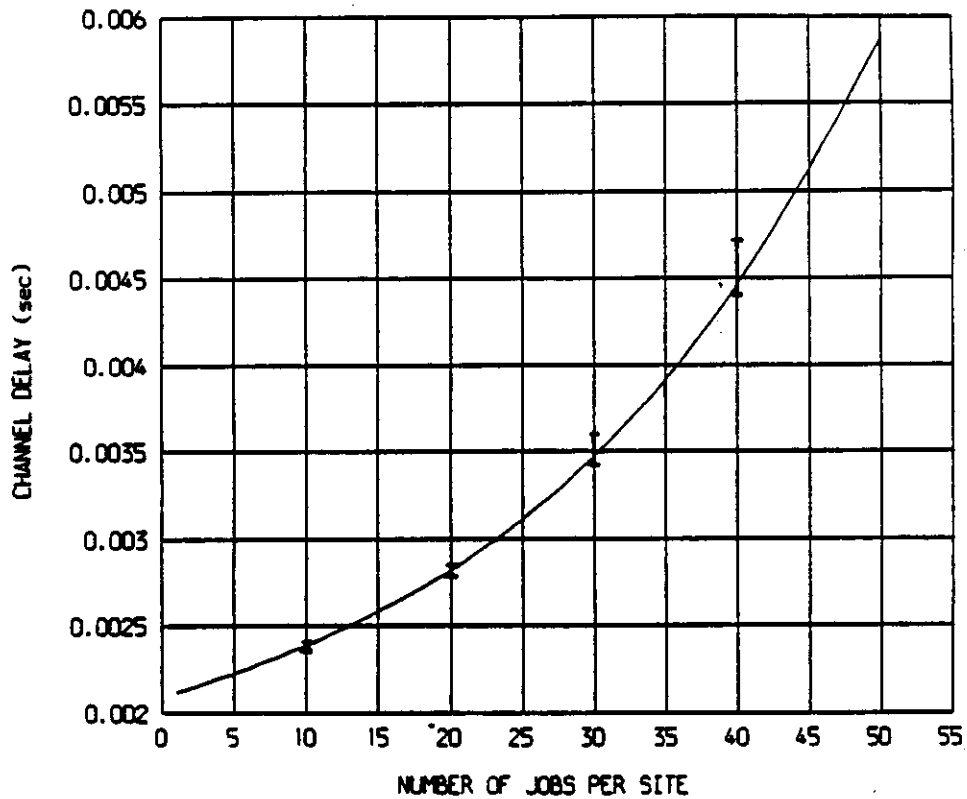


Figure 4.1. Average delay in the channel. Message = 1000 bits, MPD = .1 msec.

CPU time to analytically solve each of the fifty-one points using our approximation was between 6.1 sec and 9.2 sec in a VAX11/750. Figures 4.2 and 4.3 show the average response time when the number of sites increases, when Ethernet and FCFS models are used for the channel. We did not simulate the networks for these two Figures due to the cost of simulating such large networks. In Figure 4.2 the message size is 4000 bits and the maximum propagation delay is .1 msec ($\epsilon = .025$). In Figure 4.3 the message size is 500 bits and the maximum propagation delay is .05 msec ($\epsilon = .1$). In both cases the population of each site is 10. Observe that a FCFS service center for modeling the Ethernet channel

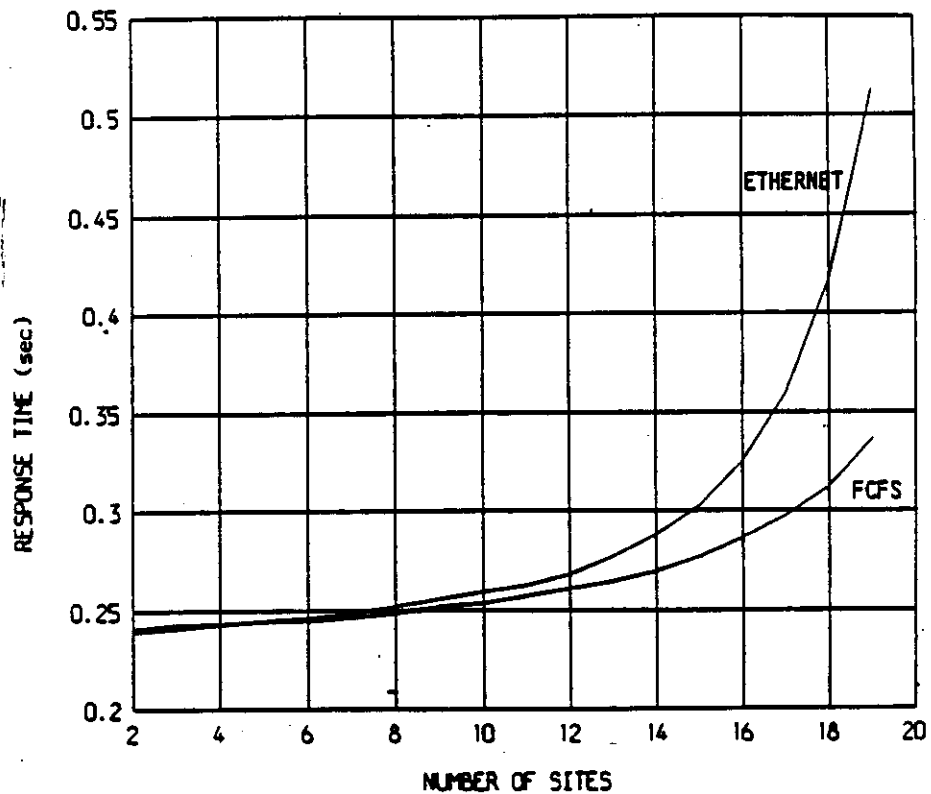


Figure 4.2. Mean response time versus number of sites.
 Message = 4000 bits, MPD = .1 msec.

provides a good approximation only when the channel is not near saturation. The sharp knee in the response time curves indicates that a threshold model [Klei76] would be a good approximation for these systems.

4.3. Simultaneous Resource Possession. Dedicated Passive Resources.

In this section we address the problem of solving a non-product form network with single or multiple chains when customers from different chains do not share the same passive resources (i.e., passive resources are dedicated to one

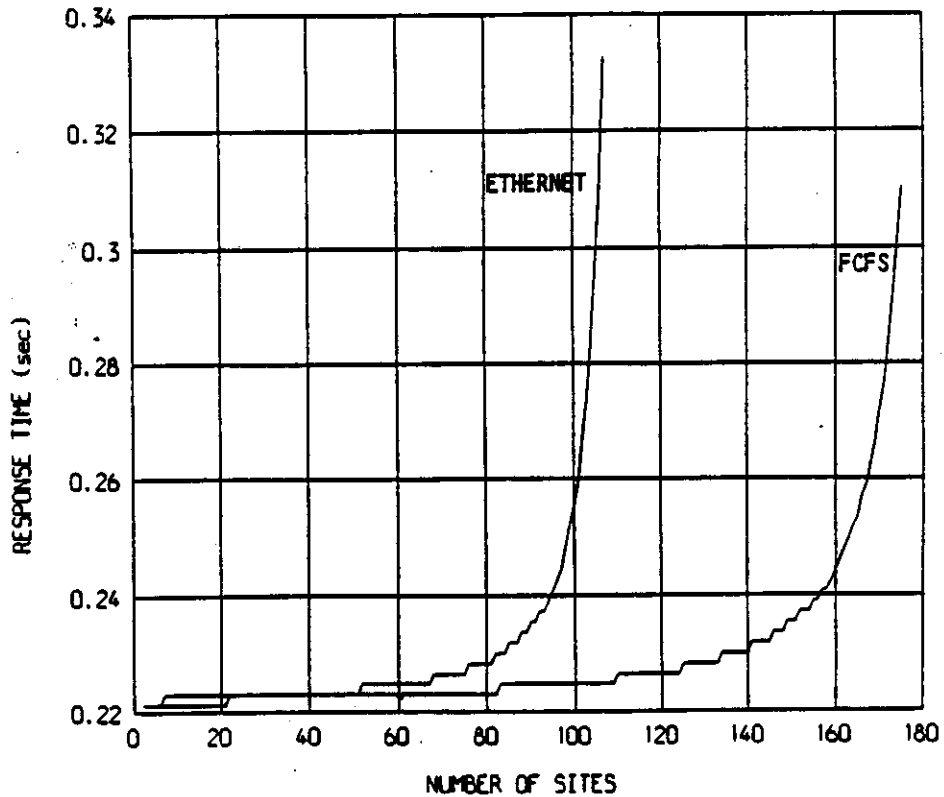


Figure 4.3. Mean response time versus number of sites.
 Message = 500 bits, MPD = .05 msec.

type of customers only). Approximation techniques for solving this problem have been proposed with good results [Saue81c, Jaco82, Jaco83]. However we will show in the following sections that the algorithm proposed here (unlike those mentioned) can be easily extended to handle shared passive resources as well.

Let us consider the example shown in Figure 2.1 transcribed to Figure 4.4 for easy reference. In this example a job has to acquire a memory partition (token), represented by the passive resource "memory", before requesting service

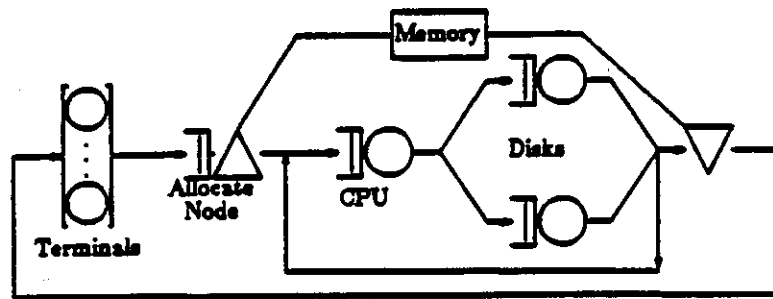
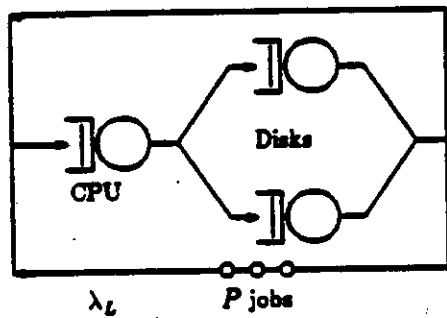
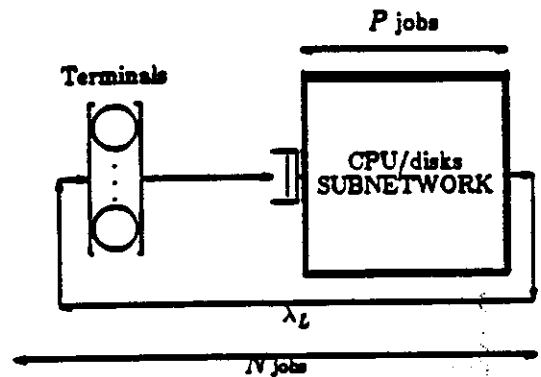


Figure 4.4. A central server model with memory constraint.

from the CPU and disks. The queue for the memory partitions is represented in the Figure by the "allocate node". The maximum number of jobs contending for CPU and disks are the number of memory partitions or tokens in the memory passive queue. Consider a saturated system where the utilization of the memory passive queue approaches one. In this limiting case, as soon as a job leaves the CPU/disks subnetwork another one is immediately allowed to enter this subnetwork since there will be always jobs waiting in the memory passive queue. Therefore, the CPU/disks subnetwork can be exactly represented by a closed network with P jobs, where P is the number of tokens in the memory passive queue. Figure 4.5a illustrates the subnetwork in this limiting case. In the saturated system, all the performance measures can be easily calculated as follows: (1) the average queue lengths, waiting times and throughputs for the CPU/disks subnetwork are calculated by solving the closed network of Figure 4.5a. (2) the limiting system throughput λ_L is also calculated from Figure 4.5a. (3) once the limiting throughput λ_L is known, the remaining performance measures can be calculated from Figure 4.5b. (For instance, the response time is simply N/λ_L - think time, where N is the total number of jobs in the network.)



(a) S_2



(b) S_1

Figure 4.5. (a) The CPU/disks subnetwork when the memory passive queue is saturated.
 (b) The entire network in the limiting case.

Now let us consider a central server model with terminals and memory constraints and with K types of jobs, each type of job represented by a single closed chain. Let us further assume that the memory is organized in P partitions and that P_1 partitions are dedicated to chain 1 jobs, P_2 partitions are dedicated to chain 2 jobs and so on so that $P_1 + P_2 + \dots + P_K = P$. As in the preceding paragraph, in the limiting case where all the passive queues are saturated, the CPU/disks subnetwork can be exactly represented by a closed queueing network with K chains, where chain i has P_i jobs. Again, the performance measures for the saturated system can be calculated by solving the closed subnetwork with K closed chains.

Returning to Figure 4.5, we note that the network of Figure 4.4 was clustered into two subnetworks: the first subnetwork (S_1) represents the terminals and the passive queue of the original network and the second subnetwork

(S_2) represents the CPU and the disks. The notion of local and foreign chains of a subnetwork introduced in the last chapter may not be directly applicable to the class of non-product form networks we are going to study, since it may not be possible to cluster the original network in order to satisfy the assumption that foreign chains contribute little to the utilization of local resources, especially in small networks. However, from the previous chapter, we note that if a chain contributes significantly to the utilization of centers in two or more different subnetworks, that chain would belong to the set of local chains of each of these subnetworks. This situation will occur in several examples we are going to study and so, we concentrate on local chains only. Due to the population constraint enforced by the passive resources, we introduce the terms **unconstrained** and **constrained**. We use the term "local unconstrained chain" (LUC) for a local chain whose customers visit the centers in the subnetwork without holding any passive resource, and "local constrained chain" (LCC), otherwise. (We generalize this notion in the next section.) Similar notation can be used for foreign chains as well. Therefore, in Figure 4.5, the only chain in the original network (chain c) constitutes the single local unconstrained chain of subnetwork S_1 and the single local constrained chain of subnetwork S_2 , i.e., $c \in LUC(S_1)$ and $c \in LCC(S_2)$. In chapter 3 we represented the complement of a subnetwork S for a local chain by an exponential infinite server center whose average delay was the response time of customers from that chain in all the centers that do not belong to S . Using the infinite server representation for the complement of S_1 for chain $c \in LUC(S_1)$, subnetwork S_1 and its complement is reduced to Figure 4.8a. It is easy to see that the network in this Figure is equivalent to the one in Figure 4.8b, where the allocate node and the infinite server center (with mean service time X_1) were collapsed into a FCFS multiple

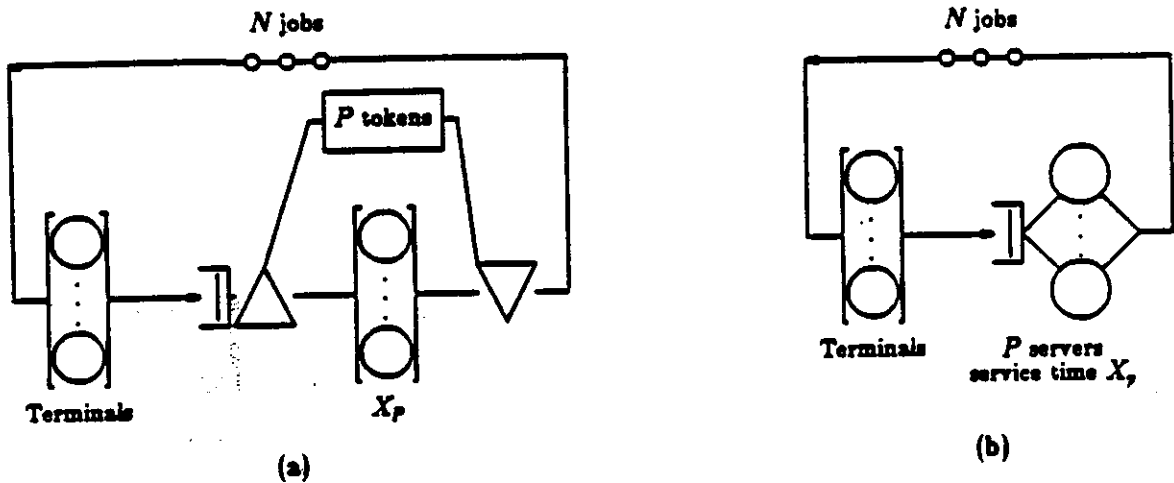


Figure 4.8. Subnetwork S_1 .

server service center with P servers (recall that P is the number of tokens in the corresponding passive queue) and mean service time X_p (*). For the complement of subnetwork S_2 we also use an infinite server center representation, as in chapter 3. However, due to the population constraint, the maximum number of jobs allowed in S_2 is the total number of tokens in the passive queue. Therefore, we represent the complement of the subnetwork for the local constrained chain $c \in LCC(S_2)$ not only by an infinite server center with average delay D but also by the number of jobs of chain $c \in LCC(S_2)$ which is equal to the number of tokens in the passive queue. Note that, using this representation for the complement of S_2 , the arrival rate of jobs into the subnetwork S_2 (given by $(1/D)(P - N_{S_2})$ where N_{S_2} is the population of subnetwork S_2) is assumed to be linear with N_{S_2} and goes

(*) We should point out that, for subnetwork S_1 , we are using a similar representation for the complement as used in [Saue81c] where the CPU/disks subnetwork and the passive resource were represented by a load dependent service center. In fact we are "linearizing" the load dependent center into a multiple server center. As we will see, this simplification also produces good results and permits the extension of the method to handle the case of shared passive resources as well.

to zero when N_{S_2} equals the maximum population P .

In order to solve subnetworks S_1 and S_2 we need to determine the value of the average service time X , and average delay D . We can define relationships between the parameters X , and D of subnetworks S_1 and S_2 , respectively, as follows:

(1) X , is equal to the response time of jobs in subnetwork S_2 . Therefore, it can be calculated by equation (4.1a) below.

$$X = \sum_{c \in LCC(S_2)} \theta_c W_c(N) \quad c \in LCC(S_2) \quad (4.1a)$$

or alternatively from Figure 4.7

$$X = \frac{P}{\lambda} - D \quad (4.1b)$$

where λ is the throughput of the network, which is equal to the throughput of subnetwork S_1 or S_2 .

(2) D is found by noting that the total population of the closed queueing network of Figure 4.7, P , is the sum of the population in the infinite server center representing the complement of subnetwork S_2 plus the number of customers in the subnetwork S_2 , N_{S_2} :

$$P = \lambda D + N_{S_2}$$

But from Figure 4.6b, N_{S_2} is the utilization U , of the multiple server center representing the complement of subnetwork S_1 . Therefore,

$$D = \frac{P - U}{\lambda}$$

or, alternatively,

$$D = X, \left(\frac{P}{U_s} - 1 \right)$$

(4.2)

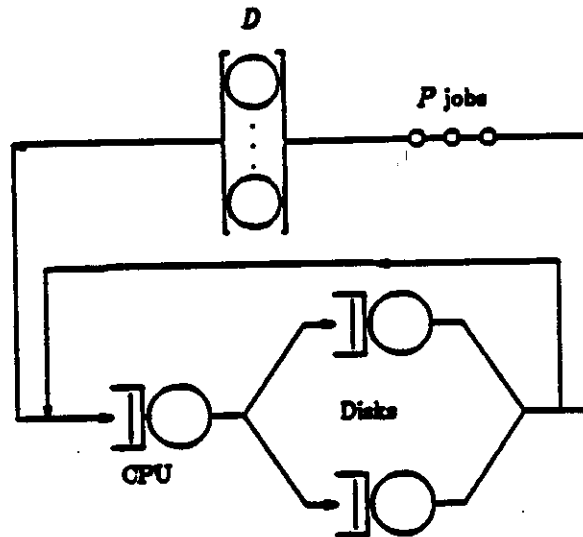


Figure 4.7. Subnetwork S_2 .

which is equation (4.1b) in a different form.

Equations (4.1b) and (4.2) have the form $Y = F(X)$ and $X = G(Y)$ and can be solved by iterating over subnetworks S_1 and S_2 . We have just outlined the steps necessary to approximately solve a non-product form network with simultaneous resource possession and dedicated resources. To conclude this section we consider the case where jobs from a single closed chain visit different passive queues as exemplified in Figure 4.8a. In this Figure, jobs from closed chain c visit allocate node 1 with probability γ and allocate node 2 with probability $1-\gamma$. The number of passive resources available are P_1 and P_2 , respectively. We label jobs visiting subnetwork S while holding passive resource P_1 as class c_{1j} jobs. In the same way, we say that jobs which hold passive resource P_2 while visiting subnetwork S belong to class c_{2j} . We use the same assumptions described above,

i.e.: (1) the waiting time of class c_1 (c_2) jobs in subnetwork S is exponential, and (2) the arrival rate of class c_1 (c_2) jobs at subnetwork S is linear with respect to class c_1 (c_2) population in S . Note that, by assumption (2), the complement of subnetwork S for chain c is represented by two closed chains, one for each class. Therefore, the original network is clustered into two subnetworks as shown in Figures 4.8b and 4.8c, and the approximation developed above can again be

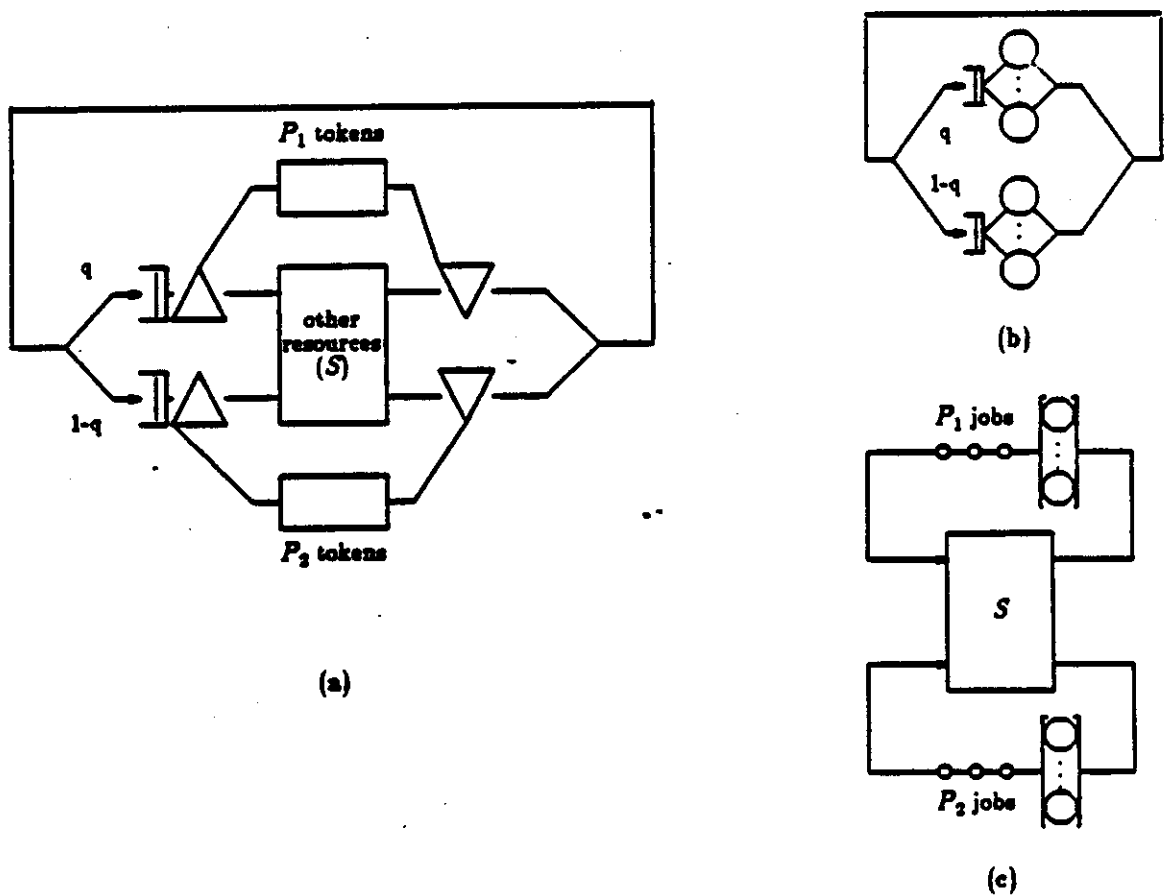


Figure 4.8. A single chain visiting 2 passive resources.

applied. Below we give the details of the general algorithm.

4.3.1. The Algorithm.

In order to describe the clustering of the original network into subnetworks, we need the following definitions:

Definition 4.1: Resource set (R-set).

Let c_p be the class of customers from chain c which visit an allocate node for passive resource p .

$$SH_c = \{ \text{resource } r \mid r \text{ is held by class } c, \text{ jobs } \}$$

Note that r can be a passive or active (*) resource.

Definition 4.2: Maximum resource set (MR-set).

The MR-sets are disjoint sets of resources of a subnetwork. They are formed in the following way:

- (1) start with the set of all R-sets of the model.
- (2) replace any pair of sets whose union is non-empty with the union of these two sets.
- (3) repeat step (2) until all set are disjoint.

Definition 4.3: Primary passive resource.

A primary passive resource is any passive resource which is not included in any MR-set. (In other words, a primary passive resource is a passive resource which is not "requested" by any job which holds another passive resource.)

The algorithm we describe next can be used to iteratively solve non-product form networks with dedicated passive resources, i.e., a passive resource

(*) "Active" resources are all resources which are not passive.

is dedicated to a single chain and a single class of customers. (Note that this implies that no class changes are allowed while a customer holds a primary passive resource. This restriction will be relaxed in the next section.) The algorithm can be applied to networks containing passive resources such that:

- (1) If a customer holds more than one passive resource at a time, these resources have to be released in the reverse order in which they were acquired, i.e., the first passive resource allocated for a customer is the last released.
- (2) Define a relation among passive resources by $P_i > P_j$, iff passive resource P_i is requested before P_j . We require that the relation be a partial ordering.

Step 1. Cluster the network with passive resources into subnetworks as follows:

- a. Form one subnetwork which contains all the resources (including passive resources) which are not in any MR-set. (Note that the only passive resources in this subnetwork will be primary passive resources.) The set of local unconstrained chains for this subnetwork is the set of all chains in the original network that visit the resources in this subnetwork.
- b. Form one subnetwork for each MR-set. The local unconstrained chains of these subnetworks will be the set of chains whose customers do not hold any primary passive resource while visiting the centers in the subnetwork. This must be true for all classes associated with this chain. All other chains will be local constrained.

Step 2.

- a. For each subnetwork formed in Step 1a or 1b, the complement of a subnetwork for a local unconstrained chain c is represented by an infinite server service center for each resource in the complement. Therefore, as

Step 6. Compare the current estimates of mean queue lengths with the previous estimates using (3.3.a1) in section 3.3.3 and terminate if this result is less than a specified threshold. Otherwise go to Step 5.

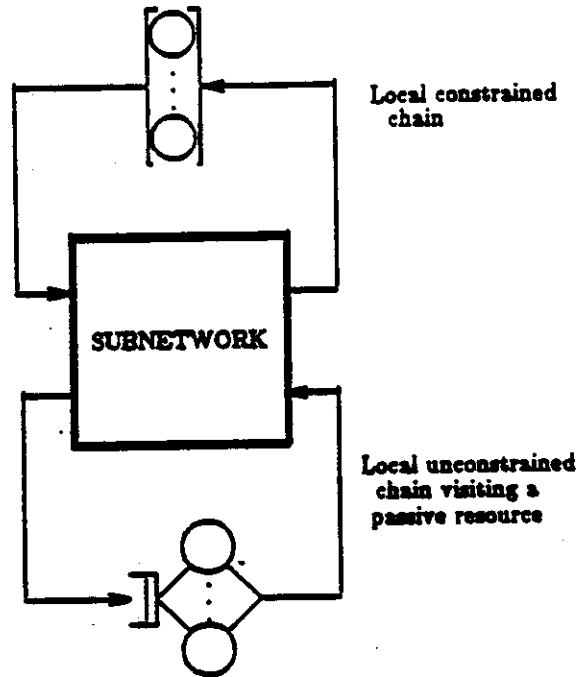


Figure 4.9. Representation of the effect of the complement for local unconstrained and constrained chains.

In the examples we solve next, we use exact MVA to recursively solve a subnetwork. Therefore, the cost per iteration is equal to the sum of the costs to solve each subnetwork, which is $O(\sum_S J_S \prod_{\substack{c \in LCC(S) \\ c \in \text{Comp}(c \in LCC(S))}} (N_c + 1)(N_c + 1) \times 2^{\#MS_S})$,

where J_S is the total number of centers in subnetwork S , N_c is the number of jobs of closed chain c , representing the complement of subnetwork S for chain $c \in LCC(S)$ when visiting the passive resource p in the complement of S , i.e., N_c is equal to the number of tokens in the passive resource p , and $\#MS_S$ is the total number of MS service centers in subnetwork S , including the ones representing

passive resources. The memory requirements of the algorithm are on the order of the memory requirements of the subnetwork which requires the largest amount of memory. Our empirical tests have shown that the convergence is very fast. In general the algorithm terminates in five iterations, when the threshold used in Step 6 is 10^{-4} .

Finally we should point out that, when the number of tokens at the passive resource is identical to the number of customers of the chain that visits the resource and each chain visits no more than one passive resource, the network will be product form and the approximation proposed in this section collapses to the approximation proposed in chapter 3, with all the chains in the network belonging to the set of local chains in all subnetworks, i.e., $c \in LC(S) \forall c, \forall i$.

4.3.2. Empirical Results.

In this section we apply our approximation technique to two examples found in the literature. The first example is a central server model with terminals, taken from [Saue81c]. There are two types of jobs represented by two closed chains and for each chain there is a separate memory constraint. The parameters for this network are:

- mean think time: chain 1 - 5 sec, chain 2 - 10 sec.
(exponential distribution, IS discipline.)
- mean CPU service time: chain 1 - 10 msec, chain 2 - 100 msec.
(exponential distribution, processor sharing discipline.)
- mean number of CPU-disk cycles: chain 1 - 10, chain 2 - 20.
(geometric distribution.)
- four identical disks with the same parameter for each chain: branching probabilities from CPU to a disk: 0.25. Mean disk service time: 35 msec.
(exponential distribution, FCFS discipline.)

Applying our approximation technique to this example, the network is

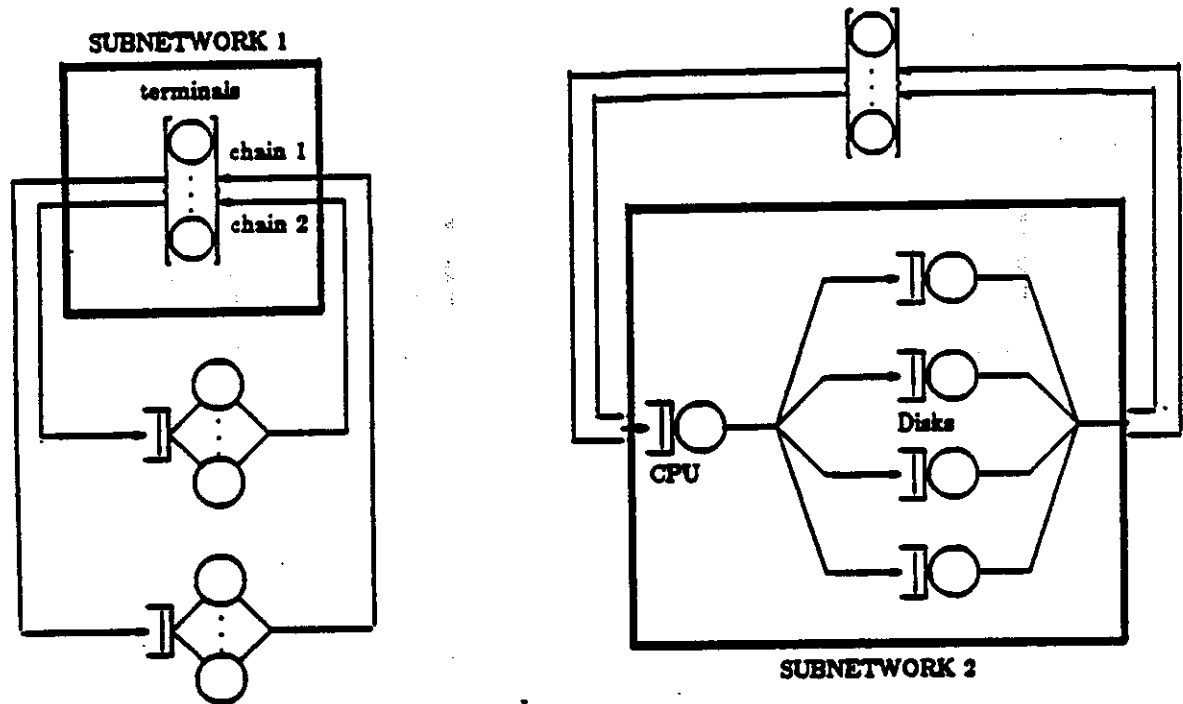


Figure 4.10. Subnetworks for the first example. Dedicated passive resources.

clustered into two subnetworks as shown in Figure 4.10. In this Figure, subnetwork 1 represents the terminals and the memory passive queues, and chains 1 and 2 are in the set of local unconstrained chains of this subnetwork. The complement of subnetwork 1 for chains 1 and 2 is represented by an infinite server center which is equivalent to two MS centers due to the memory constraints. The mean service times of these centers are given by equation (4.3). Subnetwork 2 represents the CPU and the disks. Chains 1 and 2 are in the set of local constrained chains of this subnetwork. The complement of subnetwork 2 for chain 1 and 2 is represented by: (a) limiting the number of customers in each chain to the number of memory partitions allocated for each chain, and (b) an IS service center whose mean service times for each chain is given by equation (4.4).

Sauer chose three pair of values for the total number of jobs for each chain (N_1 and N_2). For each pair, he chose three pair of values for the memory partitions P_1 and P_2 dedicated to each type of job, in order to provide low, moderate and high memory contention. He simulated the network using RESQ and 5% confidence intervals were obtained for the mean response time of all jobs at 90% confidence level. Table 4.1 presents the results obtained with our

N_1	N_2	P_1	P_2	CPU utilization		chain 1 response time		chain 2 response time	
				approx.	simul.	approx.	simul.	approx.	simul.
20	2	4	2	0.642	(0.60, 0.63)	0.717	(0.77, 0.80)	4.60	(4.49, 5.09)
		3	1	0.809	(0.59, 0.61)	0.838	(0.90, 0.95)	5.04	(4.71, 5.31)
		1	1	0.497	(0.48, 0.50)	4.83	(4.68, 4.93)	4.09	(3.86, 4.31)
30	3	7	2	0.837	(0.83, 0.85)	0.955	(1.03, 1.08)	7.2	(6.70, 7.64)
		5	1	0.808	(0.79, 0.81)	1.06	(1.13, 1.19)	9.19	(8.42, 9.89)
		2	1	0.935	(0.89, 0.71)	4.1	(3.97, 4.17)	6.49	(6.08, 6.95)
40	4	14	4	0.968	(0.96, 0.97)	1.42	(1.47, 1.54)	13.36	(12.15, 14.10)
		9	3	0.967	(0.95, 0.96)	1.58	(1.67, 1.74)	12.28	(11.98, 13.26)
		5	1	0.874	(0.87, 0.88)	2.31	(2.30, 2.42)	14.53	(13.48, 15.15)

Table 4.1. First example. Dedicated memory partitions.

approximation and using simulation. As we can observe from Table 4.1, all but one approximate value for the CPU utilization is contained within the confidence intervals. This one value falls only 0.7% outside the interval. Two thirds of the values for the response time for jobs of chain 1 fall outside the confidence intervals but the maximum relative distance of this interval is only 7.3% (for pairs (30,3) and (7,2)) when the contention for memory is low. All values for the response time of jobs of chain 2 fall inside the confidence interval. The achieved accuracy seems to be very good.

The second example is taken from [Jaco83]. This example models a computer system with passive resources. The system is shown in Figure 4.11 in a RESQ type diagram. The branching probabilities are indicated in the Figure.

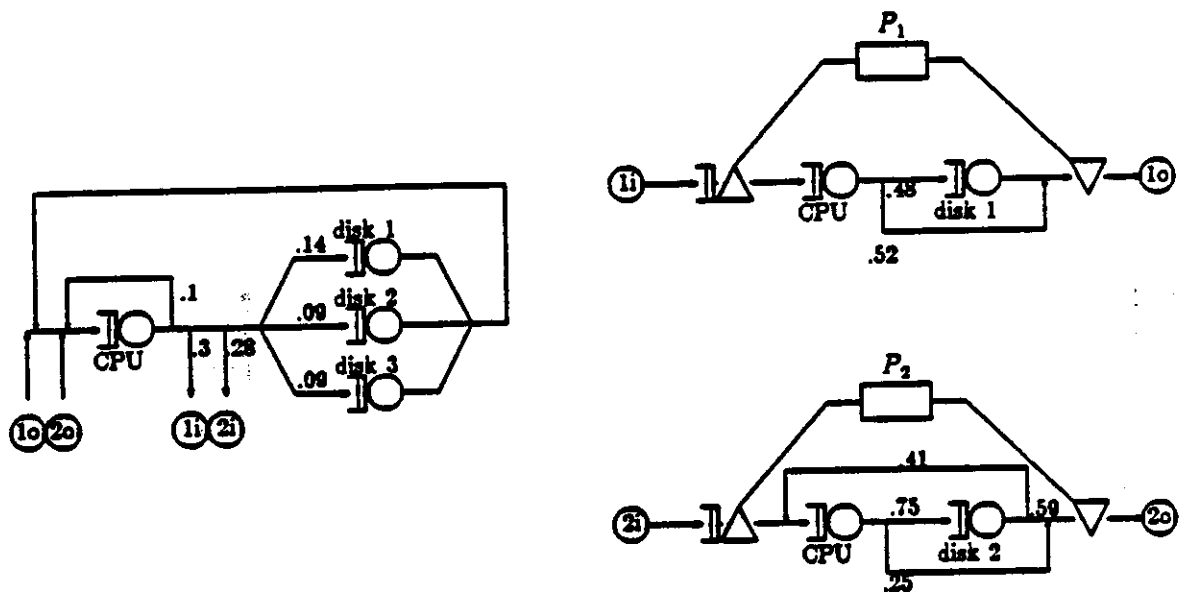


Figure 4.11. Second example (one chain network). Dedicated passive resources.

Note that, in this network, there is only one chain with three different classes. Each class models the behavior of customers when they visit the CPU and disks while holding passive resource P_1 , P_2 and without holding any passive resource, respectively. The CPU mean service time is 0.2 for all classes and the disks mean service times are 0.04 for all disks and for all classes. Applying our approximation technique to this example, the network is clustered into two subnetworks as shown in Figure 4.12. Note that subnetwork 2 (in Figure 4.12) contains three closed chains representing the complement of this subnetwork for the single local constrained chain. The visit ratios and service demands of closed chain 1, 2 and 3 correspond to the ones for the three different classes in the original network. The population of these chains is 3, 1 and 1 respectively, which correspond to the maximum number of jobs that can be present in each class (job visiting the CPU and disks without holding any passive resource, holding passive resource P_1 or holding passive resource P_2). Table 4.2 shows the visit

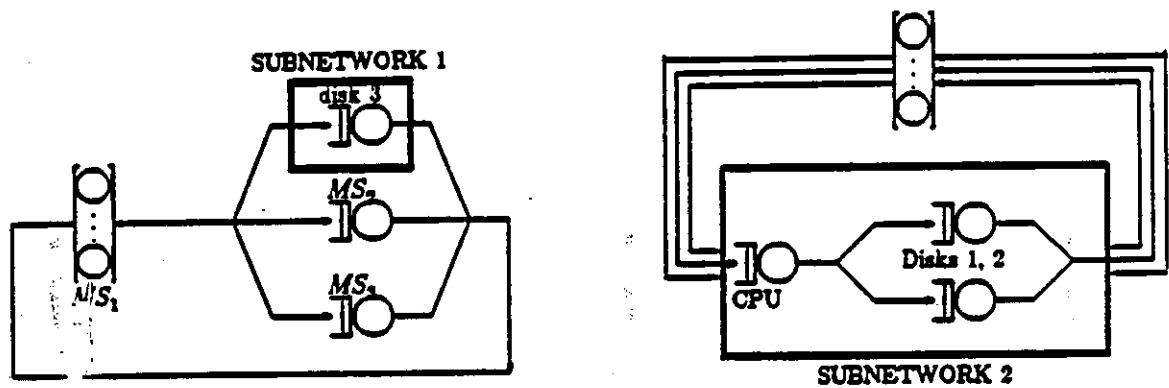


Figure 4.12. Subnetworks for the second example.

ratios for each chain at each center in the two subnetworks. Table 4.3 shows the results obtained with exact analysis, our approximation and the approxima-

	Subnetwork 1				Subnetwork 2				
	MS1	MS2	MS3	disk 3	IS	CPU	disk 1	disk 2	
1	0.4478	0.4179	0.1343		1	1.493	0.209	0.1344	chain 1
					1	1	0.48	0	chain 2
					1	1.444	0	1.083	chain 3

Table 4.2. Visit ratios. Second example.

tion proposed in [Jaco83]. As we can observe, the accuracy of our method is comparable to the accuracy of the method proposed in [Jaco83]. For low contention in the passive resources the method in [Jaco83] gives slightly better results. For moderate to high contention our method gives slightly better results. Note that in the last four results (high contention) the error is less than 1%.

We have tested our method in other networks. The results have accuracy comparable to the one obtained in the examples above.

Population	Throughput			% Error	
	Exact	Clustering	[JACO83]	Clustering	[JACO83]
3	15.88	14.93	16.52	6	4.2
4	16.68	16.1	17.1	3.5	2.8
5	17.15	16.78	17.5	2.2	2.0
6	17.42	17.19	17.7	1.3	1.5
7	17.55	17.45	17.8	0.57	1.5
8	17.62	17.6	17.8	0.11	1.1
20	17.8	17.85	18.0	0.3	1.1
50	17.8	17.79	18.0	0.06	1.1

Table 4.3. Second example. Results.

4.4. Simultaneous Resource Possession: Shared Passive Resources.

In the previous section we proposed an approximation technique for non-product form networks with simultaneous resource possession when the passive resource is dedicated to jobs of a particular type. However, the same technique can be used to approximately solve the problem of simultaneous resource possession when a passive resource is shared among jobs from different chains and/or different classes. Let us consider again the network of Figure 4.4. We assume that there is more than one type of jobs sharing the memory partitions before requesting service from the CPU and disks. Note that the maximum number of jobs allowed to contend for the CPU and disks is the number of memory partitions in the memory passive queue. Consider a saturated system where the utilization of the memory passive queue approaches one. In this limiting case, as soon as a job leaves the CPU/disks subnetwork another one is immediately allowed to enter this subnetwork. Therefore, the CPU/disks subnetwork behaves like a single chain closed network with P jobs, where P , is the number of partitions in the memory passive queue p . However, unlike the dedicated resource case, the jobs in this chain may have different behavior in the subnet-

works, since in the original network there is more than one type of jobs visiting the subnetwork. We choose to represent the behavior of different types of jobs by different classes in the subnetwork. Similar to the dedicated resource case, the network of Figure 4.4 is clustered into two subnetworks: the first one (S_1) represents the terminals and the passive queue of the original network and the second one (S_2) represents the CPU and the disks. The multiple chains in the original network belong to the set of local unconstrained chains of subnetwork S_1 . The complement of subnetwork S_1 for these chains is represented by an IS center which, combined with the allocate node, collapses to a FCFS multiple server center. The local constrained chains of subnetwork S_2 are the ones that visit the passive memory queue before visiting the CPU/disks centers, i.e., all original chains in this example. The complement of subnetwork S_2 for these local constrained chains is represented by a single closed chain (say chain f), with P_f jobs and multiple classes. Each class corresponds to a local constrained chain. As in the dedicated passive resource case, we also use an infinite server center as part of the representation of the complement of a subnetwork for a local constrained chain. The mean service times of the FCFS multiple server center and the IS center in the complement of subnetworks 1 and 2 respectively, can be obtained in the same manner as for the dedicated passive resource case. It remains to specify how to determine the probability P_i^c of a job belonging to a particular class c of closed chain f , when it enters a subnetwork (subnetwork S_2 in the example of Figure 4.4). We assume that the flows in all chains are sufficiently randomized so that P_i^c is proportional to the throughput of the corresponding local constrained chain, i.e.,

$$P_{i,p}^c = \frac{\lambda_c \theta_{i,p}^c}{\sum_i \lambda_i \theta_{i,p}^i} \quad c \in LCC(S), k \in LCC(S) \quad (4.5)$$

This assumption is similar to the one proposed by Reiser [Reis79] for approximately solving non-product networks with FCFS centers with different service times for different chains. Note that, in equation (4.5) class c of closed chain f , represents the complement of subnetwork S for a local constrained chain c which visit passive resource p .

In summary, the algorithm for the shared resource case will be identical to the algorithm of section 4.3 for the dedicated resource case, with a slight modification in Step 2 to account for the shared passive resources. Below we generalize Step 2:

Step 2.

a. For each subnetwork formed in Step 1a or 1b, the complement of a subnetwork for a local unconstrained-chain c is represented by an infinite server service center for each resource in the complement. Therefore, if a set F of l local unconstrained chains c_1, \dots, c_l visits an allocate node for a primary passive resource p , p is equivalent to a FCFS multiple server service center with as many servers as the number of tokens in the passive resource. The mean service times for each chain visiting these MS service centers are given by equation (4.6a) or (4.6b), which is identical to equation (4.3) but includes the case where different chains visit the same passive resource.

$$X_{i,p} = \sum_{j \in SH_i} \theta_{i,p}^j W_{i,p}^j \quad \theta_{i,p}^j = \theta_{c_j} \quad (4.6a)$$

where the superscript c identifies the class of closed chain f , which

represents the complement of subnetwork S for the local constrained chain c . (S is the subnetwork which contains resource j .) If subnetwork S is product form, $W_{i,j}^c$ can be obtained by solving an equivalent closed network without classes [Lave83]. If this is the case, the equation above reduces to:

$$X_{c,p} = \sum_{j \in SH_c} \frac{P_j^c \theta_{c,j} x_{c,j}}{\sum_j P_j^c \theta_{c,j} x_{c,j}} W_{i,j}^c \quad (4.8b)$$

b. For each subnetwork formed in Step 1b, the complement of subnetwork S for local constrained chains is represented by one or more closed chains. If a local constrained chain c visits allocate nodes for P distinct primary passive resources, the complement of subnetwork S for this chain is represented by P closed chains, each one (e.g., chain c_p) corresponds to local constrained chain c when customers associated to this chain hold primary passive resource p . The number of jobs in each of these chains will be equal to the number of tokens in the corresponding passive resource. If a set F of l local constrained chains c_1, \dots, c_l visits allocate nodes for the same primary passive resource p , the complement for set F is represented by only one closed chain f_p , with l classes, each one corresponding to a particular chain in F . The number of jobs in f_p will be the number of tokens of passive resource p . All chains $\{f_p\}$ or $\{c_p\}$, representing the complement of subnetwork S for one or a set of local constrained chains, visit an infinite server center whose average service time is given by equation (4.7) (the same equation holds for chains c_p):

$$D_{f_p} = X_p \left(\frac{P_p}{U_p} - 1 \right) \quad (4.7)$$

where P_p and U_p are defined in equation (4.4) and X_p is the total mean service time of passive resource p considering all chains visiting p , i.e.,

$$X_p = \sum_c \frac{\lambda_c \theta_{cp} X_{cp}}{\sum_l \lambda_l \theta_{lp}} \quad (4.8)$$

The algorithm described above and in the previous section does not include the following cases:

- (1) A passive resource p may be allocated to customers which belong to the same chain c , but are in a different class.
- (2) The class d of a customer from chain c upon releasing passive resource p is different from the class l of this same customer when he acquired p .

In general, customers from different classes may have different sojourn times in the subnetwork SH_p . We approximately model case (1) above by:

- (a) Assigning a different service time for each class at the multiple server service center representing passive resource p . Equation (4.6a) above is still valid, but a new index should be used to represent a particular class of customers.

Case (2) is modeled by:

- (b) Associating a new class l_d for each pair of classes $\langle l, d \rangle$ and assigning different service times for each class as in (a) above. Note that the routing probabilities of the new class l_d can be easily obtained from the original classes.

Unfortunately, some of the subnetworks obtained after applying Steps 1 and 2 to a network with shared passive resources will not be product form because, for these subnetworks, we will need to solve a queueing network model with FCFS multiple server service centers with different service demands for each chain (and/or each class). This is true since, as shown in Figure 4.6a, the complement of a subnetwork for a local unconstrained chain is represented by

an IS service center which, together with the passive resource, results in a multiple server center. If multiple chains (and/or classes) share the same passive resource and these chains (and/or classes) have different load requirements in the complement, the value of the mean service times of the IS center representing the complement will probably be different for each chain (and/or class), resulting in different mean service times at these multiple server centers. To our knowledge there is no published approximation technique for solving closed queueing network models with MS service centers with different service demands for jobs from different chains (and/or classes). We propose such an approximation in the next section.

4.4.1. An Approximation for Closed Queueing Network Models with FCFS Multiple Server Service Centers and Exponential Service Times.

Reiser [Reis79] proposed an approximation to handle queueing network models with single server FCFS with different service demands for jobs from different chains. Unfortunately, the results can not be easily extended for MS centers. However, we will use some of the assumptions proposed in [Reis79].

Our approximation is based on three assumptions:

1. The arrival theorem [Reis80, Lave80] is valid for this kind of non-product form networks.
2. The flows in all chains are sufficiently randomized so that, at an arrival time of a customer of chain k , the probability mass function that the servers of center j are serving n_1 customers of chain 1, ..., n_k customers of chain k given that all servers of center j are busy is multinomial, and given by equation (4.9) below:

$$PS_{bj}(\bar{n}|\bar{N}, \text{ all busy}) = \begin{cases} \frac{A_j(\bar{n}, \bar{N}-\bar{e}_j)}{C_j(\bar{n}, \bar{N}-\bar{e}_j)} & \text{if } |\bar{n}| = M_j, \bar{n} \leq (\bar{N}-\bar{e}_j) \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

where $\bar{n} = (n_1, \dots, n_k)$, $|\bar{n}| = n_1 + n_2 + \dots + n_k$ and the notation $\bar{n} \leq (\bar{N}-\bar{e}_j)$ means that the operation " \leq " is applied element by element of vectors \bar{n} and $(\bar{N}-\bar{e}_j)$.

$$A_j(\bar{n}, \bar{N}-\bar{e}_j) = \frac{M_j!}{n_1! \dots n_k!} \prod_{i=1}^k F_{bj}^{n_i}(\bar{N}-\bar{e}_j) \quad (4.10)$$

$$F_{bj}(\bar{N}-\bar{e}_j) = \frac{\lambda_j(\bar{N}-\bar{e}_j) a_{bj}}{\sum_{c=1}^K \lambda_c(\bar{N}-\bar{e}_j) a_{cj}} \quad (4.11)$$

$C_j(\bar{N}-\bar{e}_j)$ is a normalization constant given by:

$$C_j(\bar{N}-\bar{e}_j) = \sum_{\substack{|\bar{n}| = M_j \\ \bar{n} \leq (\bar{N}-\bar{e}_j)}} A_j(\bar{n}, \bar{N}-\bar{e}_j) \quad (4.12)$$

From this second assumption, the probability mass function that, at an arrival time, the servers of center j are serving n_i customers of chain i , $i = 1, \dots, K$ given that all servers of center j are busy and at least one of the servers of center j is busy with a customer of chain i is given by:

$$PS_{bj}(\bar{n}|\bar{N}, \text{ all busy, } n_i \geq 1) = \begin{cases} \frac{A_j(\bar{n}, \bar{N}-\bar{e}_j)}{C_{ij}(\bar{n}, \bar{N}-\bar{e}_j)} & \text{if } |\bar{n}| = M_j, \bar{n} \geq \bar{e}_i \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

where

$$C_{ij}(\bar{N}-\bar{e}_j) = \sum_{\substack{|\bar{n}| = M_j \\ \bar{n} \leq (\bar{N}-\bar{e}_j) \\ n_i \geq 1}} A_j(\bar{n}, \bar{N}-\bar{e}_j) \quad (4.14)$$

3. At the time when customer i starts service at center j , a customer k who

arrived at center j while customer l was in queue sees the customers in service at j distributed with probability mass function $PS_{lj}(\bar{n}|\bar{N}, \text{all busy}, n_i \geq 1)$ given above.

From the assumptions above we can find an equation analogous to equation (2.5b). For that, we note that the mean waiting time observed by a chain k customer at center j has three components, namely:

1. The mean waiting time of the arriving customer, w_{lj} .
2. If all servers are busy when chain k customer arrives, the average elapsed time until the first customer leaves, $w_{lj}(\bar{N})$.

Given that all servers are busy and there are n_i customers of chain i , $i = 1, \dots, K$, in service when chain k customer arrives, by the memoryless property of the exponential distribution, the elapsed time from the arrival of this customer until the next departure is simply the minimum of the service times of customers in service. For exponentially distributed service times, this minimum has mean

$$(n_1\mu_1 + n_2\mu_2 + \dots + n_K\mu_K)^{-1} \quad (4.15)$$

Therefore, unconditioning and using assumption 2:

$$w_{lj}(\bar{N}) = \sum_{\substack{\bar{n} = M_j \\ \bar{n} \leq (\bar{N} - r_j)}} PS_{lj}(\bar{n}|\bar{N}, \text{all busy})(n_1\mu_1 + n_2\mu_2 + \dots + n_K\mu_K)^{-1} \quad (4.16)$$

3. The mean "effective backlog" of customers in queue when a chain k customer arrives. This "effective backlog" can be calculated by noting that, whenever a customer l which is in queue in front of customer k enters in service, the elapsed time until the next departure (call the mean of this

elapsed time $se_{kj}(\bar{N})$) is simply the minimum of the service times of customers in service. Conditioning on the number of customers from each chain at the servers and using assumption 3, we get:

$$se_{kj}(\bar{N}) = \sum_{\substack{\bar{n} = M_j \\ (\bar{N} - c_j)}} PS_{kj}(\bar{n} | \bar{N}, \text{all busy}, n_i \geq 1)(n_1\mu_1 + n_2\mu_2 + \dots + n_K\mu_K)^{-1} \quad (4.17)$$

Therefore, the "effective backlog" for customer k at center j is:

$$\sum_{i=1}^K se_{kj}(\bar{N}) q_{ij}(\bar{N} - c_i) \quad (4.18)$$

Combining components 1, 2, and 3 we finally find an expression for the mean waiting time of a chain k customer at a FCFS center with exponential service times, when the population of the network is \bar{N} :

$$W_{kj} = s_{kj} + \sum_{i=1}^K se_{kj}(\bar{N}) q_{ij}(\bar{N} - c_i) + PB_j(\bar{N} - c_i) sr_{kj}(\bar{N}) \quad (4.19)$$

It remains to find an expression for $PB_j(\bar{N})$ which is the probability that all servers of service center j are busy, when center j is a FCFS center. We approximate $PB_j(\bar{N})$ by the expression used for this probability considering center j as a processor sharing center. Therefore, $PB_j(\bar{N})$ can be calculated by equation (2.4c) and (2.4a). However, equation (2.4b) for $P_j(0|\bar{N})$, the probability of zero customers at service center j , is no longer valid. The original approach to calculate $P_j(0|\bar{N})$ [Reis80],

$$P_j(0|\bar{N}) = 1 - \sum_{i=1}^{\bar{N}} P_j(i|\bar{N})$$

is known to fail numerically as $P_j(0|\bar{N})$ tends to zero ([Lave83, Chan80]), since the sum $\sum P_j(i|\bar{N})$ may result in values slightly greater than one. Therefore, we

choose to use, as an approximation, the following equation:

$$P_j(0|\bar{N}) = \max(0, 1 - \sum_{i=1}^{|\bar{N}|} P_j(i|\bar{N})) \quad (4.20)$$

($\max(A,B) \triangleq A$ if $A \geq B$ or B otherwise.)

which results in a numerical stable algorithm (*).

From equations (4.9) through (4.20) we can observe some interesting properties:

- (1) if all mean service times at service center j are identical, equation (4.19) reduces to equation (2.3d) which is the exact equation for MS service centers in a product form network.
- (2) if the number of servers at service center j is equal to one, equation (4.19) reduces to the approximate equation obtained by Reiser [Reis79] for a FCFS center with different exponential service times for different chains.
- (3) for certain particular cases, the equations obtained above converge to the exact result. For instance, consider a very simple network which has only one FCFS center with two servers and two chains. Chain 1 has two customers and chain 2 has only one customer. Assume the mean service time

(*) Although the use of equation (4.20) results in a numerical stable algorithm, its use in a product form network may give values which present small errors when $P_j(0|\bar{N})$ approaches zero. For instance, utilizations slightly over one may result in the final answer. However, although the use of equation (4.20) may be unacceptable for exact MVA, our empirical tests indicate that the results obtained are very reasonable as an approximation and the small errors obtained do not significantly affect the accuracy of the overall approximation for FCFS centers. As a consequence of these errors, the ratio P_j/U_j in equation (4.7) may be slightly over one for particular cases. This would result in a negative value for D_j . To avoid this problem the ratio P_j/U_j is truncated to one whenever its value is greater than one.

of customers from chain 1 equal to 1. If we let the mean service time of customers from chain 2 grows to infinity, the solution of the network for chain 1 customers converges to the solution of the same network with only chain 1 customers and the FCFS center with only one server. This is the exact solution since, as the mean service time of chain 2 customers grows to infinity, once this customer acquires one of the servers he never releases it, leaving only one server for chain 2 customers. The utilizations obtained in the limit are 1.0 for both chains. Note that using a processor sharing center instead of a FCFS center the utilizations would be 1.333 and 0.6667, respectively.

4.4.1.1. Empirical Results for FCFS Multiple Server Centers.

We have done several tests to validate our approximation for multiple server service centers. In this section we present only a subset of the tests which seem to stress our approximation. We choose to present the results obtained for two chain, two center networks, similar to those in [Brya83] (*). One of the centers in the network (center number 2) is a multiple server FCFS service center. The other (center number 1) is an IS center or a processor sharing center. For the first set of experiments, center number 1 was chosen to be an IS center and the following parameters were used for the network:

- mean service times: $s_{11} = 6$, $s_{21} = 3$, $s_{12} = 1$, $s_{22} = \text{variable}$.
- number of customers: $N_1 = 5$, $N_2 = 2$.
- number of servers of the FCFS center: $M_2 = 2$.

We choose a small customer population for one of the chains to stress assumption 2. The mean service time of customers from chain 2 at the FCFS multiple server center was varied from 2 to 20, so that the total average utilization of this

(*) For a discussion of the choice of this kind of networks refer to [Brya83].

FCFS center varied from moderate to high and the percentage of the utilization used by the chain with small number of customers also increased. We simulated the network using RESQ and the regenerative method. Approximately 5% confidence intervals were obtained for the mean waiting time of customers at the FCFS center at a 90% confidence level. Table 4.4a presents the results obtained for the FCFS center using our approximation and using simulation. Table 4.4b presents the results for the same network, but with the number of customers of

T_{22}	Chain 1			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.33	(0.322, 0.336)	1.58	(1.51, 1.59)
5	0.28	(0.281, 0.289)	2.93	(2.69, 2.80)
8	0.239	(0.246, 0.252)	4.48	(3.98, 4.1)
11	0.207	(0.221, 0.226)	6.06	(5.11, 5.3)
14	0.183	(0.198, 0.201)	7.65	(6.42, 6.58)
17	0.164	(0.179, 0.182)	9.23	(7.75, 7.93)
20	0.149	(0.163, 0.165)	10.81	(9.06, 9.25)
T_{22}	Chain 2			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.377	(0.368, 0.386)	2.30	(2.30, 2.41)
5	0.587	(0.571, 0.582)	5.51	(5.67, 5.87)
8	0.689	(0.663, 0.671)	8.61	(9.0, 9.27)
11	0.749	(0.714, 0.72)	11.68	(12.0, 12.3)
14	0.79	(0.755, 0.76)	14.72	(15.2, 15.6)
17	0.82	(0.783, 0.788)	17.75	(18.43, 18.79)
20	0.841	(0.807, 0.81)	20.77	(21.56, 21.97)

Table 4.4a. Results for FCFS multiple server center.
First set of experiments ($N_2 = 2$).

chain 2, N_2 , decreased to 1 (*).

(*) In this case as x_{22} increases, the probability that the customer from chain 2 will be at one of the servers from center 2 also increases. The other server will be free for customers from chain 1.

T_{22}	Chain 1			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.346	(0.341, 0.358)	1.22	(1.2, 1.27)
5	0.339	(0.332, 0.344)	1.37	(1.35, 1.41)
8	0.336	(0.328, 0.338)	1.46	(1.44, 1.51)
11	0.334	(0.328, 0.336)	1.5	(1.51, 1.57)
14	0.332	(0.325, 0.332)	1.53	(1.56, 1.62)
17	0.331	(0.33, 0.337)	1.55	(1.64, 1.69)
20	0.331	(0.326, 0.332)	1.56	(1.65, 1.71)
T_{22}	Chain 2			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.196	(0.184, 0.201)	2.1	(1.98, 2.13)
5	0.308	(0.302, 0.319)	5.1	(5.0, 5.39)
8	0.36	(0.349, 0.364)	8.1	(7.95, 8.55)
11	0.39	(0.383, 0.395)	11.1	(10.7, 11.5)
14	0.409	(0.401, 0.412)	14.1	(13.32, 14.3)
17	0.423	(0.414, 0.424)	17.1	(16.54, 17.8)
20	0.433	(0.43, 0.438)	20.1	(19.5, 20.9)

Table 4.4b. Results for FCFS multiple server center.
First set of experiments ($N_2 = 1$).

For the second set of experiments, center number 1 was chosen to be a processor sharing center and the following parameters were used for the network:

- mean service times: $\mu_{11} = 1$, $\mu_{21} = 2$, $\mu_{12} = 1$, $\mu_{22} = \text{variable}$.
- number of customers: $N_1 = 5$, $N_2 = 2$.
- number of servers of the FCFS center: $M_2 = 2$.

Again, the mean service time of customers from chain 2 at the FCFS multiple server center was varied from 2 to 20. We simulate this network obtaining confidence intervals similar to the previous set of experiments at 90% confidence level. Table 4.5a presents the results obtained using our approximation and using simulation. Table 4.4b presents the results for the same network, but with $N_2 = 2$.

T_{22}	Chain 1			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.352	(0.341, 0.36)	1.39	(1.34, 1.44)
5	0.353	(0.349, 0.361)	2.42	(2.18, 2.38)
8	0.331	(0.327, 0.337)	3.79	(3.02, 3.28)
11	0.299	(0.294, 0.304)	5.25	(4.0, 4.3)
14	0.266	(0.267, 0.277)	6.73	(5.08, 5.43)
17	0.236	(0.247, 0.255)	8.22	(5.97, 6.34)
20	0.21	(0.228, 0.236)	9.73	(6.93, 7.31)
T_{22}	Chain 2			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.145	(0.142, 0.169)	2.34	(2.35, 2.57)
5	0.307	(0.30, 0.344)	5.60	(5.77, 6.24)
8	0.423	(0.418, 0.459)	8.8	(8.94, 8.53)
11	0.507	(0.509, 0.547)	11.95	(12.5, 13.2)
14	0.572	(0.58, 0.614)	15.06	(15.72, 16.65)
17	0.621	(0.622, 0.652)	18.14	(18.8, 19.66)
20	0.661	(0.669, 0.693)	21.21	(21.52, 22.4)

Table 4.5a. Results for FCFS multiple server center.
Second set of experiments ($N_2 = 2$).

From the results shown in Tables 4.4 and 4.5 we can observe that all the utilizations obtained with the approximation are in close agreement with the results obtained with simulation. That is also true for the waiting time of chain 2 customers and the majority of results for chain 1 customers. However, we observe that the approximation overestimates the values of the mean waiting time of chain 1 customers when the utilization of chain 2 customers at the FCFS is high and when $N_2 = 2$. Note that, in this case, a small error in the estimates of the probability distribution of chain 2 customers at the servers of the FCFS center can cause large errors in the mean waiting time of chain 1 customers since chain 2 customers have much higher service demands than chain 1 customers and the chain 2 population is identical to the number of servers at the FCFS center. Furthermore, the low number of customers should stress assumption 2.

T_{22}	Chain 1			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.41	(0.399, 0.419)	1.31	(1.26, 1.35)
5	0.419	(0.412, 0.429)	1.57	(1.55, 1.65)
8	0.424	(0.412, 0.426)	1.74	(1.73, 1.84)
11	0.427	(0.42, 0.43)	1.87	(2.0, 2.12)
14	0.429	(0.413, 0.423)	1.95	(2.09, 2.21)
17	0.431	(0.414, 0.424)	2.02	(2.1, 2.21)
20	0.432	(0.413, 0.423)	2.08	(2.2, 2.31)
T_{22}	Chain 2			
	utilization		mean waiting time	
	approx.	simul.	approx.	simul.
2	0.0844	(0.0742, 0.0983)	2.28	(2.19, 2.43)
5	0.168	(0.145, 0.182)	5.28	(4.87, 5.52)
8	0.224	(0.203, 0.243)	8.28	(7.79, 8.65)
11	0.264	(0.253, 0.293)	11.28	(11.1, 12.34)
14	0.294	(0.297, 0.333)	14.28	(14.52, 16.17)
17	0.317	(0.293, 0.333)	17.28	(15.76, 17.64)
20	0.335	(0.312, 0.35)	20.28	(18.69, 20.77)

Table 4.5b. Results for FCFS multiple server center.
First set of experiments ($N_2 = 1$).

4.4.2. Shared Passive Resources. Empirical Results.

In this section we apply the algorithm of sections 4.3.1 and 4.4 to networks which possess passive resources shared among customers from different chains. We use the approximation developed in section 4.4.1 whenever any of the subnetworks obtained contains FCFS service centers with multiple servers. The first example is a central server model with terminals taken from [Saue81c]. Like the first example of section 4.3.2, there are two types of jobs represented by two closed chains. However, in this case, both jobs share the same memory partitions. The parameters for this example are identical to the ones for the first example in section 4.3.2.

Applying our approximation technique to this example, the network is

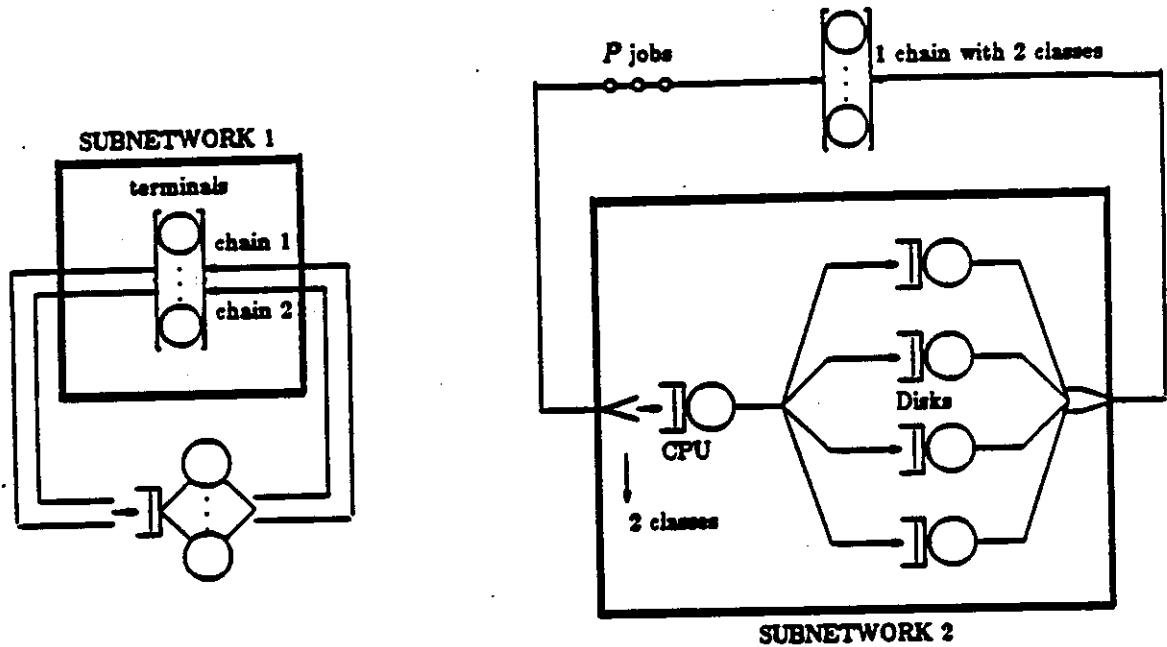


Figure 4.13. Subnetwork for the first example. Shared passive resources.

clustered in two subnetworks as shown in Figure 4.13. In this Figure, subnetwork 1 represents the terminals and the memory passive queue, and chain 1 and chain 2 are in the set of local unconstrained chains of this subnetwork. The complement of chain 1 and 2 is represented by an IS service center which collapses to one FCFS multiple server center due to the memory constraint. The mean service time of customers from each chain at this center is give by equation (4.8b). Subnetwork 1 is not product form since there are two chains visiting a FCFS multiple server center with different service demands. Therefore, this subnetwork will have to be solved using the approximation developed in section 4.4.1. Subnetwork 2 represents the CPU and the disks. Chain 1 and 2 are in the set of local constrained chains of this subnetwork. The complement of both chains is represented by: (a) a single chain f , with P jobs where P is the

number of memory partitions. This chain has two classes. Each one corresponds to one of the local constrained chains. The probability of a job of chain f , belonging to one of the classes is given by equation (4.5). (b) an IS center whose mean service time is given by equation (4.7).

Similar to the example with dedicated resources, Sauer chose three pairs of values for the total number of jobs for each chain. For each pair, three values for the memory partitions were chosen in order to provide low, moderate and high memory contention. Sauer simulated the network using RESQ and 5% confidence intervals were obtained for the mean response time of all jobs at 90% confidence level. Table 4.6a presents the results obtained with our approxima-

N_1	N_2	P	CPU utilization		chain 1 response time		chain 2 response time	
			approx.	simul.	approx.	simul.	approx.	simul.
20	2	6	0.623	(0.60, 0.63)	0.683	(0.73, 0.76)	4.77	(4.34, 4.93)
		4	0.621	(0.60, 0.62)	0.804	(0.91, 0.95)	4.49	(4.46, 4.88)
		2	0.527	(0.53, 0.54)	2.75	(2.35, 2.47)	4.88	(4.85, 5.12)
30	3	9	0.844	(0.82, 0.84)	0.958	(1.00, 1.04)	7.65	(6.95, 7.92)
		6	0.832	(0.83, 0.85)	1.28	(1.20, 1.26)	6.97	(6.87, 7.47)
		3	0.681	(0.69, 0.71)	4.15	(3.44, 3.62)	6.99	(6.69, 7.01)
40	4	18	0.952	(0.95, 0.96)	1.42	(1.47, 1.54)	14.3	(12.41, 14.0)
		12	0.956	(0.95, 0.96)	1.54	(1.64, 1.71)	13.24	(11.63, 12.74)
		6	0.885	(0.90, 0.90)	3.45	(3.19, 3.35)	9.43	(9.23, 9.81)

Table 4.6a. First example. Shared memory partitions.

tion and using simulation. As we can observe from Table 4.6a, almost all the approximate values for the CPU utilizations and response times for the second chain are within the confidence intervals. The worst error falls only 1.7% and 3.9% outside the confidence intervals for CPU utilization and response time of chain 2, respectively. Although all the approximate values for the response time of chain 1 fall outside the confidence intervals, these values are very close to the simulation results and the worst error is only 14.6%. Table 4.6b shows the CPU

times in seconds spent on the simulation on an IBM 370/168, as reported in [Saue81c]. For each of the cases, the CPU times for the approximate solution

N_1	N_2	P	approximation (VAX11/750)	simulation (IBM370/168)
20	2	6	14	379
		4	14	733
		2	11	977
30	3	9	40	422
		6	37	876
		3	20	1,394
40	4	18	82	669
		12	96	1,185
		6	37	1,343

Table 4.6b. CPU times (sec).

using a VAX11/750 is also shown (*). For the approximate solution we use 10^{-4} as the value for the threshold in Step 6 of the algorithm.

For the second example we chose to model a system where, in addition to memory contention, there is also contention for peripheral processors (I/OP). A job must have an I/OP continuously while doing I/O. Assume that the operating systems reserve a pool of I/OP for user I/O commands. The I/OP are identical. For this example we use the parameters of the first example of section 4.3.2, seventh case. Therefore, there are 40 jobs of type 1 and 4 jobs of type 2. Each type has a separate memory constraint ($P_1 = 14$ and $P_2 = 4$ in this example). Furthermore, a job has to acquire any of the I/O processors before contending for the disks. We assume that there are four I/O processors available. Figure 4.14 shows the three subnetworks obtained after applying our approxima-

(*) For comparison we point out that in our VAX11/750 a two chain two single server center network, with 40 and 4 jobs respectively, takes approximately 6 sec to be solved using MVA.

tion technique to this example. In this Figure, subnetwork 1 represents the terminal and the two memory passive queues. Chain 1 and chain 2 are in the set of local unconstrained chains for this subnetwork. The complement of chains 1 and 2 and the passive queues collapse to two FCFS multiple server centers. Subnetwork 2 represents the CPU and the I/OP passive queue. Chains 1 and 2 are in the set of local constrained chains as well in the set of local unconstrained chains for this subnetwork. Therefore, the complement of chains 1 and 2 is represented by: (a) the number of jobs in two closed chains which is equal to the memory partitions for chain 1 and 2, respectively; (b) an IS venter whose mean service time for each chain is given by equation (4.7) and (c) an IS service center which collapses to one FCFS multiple server center due to the I/OP constraint. The mean service times of each chain in this center are given by equation (4.6b). Subnetwork 3 represents the disks only. Chains 1 and 2 are in the set of local constrained chains of this subnetwork. The complement of both chains are represented by: (a) a single chain with as many jobs as the number of I/O processors. This chain has two classes. Each class corresponds to one of the local constrained chains. (b) an IS service center whose mean service time is given by equation (4.7).

Table 4.7 present the results when the approximation solution was used. The network was simulated using RESQ and the regenerative method. 5% confidence interval were obtained for the mean response time of all jobs at 90% confidence level.

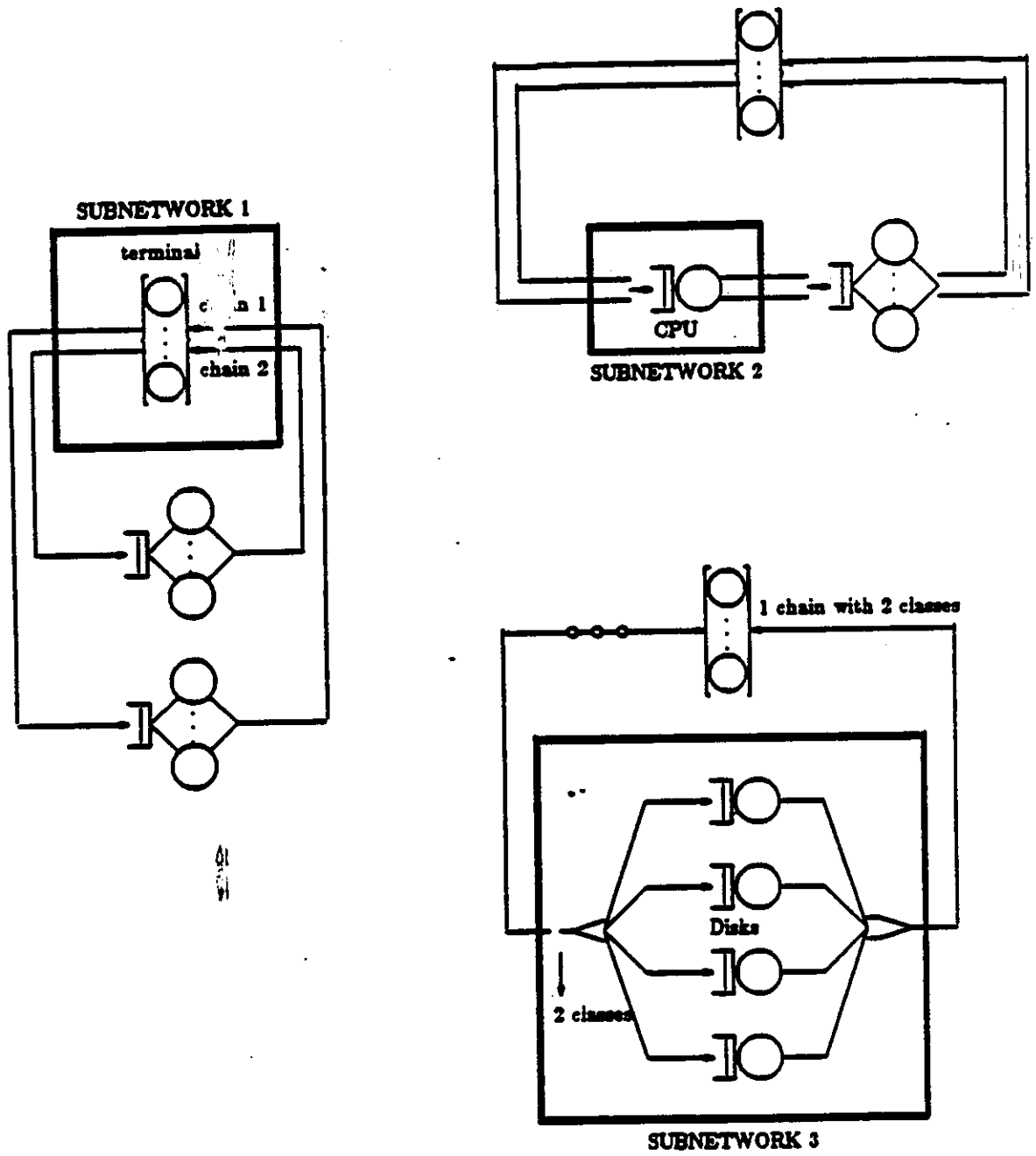


Figure 4.14. Subnetworks for the second example. Dedicated and shared passive resources.

CPU utilization		chan 1 response time		chain 2 response time	
approx.	simul.	approx.	simul.	approx.	simul.
0.939	(0.918, 0.930)	1.74	(1.91, 1.96)	13.14	(12.89, 13.96)

Table 4.7. Second example. Shared I/O processors.

4.5. Conclusions.

We applied the algorithm developed in previous chapters to approximately solve a non-product form LOCUS type of model with a detailed representation of an Ethernet communication channel. We presented examples showing the influence of the Ethernet channel on the response time of customers in the network and compared the results with those obtained when a FCFS service center was used to model the channel. The same approach used to solve this LOCUS type of model could be applied to other problems with similar characteristics. For instance, we could apply the same algorithm to approximately solve a communication network with window flow control where each channel uses a CSMA-CD protocol. (Other type of channels, e.g., a slotted ring, may also be present in the network.) However, empirical studies must be done to validate the approach in these cases.

The main part of this chapter was devoted to develop an approximation technique for non-product form queueing network models with simultaneous resource possession. We developed an accurate approximation algorithm to solve networks where the passive resource is dedicated to each type of job. The algorithm is based on the clustering technique of the previous chapter. Empirical results have shown that the algorithm has accuracy comparable to other approaches proposed in the literature. However, as pointed out by Sauer

[Saue81c], the cases where heterogeneous jobs share both simultaneous held resources seems to be more important than the dedicated resource case, but is also more difficult to solve. This is confirmed by the lack of cost efficient and accurate algorithms in the literature. We have extended the algorithm developed for the dedicated resource case to handle networks with passive queues shared by different job types. In the extension, non-product form networks with FCFS multiple server service center with different service times for different type of jobs need to be solved. Unfortunately, to our knowledge, there is no published approximation technique to handle these networks. We proposed an accurate approximation to solve this problem. Empirical studies have shown that the overall algorithm for solving non-product form networks with shared passive resources is very efficient and gives accurate results.

We have concentrated our studies on the case of subnetworks with local chains only. However, it is possible to use the notion of foreign chains if the assumption that they contribute little to the utilization of the resources of a subnetwork can be satisfied. In this case, the Poisson arrival representation for the complement would be used. Although we haven't presented any result for these cases we have done preliminary tests with good results.

Our approximation technique can be easily extended to solve mixed queueing networks where the open chain customers also share a passive resource with closed chain customers. This is true since an open chain with rate λ may be considered the limiting case of a closed chain with a bottleneck node with mean service time $1/\lambda$. The modifications necessary include: (a) substituting \bar{e}_i in equations (4.9), (4.13) and (4.19) by $\bar{0}$ if k is an open chain. This is motivated by the arrival theorem for open chain networks [Lave83]. (b) $PB_i(\bar{N})$ is still approxi-

mated by considering center j a processor sharing center and so degraded service rates from equation (3.3.12a) should be used to account for the open chains.

We may also extend the above technique to handle cases where the scheduling discipline at the passive resources is non-preemptive priority instead of FCFS. For this case equation (4.19) would be modified as in to the approach proposed in [Brya83]. Needless to say, validation experiments must be done to evaluate the accuracy of the above extensions.

CHAPTER 5

LOAD BALANCING IN DISTRIBUTED SYSTEMS.

5.1. Introduction.

The principal motivation for the research done in the preceding chapters was to find an accurate and cost effective approximate solution technique to solve for large queueing network models of the LOCUS distributed computer system, since exact solution techniques are by far too costly to handle problems of this size. We developed such technique for product form networks and extend the results for non-product form networks. Our approach, albeit general so that it can be applied to any queueing networks model, is particular suited to solve large LOCUS type of models. In this chapter we continue to focus our research on issues related to distributed computer systems. In particular we address the load balancing problem in distributed systems. In these systems a number of resources (CPU's, file servers, disks, etc) are shared among jobs originating at different computer sites. Under the LOCUS operating system, for example, processes generated by users at one site in the network are allowed to run on other sites. In general, in a distributed system environment it is desirable to equalize the usage of resources (balance the load) in order to reduce the response time of jobs and improve the utilization of the resources.

The load balancing problem in a distributed resource system is not new and can take different forms for different problems. For example, in a long haul packet computer network (where the resources are the channels), load balancing becomes the routing problem. The goal there is to find optimum paths on which to distribute the packets, so that some well defined performance criterion (overall delay, for instance) is optimized.

In a distributed computer system, the load balancing problem may be formulated as the problem of distributing the execution of processes throughout the network in order to minimize overall user response time. This load balancing problem is in general more complex than the routing problem for the following reasons:

- (a) **Optimal selection of the execution site and optimal routing** from the originating site to the execution site must be simultaneously accomplished (although the routing problem may become trivial if a bus or ring network is used).
- (b) **Multiple job classes** must be considered (e.g., interactive, batch, etc.) with drastically different resource usage characteristics. In particular, some classes behave as "closed" classes, in that their population remains constant during the life of the system (e.g., the number of batch jobs in a multiprogrammed system). Other classes are best modeled as "open", in that the number of users in the class fluctuates statistically due to random inputs (e.g., database inquires originating from a very large terminal population).
- (c) Some classes may be restricted to run on a subset of sites.

Most load balancing problems can be formulated as non-linear, multicommodity flow problems. Downhill search techniques can be efficiently used for their solution. If the system includes only open classes and does not pose site restrictions, then any of the methods available for routing optimization in computer networks can be successfully used. In particular, the Flow Deviation method, a downhill search method specially designed for open queueing networks may be employed [Frat73, Klei76]. If the distributed system has a special structure (namely, there is only one class of customers and the local network can be modeled by a single queue) the optimal equilibrium point may be obtained directly by solving (numerically) a set of non-linear equations, rather than using an iterative procedure such as Flow Deviation. An elegant "direct" solution method for this special structure was presented in [Tant84]. Unfortunately, the direct solution does not easily extend to multiple classes, site constraints, and general interconnecting networks.

If the distributed system under consideration includes only closed chains (i.e., closed classes), then a recent extension of the Flow Deviation method due to Kobayashi and Gerla [Koba83] can be efficiently used. One must bear in mind, however, that the routing problem with multiple closed chains typically leads to local minima. Thus, a further search for the minimum of the local minima is required.

In this chapter we consider the more general situation of multiple mixed classes of customers. The open classes may correspond to interactive jobs submitted at terminals or work stations. These jobs may be permitted to execute remotely. At issue is the optimal selection of the remote site, and the optimal path to it. Restrictions may apply in the remote dispatcher. The closed classes

may correspond to batch jobs running locally on a computer site or to interactive users running local application programs. The computer site is itself modeled as a network of queues (central server model). Our goal is to minimize a weighted sum of delays (over open and closed chains). Note that only the open chains need to be rerouted for load balancing. Routing is fixed within the closed chains. It can be shown that this guarantees the existence of only one local minimum, which is also the global minimum.

The main contribution of this chapter is to provide an efficient exact algorithm for the optimum solution of a very general class of load balancing problems. The algorithm can be viewed as an extension of the approach in [Koba83].

In the following, in section 5.2 we describe in detail the problem to be solved. In section 5.3 we present the model. Section 5.4 contains the development of the solution and section 5.5 gives the proposed algorithm. In section 5.6 we present some examples. Section 5.7 concludes the chapter.

5.2. Problem Description.

We consider a distributed system as shown in Figure 5.1. In this system each site is composed of a number of resources (CPU's, disks, etc), which are used by processes that run in that site. Sites may not be identical, i.e., they may have different resource configurations and resources with different capacities.

Each site is connected to a large number of terminals. These terminals generate jobs that may have different processing requirements. For instance, some jobs may request more CPU time than others. Furthermore, a class of jobs may be restricted to run only in a subset of the sites in the network. This restriction may arise from security considerations, or from the fact that a site may

not possess all the resources required by the class.

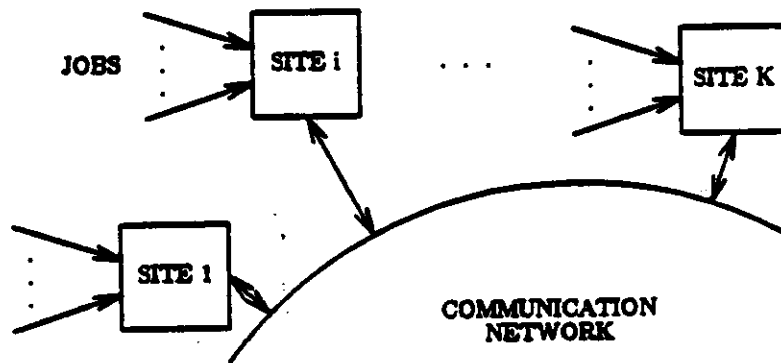


Figure 5.1. A distributed system.

We assume that a job may run until completion in its "local" site (which is defined to be the site connected to the terminal that generated the job), or may be transferred via the communication network to a "foreign" site. Once transferred to a foreign site, the job runs until completion and no further transfers are allowed.

This type of configuration may arise in a database application such as an airline reservation system where the requests of hundreds of terminals connected to different sites may also be processed at foreign sites. In addition, some or all of these sites may also be processing local application programs. We assume that these latter programs cannot execute in a foreign site. We refer to this type of programs as the "background load" of a site.

Our goal is to balance the load in the computer network, so that the overall delay is minimized. As a byproduct, the model will permit us to investigate several performance issues. For instance, we will be able to study the influence of the background load at a site on the assignment of database requests to sites, the effect of the speed of the communication network, etc.

5.3. The Model.

We model a distributed system as a collection of "central server models", one for each site, plus a queueing model for the communication network. This representation is identical to the one used in the preceding chapters. It was first proposed in [Gold83] to model the LOCUS distributed system.

Local application programs, or background load, are modeled by a closed chain for each site, as indicated in Figure 5.2. (In this figure we assume that

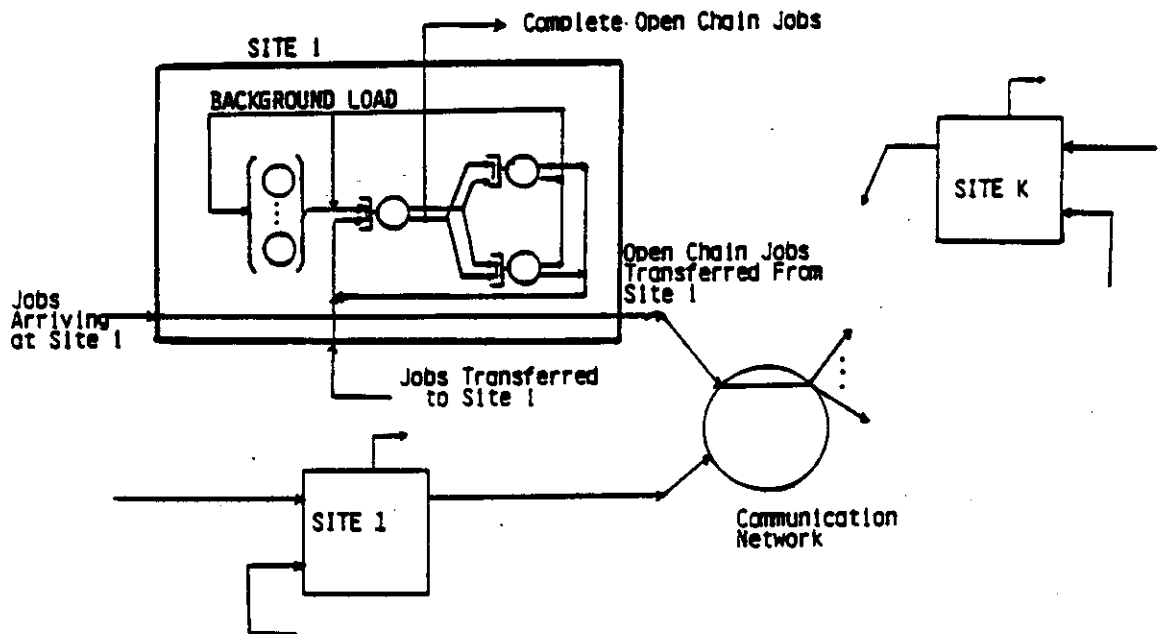


Figure 5.2. The model.

these local jobs are interactive.) We assume that the remaining jobs are generated by a large number of terminals, and so can be modeled as open chains with Poisson arrivals and total throughput Λ' . Upon arrival at site i a request r , can be either executed locally, or immediately transferred to one of the other

sites in the system that belong to the set of sites where r_i can be executed.

We assume that the service demands at each queue in the network are exponentially distributed. Each queue is either processor sharing (PS), or first-come-first-serve (FCFS) with fixed capacity, or "infinite server" (IS) (at FCFS queues it is assumed that different classes of jobs have the same service demands). Furthermore, the decision of transferring a job to another site is independent of the state of the network. In summary we have a product form mixed queueing network model.

We define the following notation, where superscripts c and o represent closed and open chain respectively (when the superscript is omitted in the table below, the parameter may refer to either an open or closed chain).

K^c	=	total number of closed chains representing background jobs at computer sites.
K^o	=	total number of open chains.
J	=	total number of service centers in the network (including the communication network).
N_k	=	population of the closed chain at site k .
\vec{N}	=	population vector = (N_1, \dots, N_K) .
$\lambda_k^c(\vec{N})$	=	mean throughput of the closed chain at site k for population \vec{N} .
λ_v^o	=	mean throughput of open chain v .
θ_{kj}	=	visit ratio of a chain k to service center j .
x_{kj}	=	mean service request of a chain k job at center j .
μ_{kj}	=	$1/x_{kj}$.
u_{kj}	=	relative utilization = $\theta_{kj} x_{kj}$.
ρ_j^o	=	total utilization of open chain jobs at center j .
λ_j^o	=	throughput (or flow) of open chain v at center j ($= \lambda_v^o \theta_{vj}$).
Λ^o	=	total open throughput = $\sum \lambda_v^o$.
$L_{kj}(\vec{N})$	=	mean number of chain k jobs at center j with population (\vec{N}) .
$L_j^c(\vec{N})$	=	total mean number of closed chain jobs at center j .
$L_j^o(\vec{N})$	=	total mean number of open chain jobs at center j .
e_k	=	unit multidimensional vector in the direction k .

We define the overall delay $D(\bar{N})$ as the following weighted sum of all chain delays:

$$D(\bar{N}) = \frac{w' \sum_{v=1}^K \lambda_v D_v(\bar{N}) + w^c \sum_{k=1}^K \lambda_k^N(\bar{N}) D_k(\bar{N})}{w' \sum_{v=1}^K \lambda_v + w^c \sum_{k=1}^K \lambda_k^N(\bar{N})} \quad (5.3.1)$$

where: (1) $D_v(\bar{N})$ and $D_k(\bar{N})$ represent the response time of open chain v and closed chain k , respectively; (2) $\lambda_k^N(\bar{N})$ represents the "nominal" throughput of a closed chain k obtained when all open chain throughputs are set to zero; (3) w' and w^c are arbitrary constant weighting factors for open and closed chains, respectively. Note that if the nominal throughput for a closed chain is replaced by the actual throughput, and the weights are set to 1, the expression above gives the total average delay. However, using total average delay as the objective function leads to unfairness to closed chains, since an increase in open chain utilization at a site causes an increase in local chain delay, but also a decrease in local chain throughput (the product remains constant by Little's result). This implies that at heavy load the impact of closed chain delays on the objective function becomes negligible with respect to the open chain delays. For this reason, we chose to assign a fixed coefficient, the nominal throughput, to each closed chain. In addition, we introduced the weighting factors $\{w\}$ to reflect the relative importance of open and closed chains in the model. Without loss of generality we assume $w' = w^c = 1$ throughout the rest of the chapter.

From Little's result:

$$D_v(\bar{N}) = \frac{\sum_{j=1}^J L_{vj}(\bar{N})}{\lambda_v} \quad \text{and} \quad D_k(\bar{N}) = \frac{N_k - \lambda_k(\bar{N})x_k}{\lambda_k(\bar{N})} \quad (5.3.2)$$

where x_k is the average think time of an interactive closed chain k job at the

terminal, and $\lambda_{ik}(\bar{N})z_{ik}$ gives the average number of closed chain k jobs at the terminals (for batch jobs, $z_{ik} = 0$).

Substituting (5.3.2) into (5.3.1),

$$D(\bar{N}) = \frac{\sum_{i=1}^I L_i(\bar{N}) + \sum_{i=1}^K \lambda_i^{N_i}(\bar{N})(N_i \lambda_i - z_{ii})}{\Lambda' + \sum_{i=1}^K \lambda_i^{N_i}(\bar{N})} \quad (5.3.3)$$

Our goal is to minimize the non-linear function $D(\bar{N})$ by optimally distributing the open chain traffic among the various sites. This is equivalent to optimizing $D(\bar{N})$ with respect to the open chain flows $\{\lambda_{vj}^o\}$, $\forall v, j$.

5.4. The Solution Approach.

We define our load balancing problem as follows:

Given:

- The service demands of jobs from all-chains at all service centers in the network.
- The visit ratios of the closed chain jobs (background jobs) of a site at each service center at that site, as well as the visit ratios of open chain jobs at centers of a site.
- The number of background jobs at each site.
- The throughput rate of each open chain v .
- The set of sites where open chain v jobs can execute.
- The "local" site of each open chain job.

Minimize: The overall delay $D(\bar{N})$.

With respect to: The open chain flows at each service center $\{\lambda_{vj}^o\}$.

The solution method we use is a downhill technique based on the "flow deviation" method [Frat73, Klei76]. The development we present below parallels the approach used by Kobayashi and Gerla [Koba83] to find the optimum rout-

ing in a closed queueing network. In this chapter we restrict the development to a single closed chain per site as indicated in section 3. However, the approach can be directly extended to multiple closed chains at each site.

To compute the steepest descent direction for the downhill technique we need to compute the (partial) derivatives of the overall delay function with respect to each open chain flow at each service center.

First, we should observe that a mixed product form queueing network with PS, FCFS fixed rate service centers and IS service centers can be reduced to an equivalent closed queueing network where the service requests x'_{kj} of closed chain k jobs at center j (PS or FCFS) is given by the following expression [Lave83]:

$$x'_{kj} = \frac{x_{kj}}{1 - \rho_j} \quad (5.4.1)$$

Second, we note that in our model, closed chain jobs (background load) at a site do not interfere with closed chain jobs from another site. Therefore, we can decompose our model into K^c independent mixed network models each with one closed chain, plus the queueing model of the communication network.

Finally, we recall that in a mixed network the open chain population at center i , L_i , can be expressed in terms of the closed chain by [Lave83]:

$$L_i(N_k) = \begin{cases} \frac{\rho_i}{1 - \rho_i} [1 + L_i(N_k)] & i \in \text{site } k \text{ or communication network, } i \neq \text{IS centers.} \\ \rho_i & i \in \text{site } k \text{ or communication network, } i = \text{IS centers.} \end{cases} \quad (5.4.2)$$

where the vector notation was dropped due to the reduction of the problem to $K^c + 1$ independent single chain mixed networks.

in Figure 4.6, if c visits an allocate node for a primary passive resource p , p is equivalent to a FCFS multiple server service center with as many servers as the number of tokens in the passive resource. The mean service time of the MS service center for passive resource p is given by equation (4.3) which is a generalization of equation (4.1a):

$$X_c = \sum_{j \in SH_c} \theta_{c,j} W_{c,j} \quad (4.3)$$

where SH_c is the R-set visited by customers of chain c simultaneously holding the passive resource p (i.e., customers from class c_j), $\theta_{c,j}$ is the visit ratio for class c , at resource j and $W_{c,j}$ is the waiting time from class c_j customers at center j .

b. For each subnetwork formed in Step 1b, the complement of the subnetwork for a local constrained chain c is represented by one or more closed chains. Each of these closed chains corresponds to a class c_j of chain c (i.e., they correspond to local constrained chain c when customers associated with this chain hold a primary passive resource p). For convenience, we introduce a fictitious primary passive resource which contains the same number of tokens as the number of chain c customers, to model the case where customers of this chain also visit resources in the subnetwork without holding any primary passive resource (*). These chains have as many customers as the number of tokens in the corresponding passive resource, and each such chain visits an infinite

(*) For illustration, consider the example in Figure 4.8a and suppose that there is a third class of chain c customers (class c_3) that visits subnetwork S without holding any passive resource. This model is equivalent to a model where chain c customers also visit a third passive resource P_3 with the same number of tokens as the number of chain c customers. This is true since there will never be a queue of customers waiting for passive resource P_3 .

server service center whose service time (delay) is given by equation (4.4) which is a generalization of equation (4.1a):

$$D_c = X_p \left(\frac{P_p}{U_p} - 1 \right) \quad (4.4)$$

where P_p and U_p are the number of tokens and the average utilization (*), respectively, of the passive resource $p \in S_p$.

Step 3. If a subnetwork S found in Step 1b still contains a passive resource, go to Step 1 using subnetwork S as the new network. Otherwise go to Step 4.

After Steps 1 through 3 are completed, a network is reduced to subnetworks of the form shown in Figure 4.9. Note that, due to Step 3, a chain c may belong to the set of local unconstrained chains of subnetwork S and to the set of local constrained chains of this same subnetwork.

Step 4. Obtain initial estimates for mean service times X_p of all passive resources $p \in S_p$. The estimates of all performance measures can be obtained by assuming that the network is unconstrained and using the estimates of the clustering algorithm of chapter 3, i.e., equations (3.3.a2) and (3.3.a3). Then, estimates of X_p are obtained from equation (4.3).

Step 5. Loop through the subnetworks and for each one solve the corresponding closed network. New estimates for $\{X_p\}$ and $\{D_c\}$ are obtained just before calculating the performance measures for a subnetwork using (4.3) and (4.4).

(*) The average utilization of the passive resource is the number of busy servers in the corresponding MS center.

Therefore, equation (5.3.3) can be rewritten as:

$$D(\bar{N}) = \frac{\sum_{\substack{i=1 \\ i \neq IS}}^J \frac{\rho_i}{1 - \rho_i} [1 + L_i(N_i)] + \sum_{\substack{i=1 \\ i \neq IS}}^J \rho_i + \sum_{k=1}^K \lambda_k^{N_k} (N_k \lambda_k(N_k) - z_{ki})}{\Lambda' + \sum_{k=1}^K \lambda_k^{N_k} (N_k)} \quad (5.4.3)$$

The independent variables in our optimization problem are the $\{\lambda_k\}$. Recalling that $\rho_j = \sum \lambda_{kj} z_{kj}$, we note that we can also use the $\{\rho_j\}$ as independent variables. In the following, for convenience, we take the derivative of $D(\bar{N})$ with respect to the total utilization of open chain jobs at center j (ρ_j). In this computation we exploit the fact that: (1) the performance measures of site (or communication network) u are not altered when we vary the open chain throughput at a center j in site (or communication network) k , $k \neq u$; and, (2) the total open throughput remains constant as do the nominal throughputs of the closed chains. For convenience of notation, we label the centers belonging to the same site (or communication network) as center j as $1, \dots, L$.

$$\frac{\partial D(\bar{N})}{\partial \rho_j} = \begin{matrix} (j \in \text{site } k \text{ or communication network}) \\ \left[\frac{1 + L_j(N_j)}{(1 - \rho_j)^2} + \sum_{\substack{i=1 \\ i \neq IS}}^L \frac{\rho_i}{1 - \rho_i} \frac{\partial L_i(N_i)}{\partial \rho_j} - \alpha(j) \frac{\lambda_j^{N_j} N_j \partial \lambda_j(N_j)}{(\lambda_j(N_j))^2 \partial \rho_j} \right] \frac{1}{\Lambda' + \Lambda^{N_c}(\bar{N})} \end{matrix} \begin{matrix} j \neq IS \\ j = IS \end{matrix} \quad (5.4.4)$$

where

$$\Lambda^{N_c}(\bar{N}) = \sum_{k=1}^K \lambda_k^{N_k} (N_k)$$

and

$$\alpha(j) = \begin{cases} 1 & j \text{ communication network} \\ 0 & \text{otherwise} \end{cases}$$

It remains to find the derivatives of the closed chain throughputs and queue lengths in relation to the utilization of open chain jobs at center j .

To this end, we first introduce the normalization constant for the corresponding closed network with service requests given by (5.4.1). Therefore, for site k [Bask75]:

$$G_k^c(N_k) = \sum_{|\bar{n}_k| = N_k} \prod_{\substack{i=1 \\ i \neq IS}}^L \left(\frac{a_k^i}{1 - \rho_i^*} \right)^{n_i} \times I_k(\bar{n}_k) \quad (5.4.5)$$

where

$$I_k(\bar{n}_k) = \prod_{\substack{i=1 \\ i \neq IS}}^L \frac{(a_k^i)^{n_i}}{n_i!} \quad \bar{n}_k = (n_1, \dots, n_L), \quad |\bar{n}_k| = n_1 + \dots + n_L$$

and the equilibrium probability of the state \bar{n}_k is:

$$P_k^c(\bar{n}_k) = \frac{1}{G_k^c(N_k)} \prod_{\substack{i=1 \\ i \neq IS}}^L \left(\frac{a_k^i}{1 - \rho_i^*} \right)^{n_i} \times I_k(\bar{n}_k) \quad (5.4.6)$$

Now, taking the derivative of (5.4.5) with respect to ρ_j^* , gives:

$$\frac{\partial G_k^c(N_k)}{\partial \rho_j^*} = \frac{1}{1 - \rho_j^*} \sum_{|\bar{n}_k| = N_k} n_j \prod_{\substack{i=1 \\ i \neq IS}}^L \left(\frac{a_k^i}{1 - \rho_i^*} \right)^{n_i} \times I_k(\bar{n}_k) \quad j \neq IS \quad (5.4.7)$$

$$= \frac{G_k^c(N_k)}{1 - \rho_j^*} \sum_{|\bar{n}_k| = N_k} n_j P_k^c(\bar{n}_k) \quad (5.4.8)$$

The summation on the right hand side of (5.4.8) is by definition the average queue length of closed chain jobs at service center j , therefore:

$$\frac{\partial G_k^c(N_k)}{\partial \rho_j^c} = \frac{G_k^c(N_k)}{1 - \rho_j^c} L_j^c(N_k) \quad (5.4.9)$$

Equation (5.4.9) is similar to one obtained by Kobayashi and Geria [Koba83].

In order to proceed with the development we need to take the second derivative of (5.4.5) with respect to ρ_j^c ,

$$\begin{aligned} \frac{\partial^2 G_k^c(N_k)}{\partial (\rho_j^c)^2} &= \frac{G_k^c(N_k)}{(1 - \rho_j^c)^2} \sum_{|\bar{n}_k| = N_k} n_j P_k^c(\bar{n}_k) + \frac{G_k^c(N_k)}{(1 - \rho_j^c)^2} \sum_{|\bar{n}_k| = N_k} n_j^2 P_k^c(\bar{n}_k) \\ &= \frac{G_k^c(N_k)}{(1 - \rho_j^c)^2} (L_j^c(N_k) + S_j^c(N_k)) \end{aligned} \quad (5.4.10)$$

where $S_j^c(N_k)$ is the second moment of queue lengths of the closed chain jobs at center j of site k .

Similarly, taking the derivative of (5.4.7) with respect to ρ_i^c , $i \neq j$, $i \neq k$:

$$\begin{aligned} \frac{\partial^2 G_k^c(N_k)}{\partial \rho_j^c \partial \rho_i^c} &= \frac{G_k^c(N_k)}{(1 - \rho_j^c)(1 - \rho_i^c)} \sum_{|\bar{n}_k| = N_k} n_i n_j P_k^c(\bar{n}_k) \\ &= \frac{G_k^c(N_k)}{(1 - \rho_j^c)(1 - \rho_i^c)} J_{j,i}^c(N_k) \end{aligned} \quad (5.4.11)$$

where $J_{j,i}^c(N_k)$ is the joint moment of queue lengths of the closed chain jobs at center j and i .

The second derivative of $G_k^c(N_k)$ with respect to ρ_j^c and the derivative of $\partial G_k^c(N_k) / \partial \rho_j^c$ with respect to ρ_i^c can also be obtained by taking the derivative of

equation (5.4.9) with respect to ρ_j^c and ρ_i^c , respectively. We obtain:

$$\begin{aligned} \frac{\partial^2 G_j^c(N_k)}{\partial (\rho_j^c)^2} &= \frac{1}{(1 - \rho_j^c)^2} G_j^c(N_k) L_j^c(N_k) + \frac{1}{1 - \rho_j^c} \frac{\partial G_j^c(N_k)}{\partial \rho_j^c} L_j^c(N_k) + \frac{1}{1 - \rho_j^c} G_j^c(N_k) \frac{\partial}{\partial \rho_j^c} L_j^c(N_k) \\ &= \frac{G_j^c(N_k)}{(1 - \rho_j^c)^2} \left(L_j^c(N_k) + (L_j^c(N_k))^2 + (1 - \rho_j^c) \frac{\partial}{\partial \rho_j^c} L_j^c(N_k) \right) \end{aligned} \quad (5.4.12)$$

$$\begin{aligned} \frac{\partial^2 G_j^c(N_k)}{\partial \rho_j^c \partial \rho_i^c} &= \frac{1}{1 - \rho_j^c} \frac{\partial G_j^c(N_k)}{\partial \rho_i^c} L_j^c(N_k) + \frac{1}{1 - \rho_j^c} G_j^c(N_k) \frac{\partial}{\partial \rho_i^c} L_j^c(N_k) \\ &= \frac{G_j^c(N_k)}{(1 - \rho_j^c)(1 - \rho_i^c)} \left(L_i^c(N_k) L_j^c(N_k) + (1 - \rho_i^c) \frac{\partial}{\partial \rho_i^c} L_j^c(N_k) \right) \end{aligned} \quad (5.4.13)$$

Now, equating (5.4.10) with (5.4.12) we obtain:

$$\begin{aligned} L_j^c(N_k) + S_j^c(N_k) &= L_j^c(N_k) + (L_j^c(N_k))^2 + (1 - \rho_j^c) \frac{\partial}{\partial \rho_j^c} L_j^c(N_k) \\ V_j^c(N_k) &= (1 - \rho_j^c) \frac{\partial}{\partial \rho_j^c} L_j^c(N_k) \end{aligned} \quad (5.4.14)$$

where $V_j^c(N_k)$ is the variance of the queue length of closed queue jobs at center j of site k .

Similarly, equating (5.4.11) with (5.4.13) we obtain:

$$\begin{aligned} J_{j,i}^c(N_k) &= L_i^c(N_k) L_j^c(N_k) + (1 - \rho_i^c) \frac{\partial}{\partial \rho_i^c} L_j^c(N_k) \\ V_{j,i}^c(N_k) &= (1 - \rho_i^c) \frac{\partial}{\partial \rho_i^c} L_j^c(N_k) \end{aligned} \quad (5.4.15)$$

where $V_{j,i}^c(N_k)$ is defined as the covariance of queue lengths of closed queue jobs at centers j and i of site k .

We note that a similar development can be used to obtain higher moments of queue lengths as a function of the derivative of lower moments with respect to an input parameter of the system. In appendix 4 we derive expressions for the variances in a queueing network model with multiple closed chains.

In order to find the derivatives of the closed chain throughput of site k ($\lambda_k(N_k)$) with respect to the open chain utilization at center j in k (ρ_j^k), we express $\lambda_k(N_k)$ in terms of the normalization constant [Lave83]:

$$\lambda_k(N_k) = \frac{G_k(N_k - 1)}{G_k(N_k)} \quad (5.4.16)$$

Now, taking the derivative of (5.4.16) with respect to ρ_j^k and using (5.4.9),

$$\begin{aligned} \frac{\partial \lambda_k(N_k)}{\partial \rho_j^k} &= \frac{\partial G_k(N_k - 1) / \partial \rho_j^k}{G_k(N_k)} - \frac{G_k(N_k - 1)}{(G_k(N_k))^2} \frac{\partial G_k(N_k)}{\partial \rho_j^k} \\ &= \frac{G_k(N_k - 1)}{G_k(N_k)} \times \frac{L_j(N_k - 1)}{1 - \rho_j^k} - \frac{G_k(N_k - 1)}{G_k(N_k)} \times \frac{L_j(N_k)}{1 - \rho_j^k} \\ &= \frac{\lambda_k(N_k)}{1 - \rho_j^k} (L_j(N_k - 1) - L_j(N_k)) \end{aligned} \quad (5.4.17)$$

where service center j is assumed to be in the same site of closed chain k .

Finally, substituting (5.4.14), (5.4.15) and (5.4.17) into (5.4.4) and noting that $\partial D(\bar{N}) / \partial \lambda_j^k = (1/\mu_j^k) \partial D(\bar{N}) / \partial \rho_j^k$, we obtain:

$$\frac{\partial D(\bar{N})}{\partial \lambda_j^k} =$$

$$\left\{ \begin{array}{l} \frac{1+L_j^q(N_k)}{1-\rho_j^q} + \frac{\rho_j^q V_j(N_k)}{1-\rho_j^q} + \sum_{\substack{i=1 \\ i \neq j, i \neq IS}}^L \frac{\rho_i^q V_{ij}(N_k)}{1-\rho_i^q} + \theta(j) \frac{\lambda_k^{N_c(N_k)}}{\lambda_k^q(N_k)} N_k |L_j^q(N_k) - L_j^q(N_k - 1)| \\ \hline \mu_{\eta_j}^q (1 - \rho_j^q) (\Lambda^q + \Lambda^{N_c(\bar{N})}) \end{array} \right. \quad j \neq IS$$

$$\left. \begin{array}{l} \frac{1}{\mu_{\eta_j}^q (\Lambda^q + \Lambda^{N_c(\bar{N})})} \\ \hline \end{array} \right. \quad j = IS \quad (5.4.18)$$

Equation (5.4.18) is given in terms of mean values of throughputs and queue lengths, variance and covariance of queue lengths. In the appendix we show that this equation can be easily obtained using mean value analysis (MVA) recursion.

5.5. The Algorithm.

The key to the flow deviation method is to associate a length or weight for an open chain job to each queue, given by:

$$l_{\eta_j} \triangleq \frac{\partial D(\bar{N})}{\partial \lambda_{\eta_j}^q} \quad j = 1, \dots, J \quad (5.5.1)$$

However, due to the particular characteristics of our problem, we can further simplify the solution by noting that, once a job is assigned to a site, its behavior is preestablished by the routing probabilities given as input parameters. In other words, the "route" of jobs within a site is fixed. This observation allow us to define a "site" weight, as:

$$l_{site_k} \triangleq \sum_{j \in k} \theta_{\eta_j}^q l_{\eta_j} \quad (5.5.2)$$

where the index k represents the site and index v , a particular open chain. In the same way, the weight of the communication network (l_{net_v}) can be defined.

The weight given by (5.5.2) represents the linear rate at which $D(\bar{N})$ increases with an infinitesimal increase of the flow of open chain v at site k . We also define $leit_{kv} \triangleq \infty$ if jobs of chain v are not allowed to visit site k .

We outline a flow deviation algorithm for load balancing (FDLB) in a distributed computer network of the type described in section 3. The algorithm is similar to the ones described in [Klei76] and [Koba83].

We define the assignment matrix \bar{A} where element A_{kv} gives the percentage of jobs of type v assigned to site k . Similarly we define the network vector \bar{F} where element F_v gives the percentage of jobs of type v assigned to a foreign site, i.e., $F_v = \sum_{k \neq \text{home site of } v} A_{kv}$.

We assume that we have an initial feasible assignment, i.e., initial values for \bar{A} so that $\rho_i^0 < 1$ for all service centers in the network which are not IS centers.

FDLB algorithm (*):

- Step 1: let $n = 0$ and let $\bar{A}^{(0)}$ be an initial feasible assignment.
- Step 2: Compute weights l_v using MVA and equations (A4.2.3) and (A4.2.4). Compute the weights of each site ($leit_{kv}$) and the weight of the network ($inet_v$) using equation (5.5.2).
- Step 3: For each class of open chain jobs, find the cheapest way to execute the job, i.e., find the matrix \bar{A}^* so that element $A_{kv}^* = 1$ if $leit_{kv} + net_{kv} < leit_{iv} + net_{iv}$ for all $i \neq k$, where

(*) The description of the algorithm follows similar descriptions given in [Klei76] and [Koba83]. However, in the implementation, we deviate the flow for one chain at a time, since it was observed that this equivalent approach tends to reduce the overall number of iterations.

$$net_{kv} \triangleq \begin{cases} 0 & \text{if } k \text{ is the local site for a job of class } v \\ lnet_v & \text{otherwise} \end{cases}$$

Find \bar{F} as defined above.

Step 4: Compute the incremental delay $\delta^{(n)}$ and δ^* for the assignment $\bar{A}^{(n)}$ and cheapest assignment \bar{A}^* , respectively:

$$\delta^{(n)} = \sum_v \left[\sum_i lrit_{iv} \times A_{iv}^{(n)} + lnet_v \times F_v^{(n)} \right]$$

$$\delta^* = \sum_v \left[\sum_i lrit_{iv} \times A_{iv}^* + lnet_v \times F_v^* \right]$$

Step 5: Stopping rule:

if $|\delta^{(n)} - \delta^*| < \epsilon$, where $\epsilon > 0$ is a properly chosen tolerance, stop. Otherwise go to Step 6.

Step 6: Find the value α in the range $0 \leq \alpha \leq 1$ such that the assignment $\bar{A}' = (1 - \alpha)\bar{A}^{(n)} + \alpha\bar{A}^*$ minimizes $D(\bar{N})$.

Let $\bar{A}^{(n+1)} = \bar{A}'$.

Step 7: Let $n = n + 1$ and go to Step 2.

In order to find a feasible initial assignment or, more generally, to determine whether a feasible solution exists we can apply one of the several methods already developed for routing in open networks [Klei76, Cant74]. We briefly describe here the method in [Klei76].

Step 1. Initially we assign all requests to their local sites. If, for all service centers j in the network, the flow due to open chain jobs at center j (ρ_j) is less than one, we have a feasible assignment. Otherwise let $n = 0$ and go to Step 2.

- Step 2. Scale down all open chain requests uniformly until the assignment above is feasible. Let this scale factor be $\eta^{(0)}$.
- Step 3. Find a new assignment using one iteration of the FDLB algorithm. (Note that at this point there is no need to find the optimum solution, since we only want to find an assignment that can possibly handle more requests.)
- Step 4. Scale up all open chain requests by an amount $\sigma = C(1 - (\rho_i^*)_{\max})$, where $(\rho_i^*)_{\max}$ is the maximum among all open chain flows at centers in the network and C is an appropriate chosen constant $0 < C < 1$. Let $\eta^{(n+1)} = \eta^{(n)} \left[1 + \frac{\sigma}{(\rho_i^*)_{\max}} \right]$ be the new scaling factor. (Note that the scaling factor η increases monotonically at each iteration.)
- Step 5. If the $\eta^{(n+1)}$ converges to a value less than one, the problem is unfeasible. If $\eta^{(n+1)}$ becomes greater or equal than one then set it to one and we have a feasible assignment. Otherwise, $n = n + 1$ and go to Step 3.

The FDLB algorithm finds the global minimum for $D(\bar{N})$ due to the convexity of $D(\bar{N})$ with respect to the open chain flows and the convexity of the space of the feasible flows. The convexity of $D(\bar{N})$ can be deduced from the fact that the function $D(\bar{N})$ in (5.3.3) is a sum of convex functions over the open chain flows. An intuitive (but not rigorous) explanation for that can be given as follows: we observe that equation (5.3.3) can be decomposed into a weighted sum of the number of open chain jobs in a site plus the response time of the closed chain jobs in that site. We can verify that both the number of open chain jobs in a site and the response time of closed chain jobs in that site are increasing

without bound (and with a non-decreasing rate) as a function of the open chain flow at that site. Thus, the delay of each site is a convex increasing function over the open chain flows, and the weighted sum of these delays is also convex. In appendix 3 we present more rigorous arguments in favor of the convexity of $D(\bar{N})$.

Convergence is guaranteed by the fact that each iteration provides an improvement in the objective function. It is worthwhile to note that if the background load is reduced to zero (no closed chain jobs) this algorithm is identical to the flow deviation algorithm reported in [Klei76], adapted for our load balance problem.

The amount of computation required by each iteration of the FDLB algorithm is only $O(\#sites \times \#open\ chains)$ for Step 3, since the computation of the "cheapest path" is trivial, and $O(\sum_{sites} (\#centers\ of\ site\ i) \times (\#background\ jobs\ of\ site\ i) + comm.\ net.)$ for the MVA algorithm, since the model consists of K^c independent single closed chains plus the model of the communication network. Note that in Step 6 the MVA algorithm is repeated several times, and the MVA computation becomes the dominant term.

5.6. Examples.

In this section we apply the FDLB algorithm to a distributed computer system consisting of 3 sites linked by a slotted ring, as illustrated in Figure 5.3. For the slotted ring, we used the closed chain model proposed by Bux [Bux81].

There are 2 classes of open chain jobs α_1 and α_3 arriving at sites 1 and 3, respectively. Class α_1 can request service from any of the 3 sites, but class α_3 can only request service from sites 2 and/or 3. Sites 1 and 2 have background

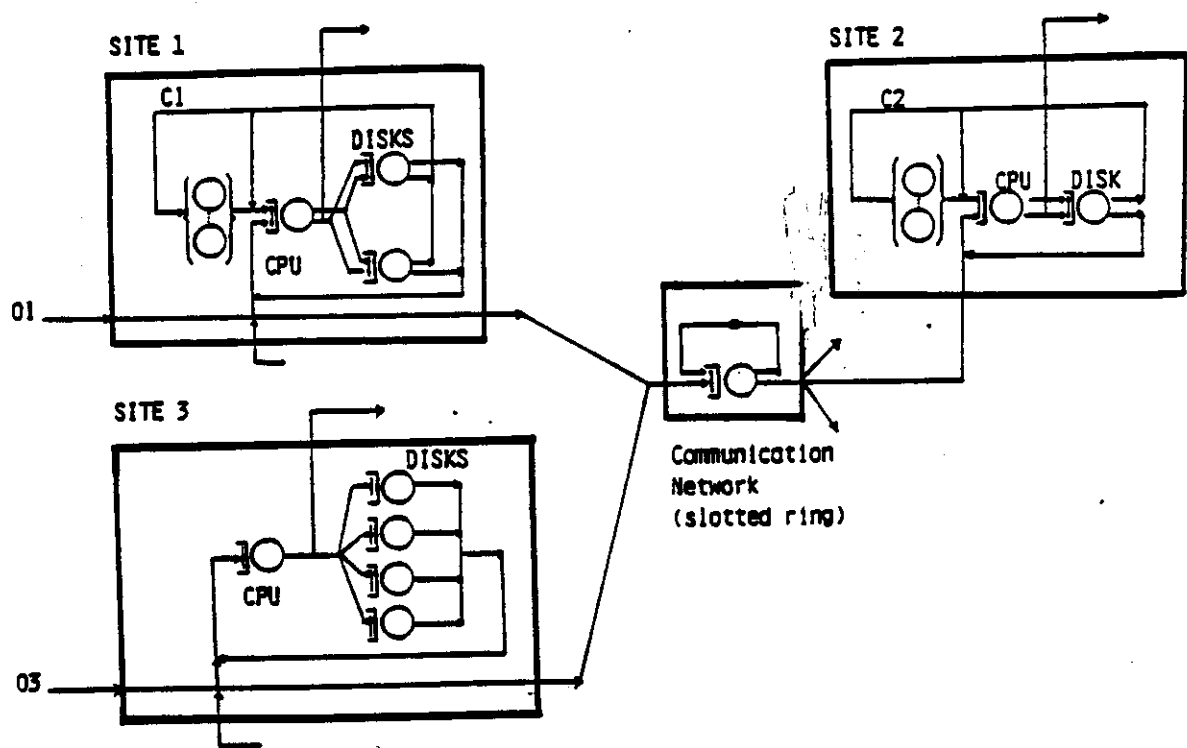


Figure 5.3. Example. A distributed computer system with 3 sites.

load, modeled as closed chains c_1 and c_2 respectively. Table 5.1 shows the visit ratios and service requirements for each class of job at each service center in the network, as well as other input parameters.

As an illustration, let us consider the behavior of jobs c_1 and c_1 at site number 1. A job of class c_1 spends an average of 30 msec at the CPU before issuing an I/O request. The service time of each I/O device is 50 msec. On the average, a job of class c_1 visits the CPU 5 times and the I/O devices 4 times before completion. On the other hand, a background job of class c_1 spends an average of 90 msec at the CPU and 50 msec at each I/O device, and it visits the

JOB CLASS	SITE	CENTER	TYPE	VISIT RATIOS	SERVICE TIMES (msec)
e1 6 jobs/sec	1	CPU	PS	5	30
		disk(1 or 2)	FCFS	2	50
	2	CPU	PS	5	21
		disk	FCFS	4	15
3	CPU	PS	5	30	
	disk(1 ... 4)	FCFS	1	140	
com. net.			PS		10
e3 3.5 jobs/sec	1 (*)	CPU	PS	5	50
		disk(1 or 2)	FCFS	2	50
	2	CPU	PS	5	35
		disk	FCFS	4	15
3	CPU	PS	5	50	
	disk(1 ... 4)	FCFS	1	140	
com. net.			PS		10
e1 5 jobs	1	terminals	IS	1	4000
		CPU	PS	10	90
		disk(1 or 2)	FCFS	5	50
e2 X jobs (variable)	2	terminals	IS	1	3000
		CPU	PS	10	40
		disk(1 or 2)	FCFS	10	15

(*) To illustrate the case where class e3 is not restricted to run at site 1.

Table 5.1. Parameters for the example.

CPU and I/O devices 10 times before a visit to the terminals.

Figure 5.4 shows the percentage of foreign jobs processed at site 2 (after balancing the load) when the number of background jobs at site 2 (N_2) varies from 0 to 20. When $N_2 = 0$, site 2 processes 80% of the jobs of class e1 and 36% of the jobs of class e3. When $N_2 = 20$, site 2 processes only 37% and 4% of the jobs of class e1 and e3, respectively. It is interesting to mention that when jobs of class e3 are allowed to be processed at site 1, and load balancing is applied, 22% of these jobs are processed at site 1 and 15% at site 2 (when $N_2 = 0$). Furthermore, all jobs of class e1 are sent to site 2.

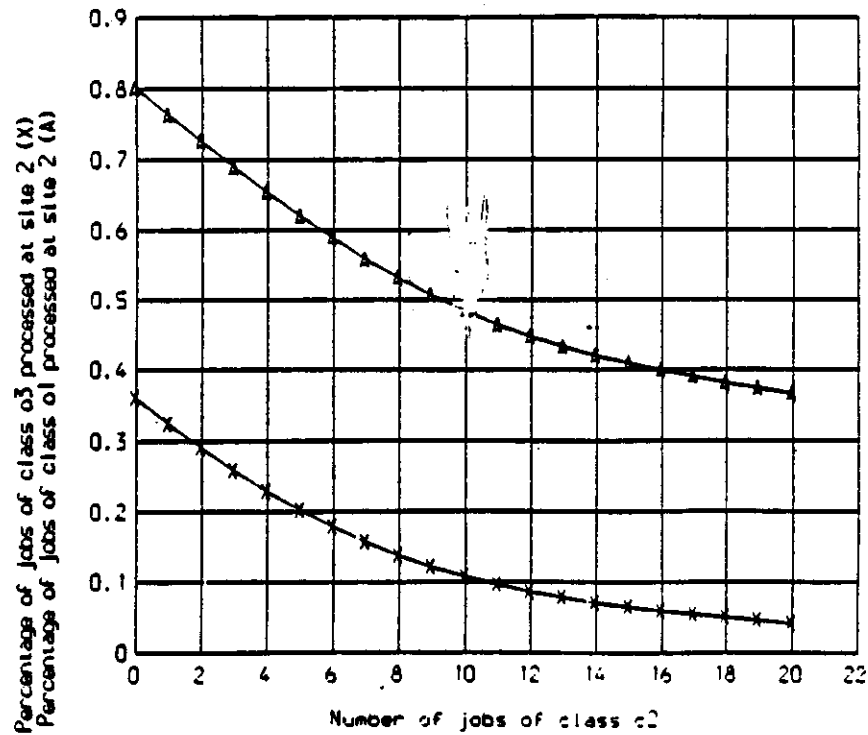


Figure 5.4. Percentage of foreign jobs processed at site 2.

Figure 5.5 shows the effect of load balancing on overall delay as a function of the number of background jobs at site 2. The effect is illustrated by plotting the relative difference between the delays when load balancing is not used and when it is used (*). For instance, when $N_2 = 0$ and load balancing is used the average overall delay is .95 sec in contrast with 9.2 sec when load balancing is not used (the relative difference is 870% in this case).

(*) We define relative difference of the delays as: $(\text{Delay without load balancing} - \text{Delay with load balancing}) / \text{Delay with load balancing}$.

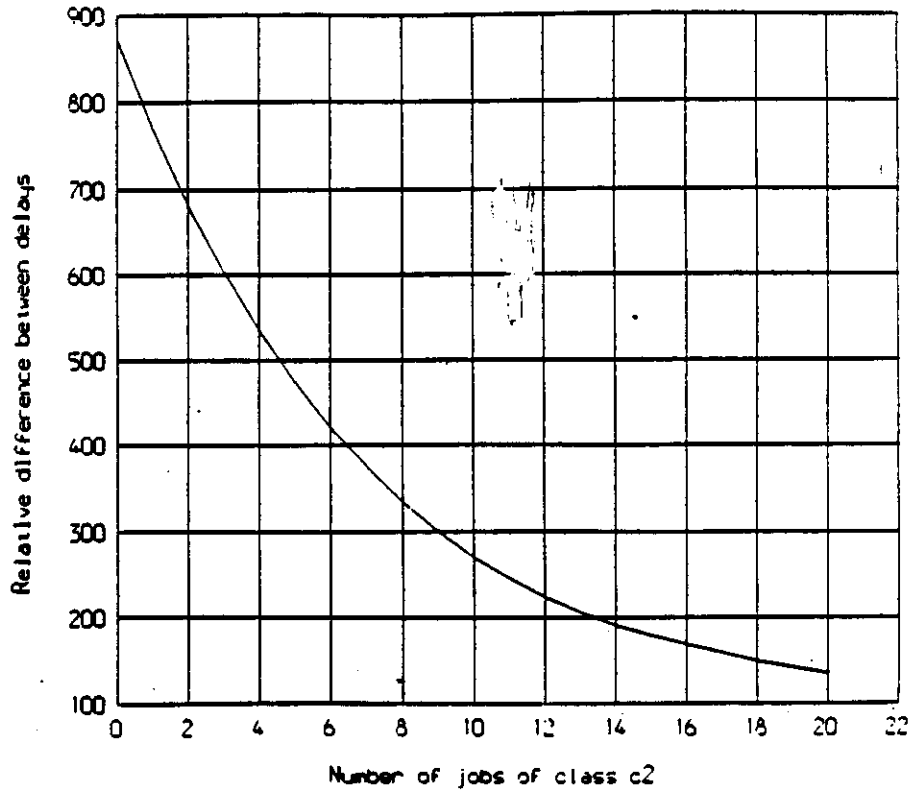


Figure 5.5. Relative difference (%) of the overall delays.

5.7. Conclusions.

We have presented an efficient method for balancing the load in a distributed system with heterogeneous computer sites and multiple classes of customers. Jobs of a particular class may be restricted to run only on a subset of the sites in the network. Each site, as well as the communication network, are represented as product form queueing networks.

We took into consideration the effect of the background load at each site, modeled as closed chain jobs. The effect of the communication delay incurred when a job requests service at a foreign site was also considered.

Our approach is based on a downhill search technique known as the Flow Deviation method. We have defined weights for each site in the network which allow the computation of the assignment which will reduce the overall delay $D(\vec{N})$. The parameters necessary for the computation of these weights can be easily obtained using the MVA algorithm.

The definition of the delay function presented in equation (5.3.1) is suitable for studying the behavior of the network under different minimization requirements. In particular, by setting $w' = 0$ ($w' = 0$) we can study the effect of load balancing when the object is to minimize the delay of the open (closed) chain jobs only.

The method developed in this chapter can be easily extended to account for multiple closed chains at each site, by using the results presented in appendix 4. Furthermore, the method is not restricted to the function defined in (5.3.1), and can be used to minimize other objective functions as well. More complex communication network structures can also be considered. In the next chapter we explore other applications.

CHAPTER 6

EXTENSIONS TO THE LOAD BALANCING ALGORITHM.

6.1. Introduction.

In the previous chapter we developed an exact algorithm to balance the load in a network of computer systems. We assumed that the network could be modeled as a mixed network of queues where transactions were modeled as open chain jobs and any background load was modeled as closed chain jobs. We considered the case where only the open chain jobs could (possibly) execute in a foreign site. Furthermore, once an "execution site" is chosen for a job, it runs on that site until completion. In this chapter we investigate several extensions to the basic approach developed in the previous chapter. In section 6.2 we extend our approach to account for jobs that request service in more than one site before completion. In section 6.3 we propose an approximate algorithm to solve for a general multiple chain closed network where closed chain jobs can be rerouted. Finally in section 6.4 we describe other possible extensions of the algorithm and conclude the chapter.

6.2. Jobs with Multiple Tasks.

6.2.1. Problem Description and Model.

We again consider a distributed system as shown in Figure 5.1, where sites may be different, i.e., they may have different resource configurations and resources with different capacities. Each site is connected to a large number of terminals which generate jobs that may have different processing requirements.

Unlike the problem of chapter 5, a job may be composed of several tasks. Each site in the network may run a subset of the tasks of a job. These restrictions may arise from the fact that a site may not possess all the resources required by a particular task. Tasks are executed in sequence and, for each task of a job, a "run site" is chosen. A job completes execution after all individual tasks are executed. Similarly to chapter 5, our goal is to balance the load in the computer network, so that the overall delay is minimized.

The model we will use for a distributed system is the same as in chapter 5, i.e., it is formed by a collection of central server models with terminals plus a model for the communication network. Local application programs or background load are modeled by a closed chain for each site and the remaining jobs are modeled as open chains.

Jobs representing local application programs are restricted to run on the site where they were generated, i.e., their local site. All the other jobs in the model (the open chain jobs) may execute in more than one site. These jobs require the execution of several tasks in sequence before completion. Suppose that a request r_i is composed by tasks t_1, \dots, t_M . Upon arrival at a site i , the first task of r_i (t_1) may be executed locally or the request may be immediately

transferred to one of the other sites in the system which can execute task t_1 . Task t_1 runs until completion in the chosen site. Then, a new site is chosen to execute task t_2 , and the job completes when all tasks are executed.

We use the same assumptions stated in section 5.3 to obtain a product form mixed queueing network. Furthermore, we use the same overall delay definition as given by equation (5.3.1). Our goal is to minimize the non-linear function $D(\vec{N})$ by optimally distributing the open chain traffic among the various sites.

6.2.2. The Solution Approach.

Similarly to chapter 5, we define our load balancing problem as follows:

Given:

- The number of tasks of each job.
- The service demands of jobs (and their tasks) from all chains at all service centers in the network.
- The visit ratios of the closed chain jobs (background jobs) of a site at each service center at that site, as well as the visit ratios of open chain jobs at centers of a site.
- The number of background jobs at each site.
- The throughput rate of each open chain v .
- The set of sites where task t_i of open chain v jobs can execute.
- The "local" site of each open chain job.

Minimize: The overall delay $D(\vec{N})$.

With respect to: The open chain flows at each service center $\{\lambda_{i,v}^o\}$. (Where the subscript i identifies a task of an open chain v job.

We show that the solution method used to solve this problem can be easily mapped to the approach used in chapter 5, with only a minor modification. In the previous chapter each open chain job was composed of only one task which could be assigned to one of the sites that was able to execute this task. Therefore, the "routing" of a job through the network was very simple since, once assigned to a site, a job executes until it completes. Figure 8.1 shows the possible paths for a job of an open chain v in a distributed system

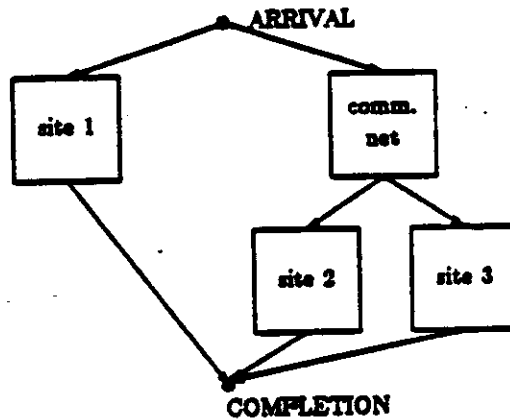


Figure 8.1. Possible paths of chain v jobs with only one task.

with three sites (1, 2 and 3) and a communication network. We assume that site 1 is the local site of chain v jobs and that they can be executed in any of the three sites. In this Figure, we see that only three paths are possible: a job may execute in its local site (1) or it may be sent via the communication network to foreign sites 2 or 3. If a job has multiple tasks to be executed, the route of this job through the distributed system is more complex, since the job can visit several sites before completion. The route of a chain v job through the system can be modeled by introducing classes for that chain, with one class mapped to

each task of a job. Since a queueing network model with chains having multiple classes can be solved by an equivalent system with the same number of chains but with only one class [Lave83], the introduction of classes does not increase the computational complexity of the model. However, the algorithm has to take

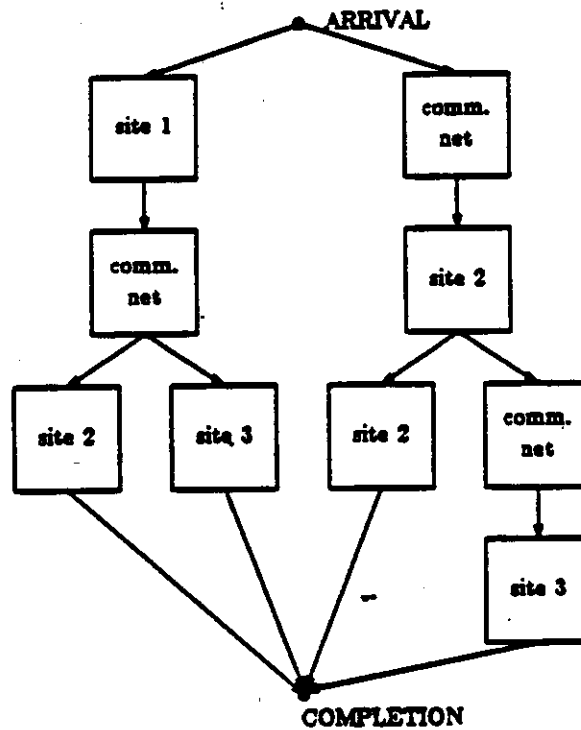


Figure 6.2. Possible paths of chain v jobs with two tasks.

into account a more complex routing structure. This is illustrated in Figure 6.2. In that Figure, we assume that chain v jobs are composed of two tasks (t_1 and t_2). Task t_1 can only be executed in sites 1 or 2 and task t_2 can only be executed in sites 2 or 3. We see that there are four possible paths of execution for a job. In one of these paths, a job is sent through the communication network to have its first task executed at site 2. Then the job is sent again through the com-

munication network to have its second task executed at site 3.

From the above discussion we can identify the changes that have to be made in the solution approach of chapter 5. These are:

1. Equation (5.18) is replaced by $\partial D(\bar{N}) / \partial \lambda_{v,j}^i = (1/\mu_{v,j}^i) \partial D(\bar{N}) / \partial \rho_j^i$ where the subscript i identifies a class of open chain v , and $\partial D(\bar{N}) / \partial \rho_j^i$ is the same as before.

2. A length or weight is associated for a class of an open chain job to each queue:

$$l_{v,j} \triangleq \frac{\partial D(\bar{N})}{\partial \lambda_{v,j}^i} \quad j = 1, \dots, J \quad (6.2.1)$$

Similarly a site weight is defined:

$$l_{sit_{k,v}} \triangleq \sum_{i,t} \theta_{v,j}^i l_{v,j} \quad (6.2.2)$$

as well as the communication network weight:

$$l_{net_v} \triangleq \sum_{j \in net} \theta_{v,j}^i l_{v,j} \quad (6.2.3)$$

The weight given by (6.2.2) represents the linear rate at which $D(\bar{N})$ increases with an infinitesimal increase of the flow of open chain v , class i , at site k . We also define $l_{sit_{k,v}} \triangleq \infty$ if task i of chain v jobs can not be executed at site k .

3. The assignment matrix of chapter 5 is replaced by a tri-dimensional matrix \bar{A} where element $A_{k,v}$ gives the percentage of jobs of type v , class i assigned to site k . The network vector in chapter 5 is replaced by the network matrix \bar{F} where elements F_v gives the percentage of jobs of type

v , class i assigned to a foreign site.

Once the above minor modifications are introduced, the algorithm of section 5.5 remains basically the same. The modifications necessary are related to the search for the optimum execution path. In this way, Step 1 has to include the identification of all possible execution paths for jobs of a particular chain. This is easily accomplished since tasks run until completion in their assigned site and in sequence. Therefore, the possible execution paths can be built in a tree structure. Step 3, which computes the assignment matrix of the open chain jobs has also to be modified to find the optimum path taking into account a more complex structure as illustrated by the example of Figure 6.2. Any known algorithm to find the optimum path can be used (e.g., see [Klei76]).

In summary, the only modifications in the algorithm of chapter 5 are the introduction of classes and the search for the optimum path. The function $D(\bar{N})$ remains essentially the same since, as we have already pointed out, the introduction of classes can be mapped into a single class problem. Therefore, the convexity of $D(\bar{N})$ can be deduced from the convexity of the delay function of chapter 5, which implies that the algorithm of this section also finds the global minimum for $D(\bar{N})$.

The amount of computation required by each iteration of the algorithm is essentially the same of the algorithm of chapter 5 since the introduction of classes do not increase the computational requirements to solve the model. However, additional effort has to be spent in the computation of the shortest path, although efficient algorithms exist to solve this problem. The computation of all possible paths for the jobs is trivial and it is done only once at the initialization of the algorithm.

6.2.3. Example.

In this section we apply the new FDLB algorithm to the same distributed computer system of section 5.6. In this example we assume that there is one class of open chain jobs correspondent to class σ_1 in the example of section 5.6. However, each job from this class is formed by two tasks which have to be executed in sequence before completion of this job. We assume for this example that the two tasks have identical computational requirements, however the first task (t_1) can only be executed in sites 1 or 2 and the second task (t_2) can only be executed in sites 2 or 3. The possible paths of execution for a chain σ_1 job is illustrated in Figure 6.2.

Table 6.1 shows the optimum load balancing strategy for the network described above for three cases: when the number of background jobs of site 2 (N_2) is equal to 1 and the time to transfer a task over the communication network is assumed to be 10 msec; when N_2 is equal to 20; and when the transfer

	tasks	Percentage of tasks of jobs of class σ_1 processed at a site			commun. net. throughput	overall delay
		site 1	site 2	site 3	(jobs/sec)	(sec)
c_2 : 1 job comm. net. delay: 10 msec	t_1	33.9	66.1		7.08	1.92
	t_2		48.2	51.8		
c_2 : 20 jobs comm. net. delay: 10 msec	t_1	67.3	32.7		6.96	5.95
	t_2		17.2	82.8		
c_2 : 1 job comm. net. delay: 160 msec	t_1	42.4	57.6		6	8.75
	t_2		57.6	42.4		

Table 6.1. Optimal load balancing for the example of section 6.2.

time over the communication network is increased to 160 msec, for N_2 equal to 1.

As we can observe from the results, in the first case site 1 processes 33.9% of tasks t_1 , since it has a considerable amount of background load. Site 2 processes the majority of tasks t_1 and almost half of tasks t_2 . As the background load of site 2 increases to 20 jobs, we see a decrease of the percentage of tasks processed at this site. Now the majority of tasks t_1 is processed at site 1 and the majority of tasks t_2 is processed at site 3. In the third case, when the communication delay increases significantly (N_2 is equal to 1 in this case), we observe that tasks are migrated so that the communication network stays with the minimum utilization possible as we can observe by the value of the throughput in the communication network, which is equal to the value of chain a_1 jobs. Finally it is interesting to observe again the importance of load balancing. For the first case above, (N_2 equal to 1 and the communication delay equal to 10 msec), if all tasks t_1 are executed in the first site and tasks t_2 are split equally between sites 2 and 3, the overall delay is 12.6 sec, a 556% increase in relation to the same network when load balancing is used.

6.3. An Approximate Algorithm for the Routing Problem in a Queueing Network Model with Multiple Closed Chains.

6.3.1. Problem Description.

Kobayashi and Gerla [Koba83] proposed an algorithm for the optimum routing of traffic in a closed queueing network. The algorithm is computationally efficient for single chain networks. Furthermore, the algorithm finds the global minimum since the object function to be minimized ($D(N)$) is convex with respect to the visit ratios of the single closed chain jobs to the service centers in the network. Unfortunately, for multiple chain networks the computational complexity of the algorithm increases drastically mainly due to Step 6 of their

algorithm, i.e., the step where the amount of flow to be deviated to the shortest path is found so that $D(\bar{N})$ is minimized. This is true since the MVA algorithm for multiple closed chain networks has to be executed several times in this step. Moreover, a search for the global optimum would have to be done since it was shown that $D(\bar{N})$ is not convex with respect to the visit ratios, so the algorithm finds only a local minima.

In this section we propose an approximate algorithm to solve the routing problem in a multiple closed chain queuing network model. The algorithm reduces the computational complexity of the original algorithm by reducing the amount of computation required in Step 6 of this algorithm. Although we haven't done extensive tests to evaluate the accuracy of the approximate algorithm, limited studies seems to indicate that the accuracy of the final result is good.

6.3.2. The Approximate Algorithm.

In the original algorithm proposed in [Koba83] for multiple closed chain networks, K "flow deviations" must be performed, one for each chain. In order to find the amount of chain k flow to be deviated to the shortest path in the current iteration step, the value of α_k in the range $0 \leq \alpha_k \leq 1$ has to be found so

that $\bar{\theta}_k' = (1 - \alpha_k)\bar{\theta}_k^{(n)} + \alpha_k\bar{\theta}_k^*$ minimizes $D(\bar{N}) = \frac{\sum_i N_i}{\sum_i \lambda_i}$ as defined in [Koba83], where

$\bar{\theta}_k^{(n)} = (\theta_{k1}^{(n)}, \dots, \theta_{kT}^{(n)})$ are the visit ratios of chain k jobs in the current iteration step n and $\bar{\theta}_k^*$ are the visit ratios obtained when all the chain k flow is deviated to the shortest path. As we have already mentioned, the determination of the α_k 's requires the execution of the MVA algorithm for the multiple closed chain network several times. These steps contribute most to the overall execution time of

the algorithm.

For the approximate algorithm we propose, we use the notion of foreign chains as in chapter 3 and 4. Our basic assumption is that the value α_k for a closed chain k can be approximately calculated by solving a subnetwork S_k which contains all the centers in the original network, but chain k is the only local chain in this subnetwork and the remaining chains are in the set of foreign chains of S_k . Using our approximation of chapter 3 we recall that the combined effect of all foreign chains at service center j is Poisson with total utilization U_j given by equation (3.3.3). Using these results, the value of α_k is calculated so that the average delay of all chains, i.e., all open chains and the single closed chain k in S_k , is minimized. In summary, we reduce the routing problem of a network with K closed chains to the routing problem of K mixed single closed chain networks where the object is to minimize the delay of the closed and open chains.

Before outlining the approximation algorithm, we need to find the path lengths for a chain k . For that, we first define the average delay $D_k(N)$ for subnetwork S_k which contains all centers of the original network, chain $k \in LC(S_k)$ and the remaining chains are foreign.

$$D_k(N) = \frac{\sum_{j=1}^J \lambda_j D_j(N) + \lambda_k(N) D_k(N)}{\Lambda' + \lambda_k(N)} \quad (6.3.1)$$

where the $\{\lambda_j\}$ are the open chain rates to center j inferred by equation (3.3.3), the summation is over all centers in the network and $\Lambda' = \sum_{i \neq k}^K \lambda_i(N)$. By Little's result:

$$D_j^*(N) = \frac{L_j^*(N)}{\lambda_j} \quad \text{and} \quad D_k(N) = \frac{N_k}{\lambda_k(N)} \quad (6.3.2)$$

The average number of open chain jobs at center j , $L_j^*(N)$ can be obtained from the average number of closed chain jobs at the same center [Lave83]:

$$L_j^*(N) = \frac{\rho_j}{1 - \rho_j} [1 + L_{bj}(N)] \quad (6.3.3)$$

where ρ_j is the total utilization of open chain jobs at center j and is equal to U_j , defined in equation (3.3.3). Substituting (6.3.2) and (6.3.3) in (6.3.1) we obtain:

$$D_k(N) = \frac{\sum_{j=1}^I \frac{U_j}{1 - U_j} [1 + L_{bj}(N)] + N_k}{\Lambda^* + \lambda_k(N)} \quad (6.3.4)$$

Taking the derivative of $D_k(N)$ with respect to θ_{bj} , and following similar steps to obtain the results shown in appendix 4, we obtain:

$$\frac{\partial D_k(N)}{\partial \theta_{bj}} = \frac{\left(\frac{U_j V_{bj}(N)}{1 - U_j} + \sum_{i \neq j} \frac{U_i CV_{bi}(N)}{1 - U_i} \right) (\Lambda^* + \lambda_k(N)) + \left(\sum_{i=1}^I \frac{U_i [1 + L_{bi}(N)]}{1 - U_i} + N_k \right) \lambda_k(N) (L_{bj}(N) - L_{bj}(N-1))}{\theta_{bj} (\Lambda^* + \lambda_k(N))^2} \quad (6.3.5)$$

where $V_{bj}(N)$ and $CV_{bi}(N)$ are the variance of queue lengths at center j and the covariance of queue lengths at center i and j , when the subnetwork k is being solved. In appendix 4 we have shown that the variance and covariance of queue lengths can be easily calculated recursively using (A4.2.3) and (A4.2.4), therefore (6.3.5) can be obtained using MVA.

In order to apply the flow deviation method we associate a length or weight to a chain k job at center j given by:

$$l_{kj} \triangleq \frac{\partial D_k(N)}{\partial \theta_{kj}} \quad j = 1, \dots, J \quad (6.3.6)$$

The weight l_{kj} above represents the linear rate at which the delay $D_k(N)$ of subnetwork S_k increases with an infinitesimal increase of the chain k flow at center j . We also define $l_{kj} \triangleq 0$ if $\theta_{kj} = 0$ and $l_{kj} \triangleq \infty$ if jobs of chain k are not allowed to visit center j .

The algorithm we describe next is similar to the one described in [Koba83]. The main differences are:

- (1) we perform K flow deviations of K single closed chain mixed networks.
- (2) the queue weights l_{kj} are calculated for each single closed chain subnetwork S_k , taking into consideration the approximation of the effect of the remaining closed chains in the network.

The Algorithm.

- Step 1. Let $n = 0$ and $\vec{\theta}^{(0)} = (\vec{\theta}_1, \dots, \vec{\theta}_K)$, $\vec{\theta}_k = (\theta_{k1}, \dots, \theta_{kJ})$ be the initial feasible flow. Let $k = 1$.
- Step 2. Compute the throughputs $\lambda_k(\vec{N}) \forall k$ by solving the entire network with MVA or an approximation algorithm, using the current visit ratios $\vec{\theta}^{(n)}$.
- Step 3. Form a subnetwork S_k which contains all centers in the network, chain k is the only local chain in S_k and all the other chains are foreign. Compute the weights l_{kj} ($j = 1, \dots, J$) using equation (6.3.6).
- Step 4. Solve the shortest path problem with the above weights. (Note that only the single closed chain k in S_k may be rerouted.) Let $\vec{\theta}_k^*$ be the

flow vector obtained by sending a unit flow on the shortest path.

Step 5. Compute the incremental delay (per unit of flow) $f_i^{(n)}$ and f_i^* for the flow $\bar{\theta}^{(n)}$ and the shortest flow $\bar{\theta}^*$, respectively:

$$f_i^{(n)} = \sum_j l_{ij} \theta_j^{(n)}$$

$$f_i^* = \sum_j l_{ij} \theta_j^*$$

Step 6. Find the value α_i in the range $0 \leq \alpha_i \leq 1$ such that the flow $\bar{\theta}_i' = (1 - \alpha_i)\bar{\theta}_i^{(n)} + \alpha_i\bar{\theta}_i^*$ minimizes $D_i(N)$ given in (6.3.4). Let $\bar{\theta}_i^{(n+1)} = \bar{\theta}_i'$.

Step 7. Let $k = k + 1$. If $k \leq K$ go to Step 2, otherwise go to Step 8.

Step 8. (Stopping rule). If $|f_i^{(n)} - f_i^*| < \epsilon \forall k$, where $\epsilon > 0$ is a properly chosen tolerance, stop. Else go to Step 9.

Step 9. Let $n = n + 1$ and go to Step 2.

This algorithm finds a local minima for $D(\bar{N})$ since this function was demonstrated not to be convex by an example in [Koba83]. This approximation algorithm requires much less computational effort than the algorithm in [Koba83] since Step 6 (the most time consuming Step in the original network) is reduced to a single chain MVA calculation. Furthermore, an approximation algorithm can be used in Step 2, eliminating the need to use exact MVA for a multiple chain network.

6.3.3. Example

In this section we apply our approximation algorithm to the same multiple chain network example presented in [Koba83]. Figure 6.3 shows this net-

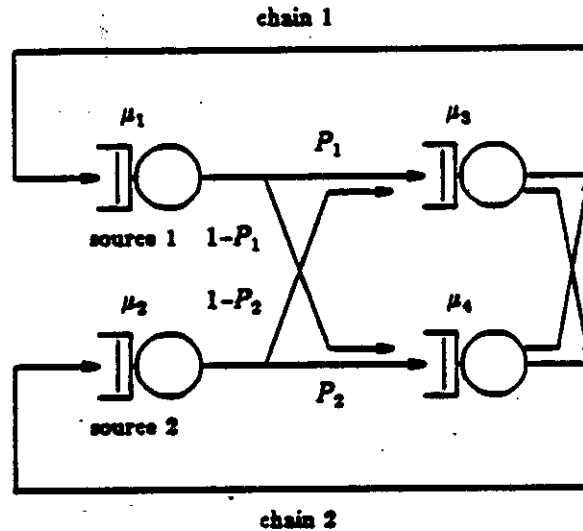


Figure 6.3. A multiple chain closed network. Example of section 6.3.3.

work. There are two source queues and two channel queues, whose service rates are given in Table 6.2. The population vector \bar{N} was chosen to be (3,3). As

j	1	2	3	4
μ_j	2	1	2	1

Table 6.2. Service rates for the example of Figure 5.3.

shown in Figure 6.3 there are two possible paths for each chain. Following what was done in [Koba83] we start from four different initial routing patterns as shown in Table 6.3. Like in [Koba83], these four initial routing patterns converge to two different local minima having different delay values. Table 6.4

presents the values of these two local minima. Initial patterns #1 and #2 converge to local minimum #1 and the others lead to local minimum #2. Also

	# 1	# 2	# 3	# 4
P_1	1	0	0	0.5
P_2	0	0	1	0.5
$D(\bar{N})$	3.173	3.214	6.016	3.309
$\lambda(\bar{N})$	1.891	1.867	0.997	1.813

Table 6.3. Initial routing pattern for the example of Figure 6.3.

	# 1		# 2	
	approx.	[KOBAS3]	approx.	[KOBAS3]
P_1	0.4332	0.544	1	1
P_2	0	0	1	0.975
$D(\bar{N})$	2.828	2.804	2.667	2.666
$\lambda(\bar{N})$	2.122	2.140	2.25	2.25

Table 6.4. Two local minima for the example of Figure 6.3.

shown in Table 6.4 the two local minima found by the algorithm in [Koba83]. As we observe from the Table, the values of $D(\bar{N})$ and $\lambda(\bar{N})$ in the two cases are extremely close.

6.4. Other Possible Extensions and Conclusions.

We have extended the results of chapter 5 to account for jobs with multiple tasks. We have shown that the same algorithm of section 5.5 can be applied with only minor modifications. We have proposed an approximate algorithm for solving the routing problem with multiple closed chains. The computational

requirements of our approximate algorithm are considerably less than the original "exact" algorithm in [Koba83] since it reduces the problem to the solution of several single closed chain mixed subnetworks. Our preliminary tests seem to indicate that the algorithm is accurate. However, much more tests need to be done to fully evaluate the accuracy of this approximation.

There are several possible applications and extensions that could be developed based on the results of chapters 5 and 6. One such application is the routing of packets through gateways of interconnected packet switching networks. Each network could be modeled as a multiple closed chain queueing model. The packets which have to travel from network to network through gateways may be modeled as open Poisson sources to these networks. The problem is to choose the routing of packets through the gateways so that an objective function is minimized. Another possible extension is to investigate the routing of tasks which are generated from other ones in a distributed system. These "split" tasks were modeled by Heidelberger and Trivedi [Heid82] as open Poisson sources with rate dependent on the closed chain rate of the jobs which generated the task. In a distributed environment similar to the one portrayed in Figure 5.1, these split tasks may represent the background load generated by interactive users. As an example, in our current LOCUS configuration some batch jobs generated by interactive users (e.g., compilation, nroff and others) are routed to a couple of "server nodes" which are assigned primarily to handle this background load. It seems that this situation could be accurately modeled by a queueing model similar to the one in Figure 5.2 plus the split tasks. Presently little is known about the efficiency of the heuristic strategy currently implemented at UCLA-LOCUS. An interesting problem would be the study of the optimum routing strategy for these background jobs so that the overall delay is

minimized and the comparison of the results obtained with the current implementation. We note that the resulting model is a mixed queueing network similar to the one described in chapter 5, but with a key difference, since the open chain rates are dependent on the closed chain rates. However, we believe that the same approach used in chapter 5 and 6 could also be used to solve this problem. Other extensions include the study of distributed routing algorithms similar to what was proposed in [Gafn84].

Finally, we point out that all the results presented in chapter 5 and 6 can be easily extended to account for queueing networks with centers that have the type of load dependent service rates studied in [Heff82].

CHAPTER 7

CONCLUDING REMARKS.

The research presented in this dissertation was motivated by our interest in analyzing queueing network models of large distributed systems, in particular the UCLA-LOCUS system. The basic representation used for each site was a central server model with terminals. These models were interconnected to represent the interaction among sites [Gold83]. Our main contribution was the development of efficient algorithms which can solve queueing models of moderate to large systems. The efficiency of the algorithms allows the analyst to include more detail in a model and provide the possibility to investigate many design alternatives.

We have proposed an iterative approximation technique to solve queueing network models with a large number of closed chains. The method is applicable to product form queueing network models with single server fixed rate, infinite server and multiple server service centers. It can also be easily extended to support the class of queue dependent service centers described in [Heff82]. Extensive empirical results demonstrate the good accuracy and cost effectiveness of the method. We have applied this technique to solve a large LOCUS type queueing network model with 41 closed chains and 85 service centers, representing a distributed system with 16 sites. The results show that the accuracy obtained is comparable to a costly solution method but with computational

requirements comparable to a cheaper, less accurate method. A critical part of the solution method is the clustering of chains and service centers to form the subnetworks. The clustering is natural for the LOCUS type of distributed systems. For general networks we developed an efficient heuristic search algorithm to perform the clustering automatically. An important feature of the approach is the ability to combine different solution techniques due to the simple physical interpretation obtained. In particular we have shown an example which combines an analytical solution for a subnetwork representing the Ethernet channel in a LOCUS type of model. Furthermore, by preserving the MVA recursion for the solution of a subnetwork, MVA based approximations for non-product form networks (e.g., priority queues, FCFS multiple server service centers with different service times for different types of customers, as proposed in chapter 4) can be easily incorporated. Needless to say that validation experiments must be done to evaluate the accuracy of the possible extensions.

We have used a clustering technique to solve the class of non-product form models with simultaneous resource possession. For the dedicated resource case empirical results have shown that the accuracy is comparable to other approaches proposed in the literature. For the more important case of shared passive resources, comparison with simulation shows that the accuracy as well the efficiency of the algorithm remains robust. We have presented an example which applies the algorithm in the case of nested passive resources, when the release of passive resource is done in the reverse order of acquisition. We have not solved the general case of nested transactions when the release of passive resources is done in an arbitrary order. Further investigation is necessary.

We have done extensive tests to evaluate the accuracy of the clustering approximation for product and non-product networks with simultaneous resource possession. However, we did not evaluate the accuracy obtained when several non-product form features are combined into a distributed system model. Although the level of difficulty in a rigorous evaluation procedure would increase with the complexity of the model, it is extremely important to have a better understanding of the type of errors that will be encountered in practice.

We have investigated the load balancing problem in a distributed system environment where multiple classes of jobs are present and restrictions may apply in the remote dispatcher. We have proposed an exact algorithm to find the optimum load balancing strategy in a LOCUS type of model. We have also extended the approach to handle jobs with multiple tasks as well, where each task is executed in sequence and may be restricted to run in a subset of all the sites in the network. The definition of the delay function in equation (5.3.1) is suitable for the study of the behavior of the network under different minimization requirements. In particular, it would be interesting to study the effect of the weights attributed to different classes of jobs in the optimum load balance strategy. Furthermore, the same approach could be used to minimize other objective functions. As we have already indicated, the algorithm of chapter 5 and 6 could be easily extended to handle queueing networks with the type of load dependent centers studied by Heffes [Heff82]. Another interesting problem (which apparently requires a more elaborate extension in the basic algorithm) is the investigation of the optimum load balancing strategy when batch jobs are generated by interactive users in a distributed environment, as indicated in section 6.4. As a byproduct of the development of chapter 5 and 6 we have shown that the variances and covariances (and larger moments) of queue lengths in a

queueing network model can be easily obtained directly from the MVA equation.

In summary, the theoretical performance study of distributed systems is in the early stages and we hope that the tools developed during this research facilitate the better understanding of these systems. We believe that it is important to develop queueing packages which include several approximation strategies to allow the analyst to get an idea of the behavior of the system being studied and to explore large design spaces. Therefore, it is important to continue the research in the area of approximation techniques for queueing networks in order to solve more complex models analytically. Finally, several load balancing problems in distributed systems remain unsolved. In particular approximations should be developed to allow the analysis of non-product form networks. The algorithms we developed may prove to be useful in this direction. Furthermore, much has to be learned in the area of distributed algorithms. A promising approach can be found in [Gafn84], where a distributed routing algorithm was proposed which combines the results of [Koba83] and [Gall77]. It would be interesting to investigate if a distributed load balancing algorithm could be developed by using this approach and the results of chapters 5 and 6.

APPENDIX 1

PROOF OF ASSUMPTION 3.2b.

In this appendix we prove assumption 3.2b. We do that by showing that for any population vector \bar{N} the average queue length of center $j \in T$ is not altered when a chain $i \in LC(T)$ is increased by one.

We assume that there is only one center per site and that N is bounded. The proof is by induction on $|\bar{N}|$. We will show that the above is true when $|\bar{N}| = 1$ and if our hypothesis is true when $|\bar{N}| = s$ it is also valid when $|\bar{N}| = s+1$. For $|\bar{N}| = 1$

$$L_j(\bar{0} + \bar{e}_i) = \lambda_i(\bar{0} + \bar{e}_i) a_{ij} \quad j \in T, i \in LC(T)$$

and from (2.1)

$$= \lambda_i(\bar{0} + \bar{e}_i) \frac{a_j}{M-1}$$

$$\lambda_i(\bar{0} + \bar{e}_i) = \frac{1}{a_{ii} + a_j} = \text{finite} \quad i \in T$$

Therefore, $L_j(\bar{0} + \bar{e}_i) = O(M)$, i.e., $L_j(\bar{0} + \bar{e}_i)$ is not altered in the limit as $M \rightarrow \infty$. By the symmetry of the problem (statements 1 and 2), this is valid for all populations so that $|\bar{N}| = 1$.

Now, by induction hypothesis, the assumption is true for all populations so that $|\bar{N}| = s$. Let us increase by one the number of customers of a chain $i \in LC(T)$. Therefore,

$$\lambda_c(\bar{N} + \bar{e}_i) = \frac{N_c}{s_{ci}[1 + L_i(\bar{N} + \bar{e}_i - \bar{e}_c)] + \sum_{j \neq i} \frac{e_j}{M-1} [1 + L_j(\bar{N} + \bar{e}_i - \bar{e}_c)] + \frac{e_f}{M-1} [1 + L_f(\bar{N} + \bar{e}_i - \bar{e}_c)]}$$

$i \in C, c \in LC(C), i \in T, i \in LC(T), C \neq T$

But $|\bar{N} + \bar{e}_i - \bar{e}_c| = e$ so, by induction hypothesis,

$$L_i(\bar{N} + \bar{e}_i - \bar{e}_c) = L_i(\bar{N} - \bar{e}_c) + O(M)$$

$$L_j(\bar{N} + \bar{e}_i - \bar{e}_c) = L_j(\bar{N} - \bar{e}_c) + O(M) \quad \forall j \neq i$$

Therefore,

$$\begin{aligned} \lambda_c(\bar{N} + \bar{e}_i) &= \frac{N_c}{s_{ci}[1 + L_i(\bar{N} - \bar{e}_c) + O(M)] + \sum_{j \neq i} \frac{e_j}{M-1} [1 + L_j(\bar{N} - \bar{e}_c) + O(M)] + O(M)} \quad c \neq LC(T) \\ &= \frac{N_c}{\sum_j s_{cj} [1 + L_j(\bar{N} - \bar{e}_c)] + O(M) + \frac{e_f(M-2)}{M-1} O(M) + O(M^2)} \\ &= \frac{N_c}{\sum_j s_{cj} W_{cj}(\bar{N}) + O(M)} \rightarrow \lambda_c(\bar{N}) \end{aligned} \quad (A1.1)$$

From equations (2.2) and (2.3)

$$L_j(\bar{N} + \bar{e}_i) = \sum_{c \neq i} \lambda_c(\bar{N} + \bar{e}_i) s_{cj} [1 + L_j(\bar{N} + \bar{e}_i - \bar{e}_c)] + \lambda_i(\bar{N} + \bar{e}_i) s_{ij} [1 + L_j(\bar{N})] \quad i \in LC(T), j \in T$$

By induction hypothesis and by (A1.1) above,

$$= \sum_{c \neq i} \frac{N_c}{\sum_{\alpha} s_{c\alpha} W_{c\alpha}(\bar{N}) + O(M)} s_{cj} [1 + L_j(\bar{N} - \bar{e}_c) + O(M)] + \frac{N_i + 1}{\sum_{\alpha} s_{c\alpha} W_{c\alpha}(\bar{N} + \bar{e}_i)} \frac{e_f}{M-1} [1 + L_j(\bar{N})]$$

$$= \sum_i \lambda_i(\bar{N}) a_{ij} [1 + L_i(\bar{N} - \bar{c}_i)] + O(M) \rightarrow L_i(\bar{N}) \quad (\text{A1.2})$$

Equation (A1.2) is valid for any population vector \bar{N} . Thus, Assumption 3.2.b is proved.

In the proof above we implicitly assumed that $L_i(\bar{N})$ is bounded for any population vector \bar{N} , i.e., $L_i(\bar{N})$ does not grow with M . We now prove that this is indeed true. The proof is by contradiction. If we assume that $L_i(\bar{N})$ is not bounded that means that center i is the bottleneck in the network. Therefore, we have:

$$\lambda_i(\bar{N}) = \frac{N_j}{a_{ij} \sum_c N_c} = \frac{N_j(M-1)}{a_{ij} \sum_c N_c} \quad j \in FC(i), i \in I$$

$$\lambda_i(\bar{N}) = \frac{N_i}{a_{ih} \sum_c N_c} \quad i \in LC(i), i \in I$$

This is true since, if center i is the bottleneck, the capacity of this center will be divided proportionally to the customers of each chain and their service demands.

(Note that if $a_j = a_h = \theta z$, $i \in LC(i)$, $\lambda_i = \sum_j \lambda_j a_{ij} = \nu z = \mu$.)

The utilization of a center $j \in J \neq I$ will be:

$$U_j(\bar{N}) = \sum_{\substack{l \in FC(j) \\ l \in LC(j)}} \lambda_l(\bar{N}) a_{lj} + \lambda_i(\bar{N}) a_{ij} + \lambda_h(\bar{N}) a_{hj} \quad h \in LC(j), l \in LC(i)$$

$$= \frac{(M-1) \sum_{l \neq i, h} N_l}{a_{ij} \sum_c N_c} \frac{a_{ij}}{M-1} + \frac{N_i}{a_{ih} \sum_c N_c} \frac{a_{ij}}{M-1} + \frac{N_h}{a_{hj} \sum_c N_c} a_{hj} (M-1)$$

$$= 1 - \frac{N_i}{\sum_c N_c} - \frac{N_h}{\sum_c N_c} + \frac{N_i}{a_{ih} \sum_c N_c} \frac{a_{ij}}{M-1} + \frac{N_h}{a_{hj} \sum_c N_c} a_{hj} (M-1)$$

$$= 1 - \frac{N_i}{\sum_i N_i} \left(1 - \frac{af}{a_{ii}(M-1)} \right) + \frac{N_i}{\sum_i N_i} \left(\frac{a_{ii}}{af}(M-1) - 1 \right)$$

$$\geq 1 + C_i - O(M)$$

where

$$C_i = \frac{N_i a_{ii}}{N_{\max} af}$$

and $N_{\max} = \max(N_1, \dots, N_M)$. Therefore, $U_i(\vec{N}) \rightarrow (1 + C_i) > 1$ as $M \rightarrow \infty$, which implies that center i can not be the bottleneck (for any i), otherwise all other centers would have average utilization greater than one. So, none of the centers can be the bottleneck of the network and $L_i(\vec{N})$ is finite $\forall i$.

APPENDIX 2
THE CLUSTERING ALGORITHM.

We describe the heuristic search algorithm developed to divide a network into subnetworks and choose the local chains for each subnetwork. The algorithm is based on the requirements presented in section 3.3.4. However, any other algorithm satisfying those requirements should work. The algorithm uses the functions $UV_k(S)$ and $LU_k(S)$ defined in section 3.3.4. Two main subsets are also used during the algorithm:

NAG = subset of all centers not yet assigned to any subnetwork and all chains not yet assigned to be local to any subnetwork.
AG = subset of all chains and centers already assigned.

The the algorithm searches for a subnetwork so that:

1. At least one new local chain is present.
2. At least one critical center not yet used is present.

Initially NAG is set to contain all centers and chains and PRCS (present number of local chains per subnetwork) is set to the initial number of local chains per subnetwork. Below are the details of the algorithm in a C-like description. The words in upper case letters followed by a parenthesis represent subroutines.

MAIN()

```
{
  while ( # critical centers in NAG is > 0 ) {
    - FIND_A_SUBNETWORK_S();
    if (S is found) {
      - REMOVE_UN_CHAINS();
      - update NAG and AG;
    }
  }
  - Remove any subnetwork covered by other.
  - Form a subnetwork containing no centers and
  whose local chains are the ones in NAG.
  - Form a subnetwork containing only centers in NAG.
}
```

FIND_A_SUBNETWORK_S()

```
{
  - Take PRCS chains from NAG which greater  $UV_H(NAG)$ 
  and store in LC(S).
  - FIND_CENTERS();

  while ( # of critical centers in S not yet present in another subnetwork == 0 ) {
    if ( # of chains in LC(S) < PRCS ) {
      - Take a new set of (PRCS - # chains in LC(S)) chains
      from the original network with greater  $UV_H(NAG)$ 
      value and store in LC(S);
      - FIND_CENTERS();
    }
    else {
      if (more than 1 chain in LC(S) have not yet been replaced &&
          REPLACE_CHAIN() == YES) {
        - FIND_CENTERS();
      }
      else {
        if (PRCS has not yet been decremented) {
          - increment PRCS;
          - return;
        }
        else {
          - "Cost is too low. Can't cluster";
          - return;
        }
      }
    }
  }

  if (cost(S) ≤ given cost) {
    - S is a new subnetwork;
    - return;
  }
  else {
    - REMOVE_UN_CHAINS();
    if (cost(S) > given cost) {
      - Remove MS service centers in S (one by one) until
      cost is satisfied or at least 1 critical MS service
      center remains (if there is any).
    }
    else S is a new subnetwork; return;
    if (cost(S) > given cost) {
      if (more than 1 chain in LC(S) has not yet been replaced) {
        - Replace a chain in LC(S) with smallest  $UV_H(S)$ 
        by a chain in the original network (which does
        not belong to LC(S)) with the highest  $UV_H(S)$ 
      }
    }
  }
}
```

AND less number of customers than the chain to be replaced. The replacement takes place only if there is at least 1 non-critical center (in the new formed subnetwork S) not yet assigned to any other subnetwork.

```

    }
  }
  else S is a new subnetwork; return;
  if (cost(S) > given cost) {
    if ( PRCS has not yet been incremented) {
      - Decrement PRCS;
      - return;
    }
    else {
      - "Cost is too low. Can't cluster";
      - return;
    }
  }
  else S is a new subnetwork; return;
}
}

```

FIND_CENTERS()

```

{
- Find all centers from the original network so that
  constraint 1 above is satisfied for the set of local
  chains of S.
  Store these centers in S.
}

```

REMOVE_UN_CHAINS()

```

{
- Remove any chain in LC(S) which is "unnecessary".
  A unnecessary chain l is the one that, if removed,
  do not alter the number of critical centers in S,
  AND

```

$$\sum_{l \in \alpha(S)} \frac{LU_l(S) - LU_l(S \text{ without local chain } l)}{LU_l(S)} < \epsilon$$

```

This last condition avoids the removal of local chains
that contribute significantly to the total utilization
of centers in S.
}

```

REPLACE_CHAIN()

```

{
- Replace a local chain of S with smallest  $UV_l(NAG)$ 
  value by a chain in the original network with
  greater  $UV_l(NAG)$  value (if possible);
  if (replacement is done)
    return(YES);
  else return(NO);
}

```

APPENDIX 3

THE CONVEXITY OF THE DELAY FUNCTION OF CHAPTER 5.

In chapter 5 section 5.5 we presented an intuitive explanation for the convexity of $D(\bar{N})$, given by equation (5.4.3), with respect to the open chain flows. In this appendix we make that argument more rigorous. From (5.4.3) we see that $D(\bar{N})$ can be written as:

$$D(\bar{N}) = \sum_S [C_1 T_S^*(N_S) + C_S R_S^*(N_S)] + C_1 T_M^*(N_N) \quad (\text{A3.1})$$

where C_i are constants, $T_S^*(N_S)$ is the total open population of site S , $R_S^*(N_S)$ is the response time of closed chain jobs at a site S and $T_M^*(N_N)$ is the total open population in the model of the communication network. Therefore, if we prove that $T_S^*(N_S)$ and $R_S^*(N_S)$ are convex with respect to the open chain flows, $D(\bar{N})$ will also be convex since it will be a sum of convex functions (*).

We start by proving that $R_S^*(N_S)$ is convex with respect to the open chain flows. For that we need the following theorem:

Theorem A3.1.

In a product form network with a single chain, the variance of queue lengths of a center j is a non decreasing function of the number of customers in this network, i.e., $V_j(N) \geq V_j(N-1) \forall N$.

(*) Since the model of the communication network is any product form queueing network model, proving that $T_S^*(N_S)$ is convex will also prove that $T_M^*(N_N)$ is convex.

Proof:

Kobayashi and Gerla [KOBAS3] proved that the inverse of the throughput in a single chain closed queueing network is convex with respect to the visit ratios of customers in the network at a center j , θ_j . But this is equivalent to show that the second derivative of $1/\lambda(N)$ with respect to θ_j is greater or equal than zero, i.e.,

$$\frac{\partial^2 1/\lambda(N)}{(\partial \theta_j)^2} \geq 0 \quad (\text{A3.2})$$

From the results of appendix 4 it is easy to show that:

$$\frac{\partial 1/\lambda(N)}{\partial \theta_j} = \frac{1}{\theta_j \lambda(N)} [L_{j,(N)} - L_{j,(N-1)}] \quad (\text{A3.3})$$

and

$$\frac{\partial^2 1/\lambda(N)}{(\partial \theta_j)^2} = \frac{1}{\theta_j^2 \lambda(N)} \left(-[1 - (L_{j,(N)} - L_{j,(N-1)})] [L_{j,(N)} - L_{j,(N-1)}] + V_{j,(N)} - V_{j,(N-1)} \right) \quad (\text{A3.4})$$

In [REIS81] it was proven that $0 \leq L_{j,(N)} - L_{j,(N-1)} \leq 1$, so the two terms in brackets in equation (A3.4) are greater or equal to zero. Since, from (A3.2), equation (A3.4) is non negative, the difference $V_{j,(N)} - V_{j,(N-1)}$ has to be non negative for all N .

We now can prove the following.

Lemma A3.2

In a product form mixed network with a single closed chain, the response time of the closed chain jobs is convex with respect to the open chain flows.

Proof:

The response time for a site S is:

$$R_S^c(N_S) = \frac{N_S}{\lambda_S^c(N_S)} - z_{S1} \quad (\text{A3.5})$$

Taking the first and second derivatives of (A3.5) with respect to ρ_j^c , $j \in$ site S and using the results of appendix 4 yields:

(we drop the subscript S to simplify the notation.)

$$\frac{\partial R^c(N)}{\partial \rho_j^c} = \frac{N}{(1 - \rho_j^c)\lambda^c(N)} [L_j^c(N-1) - L_j^c(N)] \quad (\text{A3.6})$$

and

$$\frac{\partial^2 R^c(N)}{(\partial \rho_j^c)^2} = \frac{1}{(1 - \rho_j^c)^2 \lambda^c(N)} \left([1 + L_j^c(N) - L_j^c(N-1)][L_j^c(N) - L_j^c(N-1)] + V_j^c(N) - V_j^c(N-1) \right) \quad (\text{A3.7})$$

Since $L_j(N) \geq L_j(N-1)$, the two terms in brackets in equation (A3.7) are positive. From theorem A3.1 we have shown that, in a single chain closed network, $V_j(N) \geq V_j(N-1)$. But any mixed network with one closed chain can be reduced to an equivalent single chain closed network. Therefore, in a mixed network, it is also true that $V_j(N) \geq V_j(N-1)$ and so equation (A3.7) is non negative. Since $\partial^2 R_S^c(N_S) / (\partial \lambda_{\eta}^c)^2 = (1/\mu_{\eta}^c)^2 \partial^2 R_S^c(N_S) / (\partial \rho_j^c)^2$ the Lemma is proven.

To complete the proof of the convexity of $D(\bar{N})$ we need to prove that the total open population in a site is convex with respect to an open chain flow (λ^c). Intuitively, as the throughput of an open chain to a site increases (recall that a site is a product form queueing network model with single server fixed rate and infinite server service centers and a single closed chain) the total number of open customer should also increase with an increasing rate, and thus $T_S^c(N_S)$ should be convex. We have been able to prove that $T_S^c(N_S)$ increases with the increase of an

open chain throughput to site S , but unfortunately, we haven't been able to prove that the rate of increase is positive for a general situation. We were able to prove that the rate of increase is positive only in the very particular case when site S has only two queue independent service centers and one closed chain job. In what follows we prove that $\partial T_S^c(N_S)/\partial \lambda'$ is always positive and that $\partial^2 T_S^c(N_S)/(\partial \lambda')^2$ is positive in the particular case mentioned above. We also present plots showing that $T_S^c(N_S)$ is convex for the example of chapter 5.

Theorem A3.3.

$$\frac{\partial T_S^c(N_S)}{\partial \lambda'} \geq 0$$

where λ' is an open chain flow into site S .

Proof:

For notational convenience we assume that there is no IS center in site S . From the results of appendix 4 and observing that $\rho_j' = \lambda' a_j'$, it is easy to show that:

$$\frac{\partial L_j^c(N)}{\partial \lambda'} = \frac{a_j'}{1 - \rho_j'} V_j^c(N) + \sum_{i \neq j} \frac{a_i'}{1 - \rho_i'} CV_{j,i}^c(N) \quad (\text{A3.8})$$

where J_S is the total number of centers at site S . From [LAVE83] (we drop the subscript S to simplify the notation)

$$T^c(N) = \sum_{j=1}^{J_S} \frac{\rho_j'}{1 - \rho_j'} [1 + L_j^c(N)] \quad (\text{A3.9})$$

Taking the derivative of (A3.9) with respect to λ' and using (A3.8) and, for convenience, using the notation $CV_{j,i}^c(N) \triangleq V_{j,i}^c(N)$, $CV_{j,j}^c(N) = V_{j,j}^c(N) = V_j^c(N)$, yields:

$$\frac{\partial T^*(N)}{\partial \lambda'} = \sum_j \frac{a_j^2}{(1 - \rho_j)^2} [1 + L_j^*(N)] + \lambda' A(N) \quad (\text{A3.10a})$$

where

$$A(N) = \sum_j \sum_i \frac{a_j^2 a_i^2}{(1 - \rho_j)(1 - \rho_i)} V_{ji}^*(N) \quad (\text{A3.10b})$$

But since $V_{ji}^*(N) = V_{ij}^*(N)$,

$$A(N) = \sum_{j=1}^I \frac{(a_j^2)^2}{(1 - \rho_j)^2} V_{jj}^*(N) + 2 \sum_{j=1}^{I-1} \sum_{i=j+1}^I \frac{a_j^2 a_i^2}{(1 - \rho_j)(1 - \rho_i)} V_{ji}^*(N) \quad (\text{A3.11})$$

From equations (A4.2.3) and (A4.2.4) in appendix 4, it is easy to shown that:

$$V_{ji}^*(N) = - \sum_{i \neq j} V_{ij}^*(N) \quad (\text{A3.12})$$

Substituting (A3.12) into the first term of (A3.11), and using again the fact that

$V_{ji}^*(N) = V_{ij}^*(N)$, we obtain:

$$\begin{aligned} A(N) &= - \sum_{j=1}^{I-1} \sum_{i=j+1}^I \left(\frac{(a_j^2)^2}{(1 - \rho_j)^2} + \frac{(a_i^2)^2}{(1 - \rho_i)^2} \right) V_{ji}^*(N) + 2 \sum_{j=1}^{I-1} \sum_{i=j+1}^I \frac{a_j^2 a_i^2}{(1 - \rho_j)(1 - \rho_i)} V_{ji}^*(N) \\ &= - \sum_{j=1}^{I-1} \sum_{i=j+1}^I \left(\frac{a_j^2}{1 - \rho_j} - \frac{a_i^2}{1 - \rho_i} \right)^2 V_{ji}^*(N) \end{aligned} \quad (\text{A3.13})$$

From (A4.2.4) it is easy to shown that $V_{ji}^*(N)$ is always negative for $j \neq i$. Therefore, $A(N)$ above is always positive which implies that (A3.10a) is also positive.

For a simple queueing model with only two sites and one closed chain jobs it is easy to show that $\partial^2 T_3^*(N_S)/(\lambda')^2$ is always positive. To show that we observe that, for this simple case:

$$V_{11}^*(1) = V_{22}^*(1) = - V_{12}^*(1) = - V_{21}^*(1) = L_1^*(1) L_2^*(1) \quad (\text{A.14})$$

and

$$L_i(1) = 1 - L_i^c(1) = \frac{C}{B} a_i \quad (\text{A3.15})$$

where $B = a_1(1 - \rho_2) + a_2(1 - \rho_1)$ and $C = (1 - \rho_1)(1 - \rho_2)$

Substituting (A3.14), (A3.15) and (A3.13) into (A3.10a) and taking the derivative of $\partial^2 T_3^c(N_S)/\partial \lambda'^2$ with respect to λ' , we obtain after simple algebraic manipulations:

$$\begin{aligned} \frac{\partial^2 T_3^c(1)}{(\partial \lambda')^2} = & 2 \sum_{j=1}^2 \frac{a_j^2}{(1 - \rho_j)^2} [1 + L_j^c(1)] \\ & + \frac{a_1 a_2 (a_1 - a_2)^2}{B C} \left(\frac{(1 - \rho_1 \rho_2)}{B C} + \frac{1}{B} + \frac{2\lambda' [a_1 a_2 + a_2 a_1]}{B^2} + \frac{\lambda'}{C} \right) \end{aligned} \quad (\text{A3.16})$$

Clearly $\partial^2 T_3^c(N_S)/(\partial \lambda')^2 \geq 0$.

For the general case, we haven't been able to prove that $\partial^2 T_3^c(N_S)/(\partial \lambda')^2$ is always non negative. Therefore we choose to plot $T_3^c(N_S)$ as a function of an open chain throughput for the example of chapter 5. Figures A3.1, A3.2a and A3.2b show the results. For Figure A3.1 the site chosen is site 1 and the open chain jobs have the same service requirements of jobs a_1 . The open chain throughput was varied from 0.1 jobs/sec to 6.0 jobs/sec (in steps of 0.1). For Figures A3.2 the site chosen is site 2 and the open chain jobs are assumed to have the same service requirements as jobs a_1 . Figure A3.2a (b) shows the results when the number of closed chain jobs at site 2 is equal to 2 (5). For both Figures the open chain throughput was varied from 0.1 jobs/sec to 9.0 jobs/sec (in steps of 0.1). From the Figures it is easy to see that $T_3^c(N_S)$ is indeed a function which increases with the open chain throughput with a positive rate.

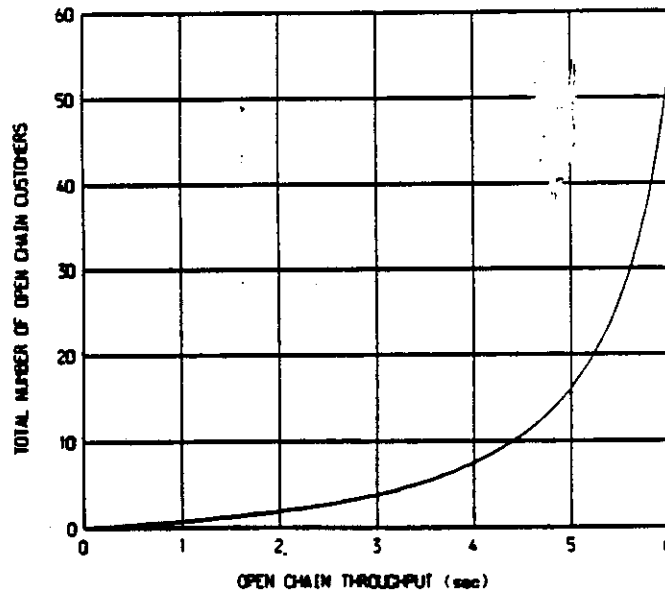
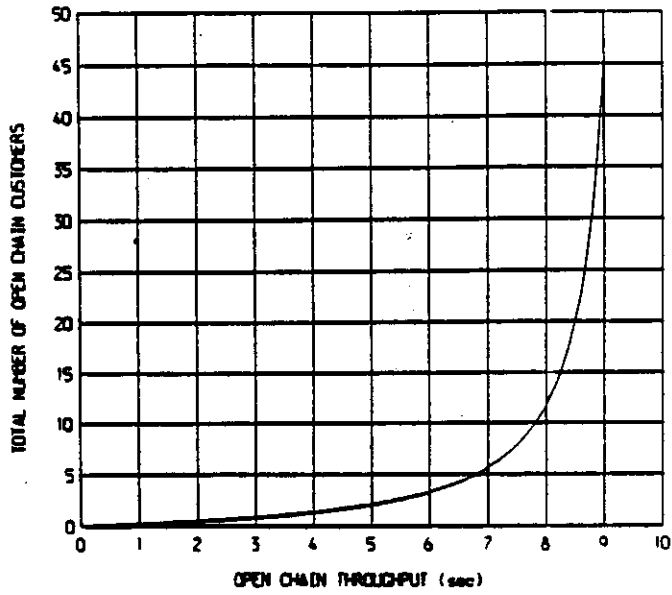
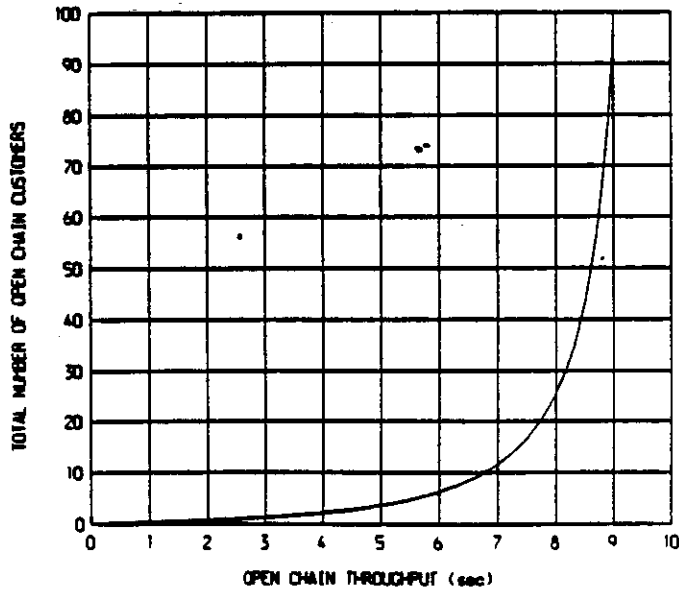


Figure A3.1. Number of open chain customers at site 1 as a function of open chain throughput



(a) Number of closed chain jobs equal to 2.



(b) Number of closed chain jobs equal to 5.

Figure A3.2. Number of open chain customers at site 2 as a function of open chain throughput

APPENDIX 4

MOMENTS OF QUEUE LENGTHS IN QUEUEING NETWORKS

A4.1. Introduction.

In this appendix we show that the moments of queue lengths of a mixed queueing network model with multiple closed chains can be easily solved recursively. Hefes [HEFF82] has already observed that the moments of queue lengths can be obtained recursively in terms of lower moments. To demonstrate that, he used the recursively formula for the marginal probability distribution given in [REIS80]. In what follows we show that the recursion for calculating the variances can be obtained directly from the MVA equations by using results similar to the ones obtained in equation (4.14) and (4.15). We first obtain a recursive equation for the variance and covariances in equation (4.18). Then we extend the approach for a closed queueing network with multiple closed chains.

A4.2. Recursive Expressions for the Variance and Covariances in Equation (4.18).

We show that the recursion for calculating the variances in equation (4.18) can be obtained directly from the MVA equations. We prove the result for single closed chain networks, but the same approach can be used for multiple chain networks.

The well known MVA equation which relates the mean queue lengths of closed chain jobs at a service center as a function of queue lengths of closed chain jobs with one less closed chain job in the network becomes, for mixed networks:

$$L_j(N) = \lambda(N) \frac{e_j}{1 - \rho_j^e} [1 + L_j(N-1)] \quad (\text{A4.2.1})$$

where the superscript e was dropped to simplify the notation, and j is a PS or FCFS service center.

Taking the derivative on both sides of equation (A4.2.1) with respect to ρ_j^e gives:

$$\frac{\partial L_j(N)}{\partial \rho_j^e} = \frac{\partial \lambda(N)}{\partial \rho_j^e} \frac{e_j}{1 - \rho_j^e} [1 + L_j(N-1)] + \frac{\lambda(N) e_j}{(1 - \rho_j^e)^2} [1 + L_j(N-1)] + \frac{\lambda(N) e_j}{1 - \rho_j^e} \frac{\partial L_j(N-1)}{\partial \rho_j^e} \quad (\text{A4.2.2})$$

Now, using (4.14) and (4.17) in (A4.2.2):

$$V_j(N) = \lambda(N) \frac{e_j}{1 - \rho_j^e} \left([1 + L_j(N-1) - L_j(N)] [1 + L_j(N-1)] + V_j(N-1) \right) \quad (\text{A4.2.3})$$

Similarly an expression for the covariance $V_{ij}(N)$ can be obtained:

$$V_{ij}(N) = \lambda(N) \frac{e_i}{1 - \rho_i^e} \left([L_j(N-1) - L_j(N)] [1 + L_i(N-1)] + V_{ij}(N-1) \right) \quad i \neq j \quad (\text{A4.2.4})$$

where $V_j(0) = V_{ij}(0) \triangleq 0$.

Clearly, equations (A4.2.3) and (A4.2.4) can be solved recursively in N .

A4.3. Moments of Queue Lengths in a Queueing Network with Multiple Closed Chains.

We start by proving the following theorems.

Theorem A4.1.

In a product form network, the variance of the number of closed chain k customers at center j is related with the partial derivative of the average queue lengths of chain k customers at the same center with respect to the visit ratios of these customers at that center:

$$V_{kj}(\bar{N}) = \theta_{kj} \frac{\partial L_{kj}(\bar{N})}{\partial \theta_{kj}} \quad (\text{A4.3.1})$$

Proof:

We assume that the network is formed by single server fixed rate service centers and IS service centers. The equilibrium probability of state $S \in \{n_u\}$ so that $\sum_{i=1}^J n_{ui} = N_i, \forall i$, is

$$P(S) = \frac{1}{G(\bar{N})} \prod_{i=1}^J P_i(\bar{n}_i) \quad \bar{n}_i = (n_{1i}, \dots, n_{ki})$$

where

$$P_i(\bar{n}_i) = n_i! \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \quad l \neq \text{IS service center}$$

$$= \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \quad l = \text{IS service center}$$

$n_i = |\bar{n}_i|$ and $G(\bar{N})$ is the normalization constant of the network, i.e.,

$$\sum_S P(S) = 1 \text{ and so } G(\bar{N}) = \sum_S \prod_{i=1}^I P_i(\bar{n}_i)$$

Let the first J_1 centers be single server fixed rate service centers and the remaining IS service centers. Taking the derivative of $G(\bar{N})$ with respect to the visit ratio of chain k customers at center j gives:

$$\frac{\partial G(\bar{N})}{\partial \theta_j} = \sum_S \frac{n_{kj}}{\theta_j} \left(\prod_{i=1}^{J_1} n_i \prod_{i=1}^K (a_{ij})^{n_i} \right) \cdot \left(\prod_{i=J_1+1}^I \prod_{i=1}^K (a_{ij})^{n_i} \right) \quad (\text{A4.3.2})$$

$$= \frac{G(\bar{N})}{\theta_j} \sum_S n_{kj} P(S)$$

$$= \frac{G(\bar{N})}{\theta_j} L_{kj}(\bar{N}) \quad (\text{A4.3.3})$$

by definition of the average queue length of a chain k customer at center j . Equation (A4.3.3) was first obtained by Kobayashi and Gerla [KOBAS3]. Taking the derivative of (A4.3.2) with respect to θ_j gives:

$$\frac{\partial^2 G(\bar{N})}{(\partial \theta_j)^2} = \frac{G(\bar{N})}{\theta_j^2} \sum_S n_{kj}^2 P(S) - \frac{G(\bar{N})}{\theta_j^2} \sum_S n_{kj} P(S)$$

$$= \frac{G(\bar{N})}{\theta_j^2} \{S_{kj}(\bar{N}) - L_{kj}(\bar{N})\} \quad (\text{A4.3.4})$$

where the second equality is obtained from the definition of the second moment of queue lengths of a chain k customer at center j ($S_{kj}(\bar{N})$). Since equations (A4.3.2) and (A4.3.3) are identities, $\partial^2 G(\bar{N}) / (\partial \theta_j)^2$ can also be obtained by taking the derivative of (A4.3.3) with respect to θ_j .

$$\frac{\partial^2 G(\bar{N})}{(\partial \theta_j)^2} = \frac{\frac{\partial (G(\bar{N}) L_{kj}(\bar{N}))}{\partial \theta_j} \theta_j - G(\bar{N}) L_{kj}(\bar{N})}{\theta_j^2}$$

$$\begin{aligned}
&= \frac{\theta_{kj} \frac{\partial G(\bar{N})}{\partial \theta_{kj}} L_{kj}(\bar{N}) + \theta_{kj} G(\bar{N}) \frac{\partial L_{kj}(\bar{N})}{\partial \theta_{kj}} - G(\bar{N}) L_{kj}(\bar{N})}{\theta_{kj}^2} \\
&= \frac{G(\bar{N})}{\theta_{kj}^2} \left(L_{kj}^2(\bar{N}) + \theta_{kj} \frac{\partial L_{kj}(\bar{N})}{\partial \theta_{kj}} - L_{kj}(\bar{N}) \right) \tag{A4.3.5}
\end{aligned}$$

where the last equality was obtained by using (A4.3.3). Now equating (A4.3.4) and (A4.3.5) and using the definition of the variance of queue lengths of chain k customers at center j , $V_{kj}(\bar{N})$ (A4.3.1) is obtained.

Theorem A4.2.

For a product form network the covariance of queue lengths of chain k and chain v customers at center j is related with the partial derivative of the average queue lengths of chain k (or v) customers at the same center, with respect to the visit ratios of these customers at that center:

$$CV_{kv}(\bar{N}) = \theta_{vj} \frac{\partial L_{kj}(\bar{N})}{\partial \theta_{vj}} \tag{A4.3.6a}$$

$$= \theta_{kj} \frac{\partial L_{vj}(\bar{N})}{\partial \theta_{kj}} \tag{A4.3.6b}$$

Proof:

We first prove equation (A4.3.6a). Taking the derivative of (A4.3.2) with respect to θ_{vj} gives:

$$\frac{\partial^2 G(\bar{N})}{\partial \theta_{kj} \partial \theta_{vj}} = \frac{G(\bar{N})}{\theta_{kj} \theta_{vj}} \sum_s n_{kj} n_{vj} P(S)$$

$$= \frac{G(\bar{N})}{\theta_k \theta_v} J_{kvj}(\bar{N}) \quad (\text{A4.3.7})$$

where the second equality is obtained from the definition of the joint moment of queue lengths of chain k and chain v customers at center j . Taking the derivative of (A4.3.3) with respect to θ_v , we obtain:

$$\begin{aligned} \frac{\partial^2 G(\bar{N})}{\partial \theta_k \partial \theta_v} &= \frac{1}{\theta_k} \left(\frac{\partial G(\bar{N})}{\partial \theta_v} L_{kj}(\bar{N}) + G(\bar{N}) \frac{\partial L_{kj}(\bar{N})}{\partial \theta_v} \right) \\ &= \frac{G(\bar{N})}{\theta_k \theta_v} \left(L_{vj}(\bar{N}) L_{kj}(\bar{N}) + \theta_v \frac{\partial L_{kj}(\bar{N})}{\partial \theta_v} \right) \end{aligned} \quad (\text{A4.3.8})$$

Equation (A.3.6a) is obtained by equating (A4.3.7) and (A4.3.8) and by the definition of the covariance of queue lengths of chain k and chain v customers at center j , i.e.,

$$CV_{kvj}(\bar{N}) = J_{kvj}(\bar{N}) - L_{kj}(\bar{N}) L_{vj}(\bar{N})$$

Equation (A4.3.6b) can be easily obtained by following the same steps above, or simply by exchanging indexes k and v in (A4.3.6a) and noting that $CV_{kvj}(\bar{N}) = CV_{vkj}(\bar{N})$

A recursively formula for the variance of queue lengths of chain k customers at center j ($V_{kj}(\bar{N})$) and covariance of queue lengths of chain k and chain v customers at center j ($CV_{kvj}(\bar{N})$) can be easily obtained from theorems (A4.1) and (A4.2) above and the MVA equations as shown by the following Lemma.

Lemma A4.3.

$$V_{kj}(\bar{N}) = [1 + L_{vj}(\bar{N} - \bar{e}_k) - L_{vj}(\bar{N})] L_{kj}(\bar{N}) + \lambda_k(\bar{N}) s_{kj} [V_{vj}(\bar{N} - \bar{e}_k) + \sum_{i \neq k} CV_{kvj}(\bar{N} - \bar{e}_i)] \quad j \neq IS \quad (\text{A4.3.9a})$$

$$= \lambda_k(\bar{N}) a_{kj} [1 + L_{kj}(\bar{N} - \bar{c}_k) - L_{kj}(\bar{N})] \quad j = IS \quad (A4.3.9b)$$

$$CV_{kj}(\bar{N}) = [L_{kj}(\bar{N} - \bar{c}_k) - L_{kj}(\bar{N})] L_j(\bar{N}) + \lambda_k(\bar{N}) a_{kj} [V_{kj}(\bar{N} - \bar{c}_k) + \sum_{i \neq k} CV_{ij}(\bar{N} - \bar{c}_i)] \quad i \neq k, j \neq IS \quad (A4.3.10)$$

$$V_{kj}(\bar{0}) \triangleq 0 \quad CV_{kj}(\bar{0}) \triangleq 0$$

Proof:

For single server fixed rate service centers the average number of chain k jobs at center j is given by the MVA equation (2.2) and (2.3a), i.e.,

$$L_{kj}(\bar{N}) = \lambda_k(\bar{N}) a_{kj} [1 + \sum_i L_{ij}(\bar{N} - \bar{c}_i)] \quad (A4.3.11)$$

Taking the derivative of the above equation with respect to θ_{kj} gives:

$$\frac{\partial L_{kj}(\bar{N})}{\partial \theta_{kj}} = \frac{\partial \lambda_k(\bar{N})}{\partial \theta_{kj}} a_{kj} [1 + L_{kj}(\bar{N} - \bar{c}_k)] + \lambda_k(\bar{N}) a_{kj} [1 + L_{kj}(\bar{N} - \bar{c}_k)] + \lambda_k(\bar{N}) a_{kj} \sum_i \frac{\partial L_{ij}(\bar{N} - \bar{c}_i)}{\partial \theta_{kj}} \quad (A4.3.12)$$

The derivative of $\lambda_k(\bar{N})$ with respect to θ_{kj} can be easily obtained by noting that

$$\lambda_k(\bar{N}) = \frac{G(\bar{N} - \bar{c}_k)}{G(\bar{N})}$$

and thus

$$\begin{aligned} \frac{\partial \lambda_k(\bar{N})}{\partial \theta_{kj}} &= \frac{G(\bar{N} - \bar{c}_k)}{\theta_{kj} G(\bar{N})} L_{kj}(\bar{N} - \bar{c}_k) - \frac{G(\bar{N} - \bar{c}_k)}{G^2(\bar{N})} \frac{G(\bar{N})}{\theta_{kj}} L_{kj}(\bar{N}) \\ &= \frac{\lambda_k(\bar{N})}{\theta_{kj}} [L_{kj}(\bar{N} - \bar{c}_k) - L_{kj}(\bar{N})] \end{aligned} \quad (A4.3.13)$$

Using (A4.3.13) in (A4.3.12) and using theorems (A4.1) and (A4.2) yields:

$$\begin{aligned} \frac{V_{hj}(\bar{N})}{\theta_{hj}} &= \frac{\lambda_H(\bar{N})}{\theta_{hj}} [L_{hj}(\bar{N}-\bar{c}_k) - L_{hj}(\bar{N})] a_{hj} [1 + L_j(\bar{N}-\bar{c}_k)] + \frac{\lambda_H(\bar{N}) a_{hj}}{\theta_{hj}} [1 + L_j(\bar{N}-\bar{c}_k)] \\ &+ \frac{\lambda_H(\bar{N}) a_{hj}}{\theta_{hj}} [V_{hj}(\bar{N}-\bar{c}_k) + \sum_{i \neq k} CV_{hj}(\bar{N}-\bar{c}_k)] \end{aligned} \quad (\text{A4.3.14})$$

Finally, equation (A4.3.9a) is obtained by substituting (A4.3.11) into (A4.3.14).

For IS service centers the average number of chain k customers at center j is given by the MVA equations (2.2) and (2.3c),

$$L_{hj}(\bar{N}) = \lambda_H(\bar{N}) a_{hj} \quad (\text{A4.3.15})$$

Taking the derivative of (A4.3.15) with respect to θ_{hj} and using (A4.3.13) and theorem (A4.1) yields (A4.3.9b).

To prove (A4.3.10) we take the derivative of (A4.3.11) with respect to θ_{hj} , $i \neq k$:

$$\frac{\partial L_{hj}(\bar{N})}{\partial \theta_{hj}} = \frac{\partial \lambda_H(\bar{N})}{\partial \theta_{hj}} a_{hj} [1 + L_j(\bar{N}-\bar{c}_k)] + \lambda_H(\bar{N}) a_{hj} \sum_i \frac{\partial L_{ij}(\bar{N}-\bar{c}_k)}{\partial \theta_{hj}} \quad (\text{A4.3.16})$$

$\partial \lambda_H(\bar{N}) / \partial \theta_{hj}$ can be easily obtained in the same way equation (A4.3.13) was obtained:

$$\frac{\partial \lambda_H(\bar{N})}{\partial \theta_{hj}} = \frac{\lambda_H(\bar{N})}{\theta_{hj}} [L_{hj}(\bar{N}-\bar{c}_k) - L_{hj}(\bar{N})] \quad (\text{A4.3.17})$$

Substituting (A4.3.17) in (A4.3.16) and using theorem (A4.2) we get (A4.3.10).

Next we find a recursive formula for the variance of the total queue length at center j . First we need the following theorem.

Theorem A4.4.

For a product form network the variance of the total queue length at a center j is related to the partial derivative of the total mean queue length with respect to the capacity of that center:

$$V_j(\bar{N}) = -c_j \frac{\partial L_j(\bar{N})}{\partial c_j} \quad (\text{A4.3.18})$$

where c_j is defined to be the capacity of center j .

Proof:

Up to now we have assumed that the capacity of all service centers in the network were equal to one. In general, for $c_j \neq 1$, the equilibrium probability of a state S ($P(S)$) is given by

$$P(S) = \frac{1}{G(\bar{N})} \prod_{i=1}^J P_i(\bar{n}_i)$$

where

$$P_i(\bar{n}_i) = \frac{n_i!}{c_i^{n_i}} \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \quad i \neq \text{IS service center}$$

$$= \frac{1}{c_i^{n_i}} \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \quad i = \text{IS service center}$$

$$\text{and } G(\bar{N}) = \sum_S \prod_{i=1}^J P_{\text{sub}}(\bar{n}_i).$$

The proof follows similar steps used in the proof of theorem A4.1. Taking the derivative of $G(\bar{N})$ with respect to the capacity of center j gives:

$$\frac{\partial G(\bar{N})}{\partial c_j} = -\sum_S \frac{n_j}{c_j} \left(\prod_{i=1}^{j-1} \frac{n_i!}{c_i^{n_i}} \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \right) \cdot \left(\prod_{i=j+1}^J \frac{1}{c_i^{n_i}} \prod_{l=1}^K \frac{(a_{il})^{n_{il}}}{n_{il}!} \right) \quad (\text{A.3.19})$$

$$\begin{aligned}
&= -\sum_j \frac{n_j}{c_j} P(S) G(\bar{N}) \\
&= -\frac{G(\bar{N})}{c_j} L_j(\bar{N})
\end{aligned} \tag{A4.3.20}$$

where the last equality is obtained by the definition of the total average queue length at center j .

Taking the derivative of (A4.3.19) with respect to c_j gives:

$$\begin{aligned}
\frac{\partial^2 G(\bar{N})}{(\partial c_j)^2} &= \sum_j \frac{n_j^2}{c_j^2} P(S) G(\bar{N}) + \sum_j \frac{n_j}{c_j^2} P(S) G(\bar{N}) + \\
&= \frac{G(\bar{N})}{c_j^2} \{S_j(\bar{N}) + L_j(\bar{N})\}
\end{aligned} \tag{A4.3.21}$$

where, again, the second equality is obtained by the definition of the second moment of total queue lengths at center j ($S_j(\bar{N})$). Taking the derivative of (A4.3.20) with respect to c_j gives:

$$\begin{aligned}
\frac{\partial^2 G(\bar{N})}{(\partial c_j)^2} &= -\frac{\frac{\partial G(\bar{N})}{\partial c_j} L_j(\bar{N}) + G(\bar{N}) \frac{\partial L_j(\bar{N})}{\partial c_j}}{c_j} + \frac{G(\bar{N}) L_j(\bar{N})}{c_j^2} \\
&= \frac{G(\bar{N})}{c_j^2} \left(L_j^2(\bar{N}) - c_j \frac{\partial L_j(\bar{N})}{\partial c_j} + L_j(\bar{N}) \right)
\end{aligned} \tag{A4.3.22}$$

Equating (A4.3.20) and (A4.3.22), (A4.3.18) is obtained.

A recursively formula for $V_j(\bar{N})$ can be easily obtained from the theorem above and the MVA equations (2.2), (2.3a) and (2.3c) as shown by Lemma A4.5 below.

Lemma A4.4.

$$V_j(\bar{N}) = \sum_{i=1}^K \left([1 + L_j(\bar{N}-\bar{e}_i) - L_j(\bar{N})] L_{ij}(\bar{N}) + \lambda_H(\bar{N}) \sigma'_{ij} V_j(\bar{N}-\bar{e}_i) \right) \quad j \neq \text{IS center} \quad (\text{A4.3.23a})$$

$$= \sum_{i=1}^K \lambda_H(\bar{N}) \sigma'_{ij} [1 + L_j(\bar{N}-\bar{e}_i) - L_j(\bar{N})] \quad j = \text{IS center} \quad (\text{A4.3.23b})$$

where $\sigma'_{ij} = \sigma_{ij}/c_j$.

Proof:

The proof is analogous to the proof of Lemma A4.3. For single server fixed rate service centers the MVA equation for the total average queue length is:

$$L_j(\bar{N}) = \sum_{i=1}^K \lambda_H(\bar{N}) \sigma'_{ij} [1 + L_j(\bar{N}-\bar{e}_i)] \quad (\text{A4.3.24})$$

Taking the derivative of (A4.3.24) with respect to c_j gives:

$$\frac{\partial L_j(\bar{N})}{\partial c_j} = \sum_{i=1}^K \left(\frac{\partial \lambda_H(\bar{N})}{\partial c_j} \sigma'_{ij} [1 + L_j(\bar{N}-\bar{e}_i)] - \frac{\lambda_H(\bar{N})}{c_j} \sigma'_{ij} [1 + L_j(\bar{N}-\bar{e}_i)] + \lambda_H(\bar{N}) \sigma'_{ij} \frac{\partial L_j(\bar{N}-\bar{e}_i)}{\partial c_j} \right) \quad (\text{A4.3.25})$$

$\partial \lambda_H(\bar{N})/\partial c_j$ can be easily obtained in the same manner as in the previous lemmas, i.e., by taking the derivative of $G(\bar{N}-\bar{e}_i)/G(\bar{N})$ with respect to c_j . We obtain:

$$\frac{\partial \lambda_H(\bar{N})}{\partial c_j} = \frac{\lambda_H(\bar{N})}{c_j} [L_j(\bar{N}) - L_j(\bar{N}-\bar{e}_i)] \quad (\text{A4.3.26})$$

Substituting (A4.3.26) in (A4.3.25) and using theorem A4.4 gives (A4.3.23a).

For IS service centers the MVA equation for the total average queue length is:

$$L_j(\bar{N}) = \sum_{k=1}^K \lambda_k(\bar{N}) a'_{kj} \quad (\text{A4.3.27})$$

Taking the derivative of (A4.3.27) with respect to ϵ , and using (A4.3.26) and theorem A4.4, (A4.3.23b) is obtained.

The same method used above to obtain the variances and covariances of queue lengths can be used to obtain higher moments as well. In particular we outline the development for obtaining the third moment of queue lengths at a center j ($T_j(N)$) for a single chain network. However, the extension for multiple chain networks is straightforward.

Theorem A4.6.

For a product form single chain network, the third moment of queue lengths at a center j ($T_j(N)$) is related to the partial derivative of the second moment of queue lengths with respect to the visit ratios of customers at that center:

$$T_j(N) - S_j(N)L_j(N) = \frac{\partial S_j(N)}{\partial \theta_j} \quad (\text{A4.3.28})$$

Proof:

Analogously to what was done in the proof of the previous theorems, the third derivative of $G(N)$ with respect to θ_j , gives:

$$\frac{\partial^3 G(N)}{(\partial \theta_j)^3} = \frac{G(N)}{\theta_j^3} \{T_j(N) - 3S_j^2(N) + 2L_j(N)\} \quad (\text{A4.3.29})$$

$\partial^3 G(N)/\partial(\theta_j)^3$ can also be obtained by taking the derivative of equation (A4.3.4) (simplified for single closed chain networks) with respect to θ_j :

$$\frac{\partial^2 G(N)}{(\partial \theta)^2} = \frac{G(N)}{\theta^2} \left(V_{j,(N)}[3L_{j,(N)} - 3] + L_{j,(N)}^2[L_{j,(N)} - 3] + 2L_{j,(N)} + \theta \frac{\partial V_{j,(N)}}{\partial \theta} \right) \quad (\text{A4.3.30})$$

Equating (A4.3.29) and (A4.3.30) and using our previous results yields (A4.3.28).

A recursively formula for $T_{j,(N)}$ can be obtained by taking the derivative of (A4.3.23a) or (A4.3.23b) with respect to θ , and using theorem A4.6.

A4.4. Conclusions.

In this appendix we have shown that, for product form networks, moments of queue lengths of a service center are related with the partial derivative of lower moments with respect to a parameter (e.g., visit ratios, capacity, etc) of that center. This relation is important since optimization problems may require to obtain the derivative of queue lengths with respect to an input parameter (as in chapter 4) or even the derivative of higher moments. The relation obtained also provides an easy method to obtain a recursively formula for moments of queue lengths by simply taking the derivative of the main MVA equation.

The same relation could be heuristically used, in principle, to calculate moments of queue lengths in a non-product form network. For instance, for networks with FCFS service centers with different chains, a recursively formula for the variance of queue lengths could be heuristically obtained by taking the derivative of the approximate formula obtained by Reiser [REIS79] (see equation (2.5b)) with respect to an input parameter of the center. Needless to say that, in this case, validation studies must be done.

Finally it is interesting to point out that the theorems proved in this appendix provide a method of proving some important relations in queueing networks. In particular, we shown three examples.

(1) We show that the following well known relationship for average queue lengths of open and closed chain jobs in a mixed queueing network,

$$L_j(\bar{N}) = \frac{\rho_j}{1 - \rho_j} [1 + L_j^c(\bar{N})]$$

which has previously been derived from the arrival theorem, can be easily obtained by taking the derivatives of the normalization constant:

The normalization constant for a mixed network, $G(\bar{N})$ is given by:

$$G(\bar{N}) = \sum_{s=1}^J \prod_{i=1}^s \frac{(n_i + n_i^c)!}{n_i!} (\rho_i)^{n_i} \prod_{i=1}^K \frac{(a_i^c)^{n_i^c}}{n_i^c!} \quad (\text{A4.4.1})$$

where the superscript c or o indicates "open" or "closed" chain, respectively, and we assume that all center are single server fixed rate to simplify the notation. It is known that $G(\bar{N})$ can also be obtained from the normalization constant of an equivalent closed network, $G^c(\bar{N})$ [LAVE83] (this can be easily shown by a simple manipulation of equation (A4.4.1)):

$$G(\bar{N}) = \left(\prod_{i=1}^J \frac{1}{1 - \rho_i} \right) G^c(\bar{N}) = G^o G^c(\bar{N}) \quad (\text{A4.4.2})$$

where

$$G^o(\bar{N}) = \sum_{s=1}^J \prod_{i=1}^s n_i^o \prod_{i=1}^K \frac{\left(\frac{a_i^c}{1 - \rho_i} \right)^{n_i^c}}{n_i^c!}$$

Now taking the derivative of (A4.4.1) with respect to the open throughput at center j , ρ_j^o , we obtain:

$$\frac{\partial G(\bar{N})}{\partial \rho_j^*} = \frac{G(\bar{N})}{\rho_j^*} L_j^*(\bar{N}) \quad (\text{A4.4.3})$$

Taking the derivative of (A4.4.2) with respect to ρ_j^* gives:

$$\begin{aligned} \frac{\partial G(\bar{N})}{\partial \rho_j^*} &= \frac{1}{1 - \rho_j^*} G(\bar{N}) + G^* \left(\frac{G^*(\bar{N})}{1 - \rho_j^*} \sum_{i=1}^K L_{ij}^*(\bar{N}) \right) \\ &= \frac{G(\bar{N})}{1 - \rho_j^*} [1 + L_j^*(\bar{N})] \end{aligned} \quad (\text{A4.4.4})$$

Equating (A4.4.3) and (A4.4.4) gives:

$$L_j^*(\bar{N}) = \frac{\rho_j^*}{1 - \rho_j^*} [1 + L_j^*(\bar{N})] \quad (\text{A4.4.5})$$

which is the desired result.

(2) We next show that the main MVA equation can be easily proved, by taking the derivatives of the normalization constant. Again, assume that the centers in the network are single server fixed rate. A recursive expression for the normalization constant of a closed queueing network is [LAVE83]:

$$G_l(\bar{N}) = G_{l-1}(\bar{N}) + \sum_{i=1}^K e_{il} G_l(\bar{N} - \bar{e}_i) \quad (\text{A4.4.6})$$

where the subscript l in $G_l(\bar{N})$ indicates that $G_l(\bar{N})$ is the normalization constant of a network with centers 1 to l only. Equation (A4.4.6) is the key expression for the convolution algorithm [LAVE83]. Assuming $l = J = j$ and taking the derivative of both sides of (A4.4.6) with respect to θ_j , yields:

$$\frac{\partial G_l(\bar{N})}{\partial \theta_j} = z_j G_l(\bar{N} - \bar{e}_j) + \sum_{i=1}^K e_{ij} \frac{\partial G_l(\bar{N} - \bar{e}_i)}{\partial \theta_j} \quad (\text{A4.4.7})$$

where $\partial G_{l-1}(\bar{N}) / \partial \theta_j = 0$ since center j is not present in the network when the normalization constant is $G_{l-1}(\bar{N})$. Using the theorems of the previous sections we

obtain:

$$\frac{G_j(\bar{N})}{\theta_j} L_{hj}(\bar{N}) = z_{hj} G_j(\bar{N}-\bar{e}_h) + \sum_{i=1}^K \frac{a_{ij}}{\theta_j} G_j(\bar{N}-\bar{e}_i) L_{hj}(\bar{N}-\bar{e}_i)$$

and so,

$$\begin{aligned} L_{hj}(\bar{N}) &= a_{hj} \frac{G_j(\bar{N}-\bar{e}_h)}{G_j(\bar{N})} + \sum_{i=1}^K a_{ij} \frac{G_j(\bar{N}-\bar{e}_i)}{G_j(\bar{N})} L_{hj}(\bar{N}-\bar{e}_i) \\ &= a_{hj} \lambda_{hj}(\bar{N}) + \sum_{i=1}^K a_{ij} \lambda_{ij}(\bar{N}) L_{hj}(\bar{N}-\bar{e}_i) \end{aligned} \quad (\text{A4.4.8})$$

which implies that

$$\begin{aligned} L_j(\bar{N}) &= \sum_{i=1}^K a_{ij} \lambda_{ij}(\bar{N}) + \sum_{i=1}^K a_{ij} \lambda_{ij}(\bar{N}) L_j(\bar{N}-\bar{e}_i) \\ &= \sum_{i=1}^K a_{ij} \lambda_{ij}(\bar{N}) [1 + L_j(\bar{N}-\bar{e}_i)] \end{aligned} \quad (\text{A4.4.9})$$

since $L_j(\bar{N}) = \sum_{i=1}^K L_{hj}(\bar{N})$. This last equation is the MVA equation for the total number of jobs at center j .

The form of equation (A4.4.9) suggests that

$$L_{hj}(\bar{N}) = a_{ij} \lambda_{ij}(\bar{N}) [1 + L_j(\bar{N}-\bar{e}_i)] \quad (\text{A4.4.10})$$

which is the main MVA equation. To prove that this is indeed true, substituting (A4.4.10) into (A4.4.8) should lead to an identity. From (A4.4.10) we have that

$$L_{hj}(\bar{N}-\bar{e}_i) = \frac{L_{hj}(\bar{N})}{\lambda_{ij}(\bar{N}) a_{ij}} - [1 + \sum_{i \neq j} L_{hj}(\bar{N}-\bar{e}_i)] \quad (\text{A4.4.11})$$

Substituting (A4.4.11) into (A4.4.8) we get

$$\begin{aligned}
L_{b_j}(\bar{N}) &= a_{b_j} \lambda_{b_j}(\bar{N}) + \sum_{i=1}^K a_{b_j} \lambda_{b_j}(\bar{N}) \left(\frac{L_{b_j}(\bar{N})}{\lambda_{b_j}(\bar{N}) a_{b_j}} - 1 - \sum_{s \neq b_j} L_{b_j}(\bar{N} - \bar{e}_s) \right) \\
&= a_{b_j} \lambda_{b_j}(\bar{N}) + L_{b_j}(\bar{N}) - \sum_{i=1}^K a_{b_j} \lambda_{b_j}(\bar{N}) - \sum_{s \neq b_j} \sum_{k=1}^K a_{b_j} \lambda_{b_j}(\bar{N}) L_{b_j}(\bar{N} - \bar{e}_s) \\
&= a_{b_j} \lambda_{b_j}(\bar{N}) + L_{b_j}(\bar{N}) - \sum_{i=1}^K a_{b_j} \lambda_{b_j}(\bar{N}) - \sum_{s \neq b_j} [L_{b_j}(\bar{N}) - a_{b_j} \lambda_{b_j}(\bar{N})] \\
&= a_{b_j} \lambda_{b_j}(\bar{N}) + L_{b_j}(\bar{N}) - \sum_{i=1}^K a_{b_j} \lambda_{b_j}(\bar{N}) - L_{b_j}(\bar{N}) + L_{b_j}(\bar{N}) + \sum_{i \neq b_j} a_{b_j} \lambda_{b_j}(\bar{N}) \\
&= L_{b_j}(\bar{N})
\end{aligned}$$

where the third equality follows from (A4.4.8). This is the identity we needed to obtain.

(3) Finally, we show that another recursively relation for the variance of queue lengths of a queueing network with multiple chains can be obtained only in terms of the same variance for lower populations, without the need to calculate the covariances as in Lemma A4.3. To see that we take the derivative of equation (A4.4.8) with respect to θ_{b_j} (instead of using equation (A4.3.10) as in Lemma A4.3). Following the same steps used in Lemma A4.3 we get:

$$\begin{aligned}
V_{b_j}(\bar{N}) &= \lambda_{b_j}(\bar{N}) a_{b_j} [1 - L_{b_j}(\bar{N}) + 2L_{b_j}(\bar{N} - \bar{e}_{b_j})] \\
&\quad - \sum_{i=1}^K \lambda_{b_j}(\bar{N}) a_{b_j} \left([L_{b_j}(\bar{N}) - L_{b_j}(\bar{N} - \bar{e}_i)] L_{b_j}(\bar{N} - \bar{e}_i) + V_{b_j}(\bar{N} - \bar{e}_i) \right)
\end{aligned} \tag{A4.4.12}$$

REFERENCES

- [Agra83] Agrawal, S. C. and J. P. Buzen, "The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems," *ACM Transactions on Computer Systems*, Vol. 1, 1983, pp. 116-143.
- [Baer76] Baer, J. L., "Multiprocessing Systems," *IEEE Transactions on Computers*, December 1978.
- [Bard78] Bard, Y., "An Analytic Model of the VM/370 System," *IBM Journal of Research and Development*, No. 22, 1978, pp. 498-508.
- [Bard79] Bard, Y., "Some Extensions to Multiclass Queueing Network Analysis," in *Performance of Computer Systems*, E. Gelenbe, Ed. North-Holland, 1979, pp. 51-61.
- [Bard80] Bard, Y., "A Model of Shared DASD and Multipathing," *Communications of the ACM*, Vol. 23, No. 10, 1980.
- [Bard81] Bard, Y., "A Simple Approach to System Modeling," *Performance Evaluation*, Vol. 1, 1981, pp. 225-248.
- [Bask75] Baskett, F., K. M. Chandy, R. R. Muntz, and F. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM*, Vol. 22, No. 2, April 1975, pp. 248-260.
- [Berr82] Berry, R., K. M. Chandy, J. Misra, and D. M. Neuse, *Paws 2.0: Performance Analyst's Workbench Modelling Methodology and User's Manual*, Austin, Texas: Information Research Associates, 1982.
- [Bran82] Brandwajn, A., "Fast Approximate Solution of Multiprogramming Models," *Performance Evaluation Review*, No. 11, 1982, pp. 141-149.
- [Brya83] Bryant, R. M., A. E. Krzesinski, and P. Teunissen, "The MVA Pre-empt Resume Priority Approximation," in *Proceedings ACM Sigmetrics Conference on Measurements and Modeling of Computer Systems*, Minneapolis: 1983.

- [Bux81] Bux, W., "Local Area Subnetworks: A Performance Comparison," *IEEE Transactions on Communications*, Vol. 29, No. 10, 1981, pp. 1465-1473.
- [Buze73] Buzen, J. P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM*, Vol. 16, 1973, pp. 527-531.
- [Cant74] Cantor, D. G. and M. Gerla, "Capacity Allocation in Distributed Computer Networks," in *Proceedings of the Seventh Hawaii International Conference on System Sciences*, Honolulu, Hawaii: January 1974, pp. 115-117.
- [Chan75] Chandy, K. M., U. Herzog, and L. S. Woo, "Parametric Analysis of Queueing Networks," *IBM Journal of Research and Development*, Vol. 19, 1975, pp. 43-49.
- [Chan80] Chandy, K. M. and C. H. Sauer, "Computational Algorithms for Product Form Queueing Networks," *Communications of the ACM*, Vol. 23, No. 10, 1980, pp. 573-583.
- [Chan82a] Chandy, K. M. and S. Lakshmi, "An Approximation Technique for Queueing Networks with Preemptive Priority Queues," University of Texas at Austin, 1982.
- [Chan82b] Chandy, K. M. and D. Neuse, "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computer Systems," *Communications of the ACM*, Vol. 25, 1982, pp. 126-134.
- [Chan83] Chandy, K. M. and A. J. Martin, "A Characterization of Product-Form Queueing Networks," *JACM*, Vol. 30, No. 2, April 1983, pp. 286-299.
- [Chow83] Chow, W. M., "Approximations for Large Scale Closed Queueing Networks," *Performance Evaluation*, No. 3, 1983, pp. 1-12.
- [Cour75] Courtois, P. J., "Decomposability, Instabilities and Saturation in Multiprogramming Systems," *Communication of the ACM*, Vol. 18, No. 7, July 1975.
- [Cour77] Courtois, P. J., *Decomposability: Queueing and Computer System Applications*, New York: Academic Press, 1977.
- [DeSo83] de-Souza-e-Silva, E., S. S. Lavenberg, and R. R. Muntz, "A Perspective on Iterative Methods for the Approximate Analysis of Closed Queueing Networks," *Proceedings of the International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, 1983, pp. 191-210.

- [Ensl77] Enslow, P., "Multiprocessor Organization - A Survey," *Computing Surveys*, March 1977.
- [Frat73] Fratta, L., M. Gerla, and L. Kleinrock, "The Flow Deviation Method - An Approach to Store-and-Forward Communication Network Design," *Networks*, Vol. 3, 1973.
- [Gafn84] Gafni, E. and M. Gerla, "Distributed Load Balancing in a Computer System.," Tech. Rep. To be published, 1984.
- [Gall77] Gallager, R. G., "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications*, Vol. COM-25, No. 1, January 1977, pp. 73-85.
- [Gold83] Goldberg, A., G. Popek, and S. S. Lavenberg, "A Validated Distributed System Performance Model," *Performance 83*, 1983, pp. 251-268.
- [Heff82] Heffes, H., "Moment Formulae for a Class of Mixed Multi-Job-Type Queueing Networks," *The Bell System Technical Journal*, Vol. 61, No. 5, May-June 1982, pp. 709-745.
- [Heid82] Heidelberger, P. and K. S. Trivedi, "Queueing Network Models for Parallel Processing with Asynchronous Tasks," *IEEE Transactions on Computers*, No. C-31, 1982, pp. 1099-1108.
- [Heid83] Heidelberger, P. and K. S. Trivedi, "Analytic Queueing Models for Programs with Internal Concurrency," *IEEE Transactions on Computers*, No. C-32, 1983, pp. 73-82.
- [Jack63] Jackson, J. R., "Jobshop-like Queueing Systems," *Management Science*, Vol. 10, 1963.
- [Jaco80] Jacobson, P. A. and E. D. Lazowska, "The Method of Surrogate Delays: Simultaneous Resource Possession in Analytic Models of Computer Systems," Department of Computer Science, University of Washington, Tech. Rep. 80-04-03, December 1980.
- [Jaco82] Jacobson, P. A. and E. D. Lazowska, "Analyzing Queueing Networks with Simultaneous Resource Possession," *Communications of the ACM*, No. 25, 1982, pp. 142-151.
- [Jaco83] Jacobson, P. A. and E. D. Lazowska, "A Reduction Technique for Evaluating Queueing Networks with Serialization Delays," *Performance 83*, North-Holland, 1983, pp. 45-59.
- [Klei76] Kleinrock, L., in *Queueing Systems, Volume II: Computer Applications*, New York: Wiley-Interscience, 1976.

- [Koba83] Kobayashi, H. and M. Gerla, "Optimal Routing in Closed Queueing Networks," *ACM Transactions on Computer Systems*, Vol. 1, No. 4, November 1983, pp. 294-310.
- [Krit82] Kritzinger, P. S., S. van Wyk, and A. E. Krzesinski, "A Generalization of Norton's Theorem for Multiclass Queueing Networks," *Performance Evaluation*, No. 2, 1982, pp. 98-107.
- [Lam80] Lam, S. S., "Carrier Sense Multiple Access Protocol for Local Networks," *Computer Networks*, Vol. 4, 1980, pp. 21-32.
- [Lam82a] Lam, S. S. and J. W. Wong, "Queueing Network Models of Packet Switching Networks, Part 1: Open Networks," *Performance Evaluation*, No. 2, 1982, pp. 9-21.
- [Lam82b] Lam, S. S. and J. W. Wong, "Queueing Network Models of Packet Switching Networks, Part 2: Networks with Population Size Constraints," *Performance Evaluation*, No. 2, 1982, pp. 161-180.
- [Lam83a] Lam, S. S. and Y. L. Lien, "A Tree Convolution Algorithm for the Solution of Queueing Networks," *Communications of the ACM*, No. 26, 1983, pp. 203-215.
- [Lam83b] Lam, S. S., "A Simple Derivation of the MVA and LBANC Algorithms from the Convolution Algorithm," *IEEE Transactions on Computers*, Vol. C-32, No. 11, November 1983, pp. 1062-1064.
- [Lave80] Lavenberg, S. S. and M. Reiser, "Stationary State Probabilities of Arrival Instants for Closed Queueing Networks with Multiple Types of Customers," *Journal of Applied Probabilities*, No. 17, 1980, pp. 1048-1061.
- [Lave82] Lavenberg, S. S., "Closed Multichain Queueing Networks with Large Population Sizes," in *Applied Probability - Computer Science: The Interface Volume I*, R. L. Disney T. J. Ott, Ed. Birkhauser, 1982, pp. 219-249.
- [Lave83] Lavenberg, S. S. (Editor), *Computer Performance Modeling Handbook*, New York: Academic Press, 1983.
- [Lazo82] Lazowska, E. D. and J. Zahorjan, "Multiple Class Memory Constrained Queueing Networks," *Performance Evaluation Review*, No. 11, 1982, pp. 130-140.
- [Lien82] Lien, Y. L. and S. S. Lam, "Optimal Routing in Networks with Flow Controlled Virtual Channels," in *Proceedings Computer Network Performance Symposium*, Maryland: April 1982.

- [Mari79a] Mariani, M. P. and D. F. Palmer (Editors), *Tutorial: Distributed System Design*: IEEE, 1979. Catalog no. EH0151-1.
- [Mari79b] Marie, R. A., "An Approximate Analytical Method for General Queueing Networks," *IEEE Transactions on Software Engineering*, Vol. 5, 1979, pp. 530-538.
- [McKe81] McKenna, J., D. Mitra, and K. G. Ramakrishnan, "A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, and Generalizations," *The Bell System Technical Journal*, Vol. 60, No. 5, May-June 1981.
- [McKe82] McKenna, J. and D. Mitra, "Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage," *The Bell System Technical Journal*, Vol. 61, No. 5, 1982, pp. 661-683.
- [McKe83] McKenna, J. and D. Mitra, "Asymptotic Expansions and Integral Representations of Moments of Queue Lengths in Closed Markovian Networks," *Int. Seminar on Modelling and Performance Evaluation Methodology*, January 1983.
- [Moll81] Molloy, M., "On the Integration of Delay and Throughput Measures in Distributed Processing Models," UCLA Computer Science Department, Tech. Rep. CSD-810921, September 1981.
- [Munt74] Muntz, R. R. and J. W. Wong, "Asymptotic Properties of Closed Queueing Network Models," in *Proceedings Eighth Annual Princeton Conference on Information Systems*, Princeton, N.J.: March 1974.
- [Neus81] Neuse, D. and K. M. Chandy, "SCAT: A Heuristic Algorithm for Queueing Network Models of Computing Systems," *Performance Evaluation Review*, No. 10, 1981, pp. 59-79.
- [Neus82] Neuse, D. and K. M. Chandy, "HAM: The Heuristic Aggregation Method for Solving General Closed Queueing Network Models of Computer Systems," *Performance Evaluation Review*, No. 11, 1982, pp. 195-212.
- [Niel82] Nielsson, P. O. and M. Gerla, "Routing and Flow Control Interplay in Computer Networks," in *Proceedings Fifth International Conference on Computer Communications*, Atlanta, GA: October 1982, pp. 84-89.
- [Orte70] Ortega, J. M. and W. C. Rheinboldt, *Iterative Solution of Non-linear Equations in Several Variables*, New York: Academic Press, 1970.

- [Penn75] Pennoti, M. C. and M. Schwartz, "Congestion Control in Store and Forward Tandem Links," *IEEE Transactions on Communications*, Vol. 23, No. 12, December 1975.
- [Pope81] Popek, G., B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel, "LOCUS: A Network Transparent, High Reliability Distributed System," *Proceedings of the Eighth Symposium on Operating Systems Principles*, December 1981.
- [Rama82] Ramakrishnan, K. G. and D. Mitra, "An Overview of PANACEA, A Software Package for Analyzing Markovian Queueing Networks," *The Bell System Journal*, Vol. 61, No. 10, 1982, pp. 2849-2871.
- [Rask78] Raskin, L., "A Performance Evaluation of Multiple Processor Systems," Carnegie Mellon University, Tech. Rep. CMU-CS-78-141, August 1978. PhD dissertation.
- [Reis78] Reiser, M. and H. Kobayashi, "On the Convolution Algorithm for Separable Queueing Networks," *International Symposium on Computer Performance*, 1978, pp. 109-117.
- [Reis79] Reiser, M., "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control," *IEEE Transactions on Communications*, No. 27, 1979, pp. 1199-1209.
- [Reis80] Reiser, M. and S. S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," *Journal of the ACM*, No. 27, 1980, pp. 313-322.
- [Reis81] Reiser, M., "Mean-Value Analysis and Convolution Method for Queue-Dependent Servers in Closed Queueing Networks," *Performance Evaluation*, No. 1, 1981, pp. 7-18.
- [Saty80] Satyanarayan, M., "Commercial Multiprocessing Systems," *Computer*, May 1980.
- [Saue81a] Sauer, C. H. and K. M. Chandy, *Computer System Performance Modelling*. Prentice Hall, 1981.
- [Saue81b] Sauer, C. H., "Approximate Solution of Queueing Networks with Simultaneous Resource Possession," IBM T. J. Watson Research Center, Yorktown Heights, Tech. Rep. RC-8679, January 1981.
- [Saue81c] Sauer, C. H., E. A. MacNair, and J. F. Kurse, "Computer Communication System Modelling with the Research Queueing Package Version 2," IBM T. J. Watson Research Center, Yorktown Heights, Tech. Rep. RA-128, November 1981.

- [Saue81d] Sauer, C. H., "Computational Algorithms for State-Dependent Queueing Networks," IBM T. J. Watson Research Center, Yorktown Heights, Tech. Rep. RC-8698, February 1981.
- [Saue83] Sauer, C. M., "Corrigendum: Computational Algorithms for State Dependent Queueing Networks," *ACM Transactions on Computer Systems*, No. 4, 1983, p. 369.
- [Saue84] Sauer, C. M., E. A. MacNair, and J. F. Kurose, "Queueing Network Simulations of Computer Communication," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, No. 1, January 1984, pp. 203-220.
- [Schw79] Schweitzer, P., "Approximate Analysis of Multiclass Closed Networks of Queues," *International Conference on Stochastic Control and Optimization*, 1979.
- [Sevc79] Sevcik, K. and I. Mitrani, "The Distribution of Queueing Network States at Input and Output Instants," in *Proceedings of the 4th International Symposium on Modeling and Performance Analysis of Computer Systems*, Amsterdam: North-Holland, 1979.
- [Sevc77] Sevcik, K. C., "Priority Scheduling Disciplines in Queueing Network Models of Computer Systems," in *Information Processing 77*, H. Gilchrist, Ed. North-Holland, 1977, pp. 565-570.
- [Suri83] Suri, R., "Robustness of Queueing Network Formulas," *JACM*, Vol. 30, No. 3, July 1983, pp. 584-594.
- [Tant84] Tantawi, A.N. and D. Towsley, "Optimal Load Balancing in Distributed Computer Systems," IBM Research Report, Tech. Rep. RC 10346, January 1984.
- [Thom83] Thomasian, A., "Queueing Network Models to Estimate Serialization Delays in Computer Systems," in *Performance 83*, S. K. Tripathi, Ed. North-Holland, 1983.
- [Thur79] Thurber, K. S. (Editor), *Tutorial: Distributed Processor Communication Architecture*: IEEE, 1979. Catalog no. EH0152-9.
- [Tucc82a] Tucci, S. and E. A. MacNair, "Implementation of Mean Value Analysis for Open, Closed and Mixed Queueing Networks," IBM T. J. Watson Research Center, Yorktown Heights, N. Y., Tech. Rep. RC 9392, 1982.
- [Tucc82b] Tucci, S. and C. H. Sauer, "The Tree MVA Algorithm," IBM T. J. Watson Research Center, Tech. Rep. RC-9338, April 1982.

L

4

1

v

- [Vern83] Vernon, M., E. de Souza e Silva, and G. Estrin, "Performance Evaluation of Asynchronous Concurrent Systems: The UCLA Graph Model of Behavior," *Performance 83*, May 1983.
- [Walk83a] Walker, B., G. Popek, R. English, C. Kline, and G. Thiel, "The LOCUS Distributed Operating System," in *Proceedings of the Ninth Symposium on Operating Systems Principle*, Bretton Woods, NH: October 1983, pp. 10-13.
- [Walk83b] Walker, B., *Issues of Network Transparency and File Replication in Distributed Systems: LOCUS*: UCLA Computer Science Department, 1983. PhD dissertation.
- [Will76] Williams, A. C. and R. A. Bhandiwand, "A Generating Function Approach to Queueing Network Analysis of Multiprogrammed Computers," *Networks*, No. 6, 1976, pp. 1-22.
- [Zaho81] Zahorjan, J. and E. Wong, "The Solution of Separable Queueing Network Models Using Mean Value Analysis," *Performance Evaluation Review*, Vol. 10, 1981, pp. 80-85.
- [Zaho82] Zahorjan, J., K. C. Sevcik, D. L. Eager, and B. Galler, *Balanced Job Bound Analysis of Queueing Networks* *J Communications of the ACM*, February 1982.
- [Zaho83] Zahorjan, J., "Workload Representations in Queueing Models of Computer Systems," *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 1983.