RELATÓRIO TÉCNICO

QUEUEING NETWORKS: SOLUTIONS AND APPLICATIONS

E. de Souza e Silva
* Richard R. Muntz

NCE-08/89
Dezembro/89

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

* UCLA Computer Science Department

Esta pesquisa contou com o apoio financeiro do programa de cooperação internacional CNPq-MSF(USA), da CAPES e da IBM-Brasil.
Este trabalho foi publicado como relatório técnico do Departamento de Ciência da Computação da UCLA CSD-890052 e será publicado como um capítulo do livro "Stochastic Analysis of Computer and Communication Systems" (North-Holland, 1990)

RESUMO

Durante as últimas duas décadas modelos de redes de filas provaram ser uma ferramenta versátil para avaliação de desempenho de sistemas de computação e sistemas de comunicação. Este capítulo faz um apanhado geral da área com ênfase em aplicações.

Começamos com uma breve retrospectiva histórica que serve também para introduzir os pontos mais importantes e as áreas de aplicação. Resultados formais para redes de filas em forma de produto são revisados com ênfase na modelagem de sistemas de computação. Algoritmos de computação, análise de sensibilidade e técnicas de otimização estão entre os tópicos revistos. Muitas dentre importantes aplicações de redes de filas não são tratáveis por análise exata e uma série (frequentemente confusa) de métodos de aproximação tem sido desenvolvida. Uma taxonomia de métodos de aproximação é dada e usada como base para a revisão dos mais importantes métodos de aproximação propostos. Uma revisão das aplicações de redes de filas em um número de áreas é feita, incluindo planejamento de capacidade de sistemas de computação, redes de comunicação por chaveamento de pacotes, processamento paralelo, sistemas de banco de dados e modelagem de confiabilidade.

ABSTRACT

During the past two decades queueing network models have proven to be a versatile tool for computer system and computer communication system performance evaluation. This chapter provides a survey of the field with a particular emphasis on applications.

We start with a brief historical retrospective which also serves to introduce the major issues and application areas. Formal results for product form queueing networks are reviewed with particular emphasis on the implications for computer systems modeling. Computation algorithms, sensitivity analysis and optimization techniques are among the topics covered. Many of the important applications of queueing networks are not amenable to exact analysis and an (often confusing) array of approximation methods have been developed over the years. A taxonomy of approximation methods is given and used as the basis for surveying the major approximation methods that have been studied. The application of queueing networks to a number of areas is surveyed, including computer system capacity planning, packet switching networks, parallel processing, database systems and availability modeling.

Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

QUEUEING NETWORKS: SOLUTIONS AND APPLICATIONS

Edmundo de Souza e Silva
Richard R. Muntz

September 1989
CSD-890052

Abstract

During the past two decades queueing network models have proven to be a versatile tool for computer system and computer communication system performance evaluation. This chapter provides a survey of the field with a particular emphasis on applications.

We start with a brief historical retrospective which also serves to introduce the major issues and application areas. Formal results for product form queueing networks are reviewed with particular emphasis on the implications for computer systems modeling. Computation algorithms, sensitivity analysis and optimization techniques are among the topics covered. Many of the important applications of queueing networks are not amenable to exact analysis and an (often confusing) array of approximation methods have been developed over the years. A taxonomy of approximation methods is given and used as the basis for surveying the major approximation methods that have been studied. The application of queueing networks to a number of areas is surveyed, including computer system capacity planning, packet switching networks, parallel processing, database systems and availability modeling.

Contents

1 Introduction.

Computer system and computer communication performance evaluation is concerned with the measurement, analysis and prediction of the behavior of these systems. Many different disciplines contribute to the set of techniques that are applicable to the problems in this area, including measurement instrumentation, statistical data analysis, simulation and various topics in operations research. This chapter concentrates on one particular tool for computer system performance evaluation: queueing networks. Queueing networks continue to be widely used in many aspects of computer system modeling. It is the purpose of this chapter to survey the available results and their application to practical problems. We start with a brief historical review of the developments in queueing networks and their application to computer system and computer communication problems. We assume that the reader has basic familiarity with queueing networks and concentrate on organizing the topics in the field rather than presenting a tutorial. In the latter part of this section, we will outline the remainder of the chapter.

In the early 1960's the first queueing theoretic models of computer and communication systems began to appear [100,101]. The computer system models were most often single server models in which the main interest was in studying "time-sharing" scheduling algorithms, e.g. round-robin. These preemptive algorithms had became feasible due to efficiently preemptable processors. Representing the computer as a single resource (the central processor) was justified on the basis of the processor being the limiting resource in the system.

The machine repairman model (or finite population model) might be considered the first rudimentary form of queueing network to be applied to computer system performance analysis. As early as 1967, Scherr [158] had applied the machine repairman model to the CTSS (Compatible Time-Sharing System) at MIT. Lassettre and Scherr later applied the machine repairman model to IBM's TSO (Time Sharing Option) system. This was a much bigger leap of faith since the system software and hardware was now much more complex [115]. A particularly interesting observation was made in the paper on the basis of simulation experiments. It appeared that the mean response time predicted by the model was insensitive to the exact distribution of service time and appeared to be dependent only on the mean service required. Theoretical verification of this "insensitivity" property came much later. A survey of the results in this modeling era can be found in [102].

While research into "time-sharing scheduling algorithms" did yield many interesting results, computer systems quite rapidly became more parallel and more complex. It was then no longer sufficient to model such a system as a single resource (or server). Advances in I/O architectures permitted asynchronous I/O operation and, coupled with larger memories, multiprogramming became commonplace. A major impact of stochastic modeling has come from concentrating on the multiple resources in systems and accounting for the behavior patterns of customers in demanding service from these resources. These early models concentrated on sequential resource usage for a given customer, but did represent concurrent activity by different customers. It should be noted that Kleinrock's work [101] on communication network models is an early example of this concern with multiple resource models (in this case sets of

communication channels) rather than with the details of scheduling disciplines at a single server. During the next decade, the 1970s, the computer science world became aware of queueing networks and their applicability to many of the performance modeling problems being raised by more and more parallel, complex systems in single site computer systems, computer communication networks, and distributed systems.

It is interesting that Jackson's first results on networks of exponential servers had been available for some time [90,91] (see also [72]) but had not been capitalized on by the computer science community until the 1970's. In the early 1970's, two pioneering efforts were carried out in the application of Jackson's results to modeling of multiprogramming computer systems. One was a study by Moore [131] at the University of Michigan in the modeling of the Michigan Time-sharing System using Jackson networks. A validation study showed that the model predictions were typically within ten percent when the model parameters could be estimated accurately. Moore observed during the study that it appeared in some cases that the response time was determined solely by the relative load at the resources and not by any details of the routing behavior. This is another case of an empirically observed "insensitivity" that was later proved formally for certain classes of queueing networks [148]. The second research effort was that of Buzen at Harvard [27]. Buzen's work concentrated on the central server model which is illustrated in Figure 1. Buzen introduced several of the research topics that were to occupy many researchers in the next decade. He introduced the routing optimization problem in the context of the central server model and was the first to consider the computational aspects of queueing networks (the complexity of actually computing performance measures), and he introduced the now well known convolution algorithm. It is interesting to note that computational problems apparently did not become an issue until actual applications were attempted.

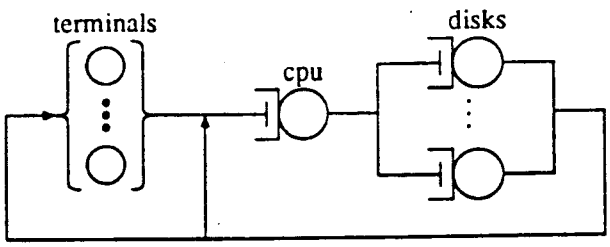

Figure 1: Central server model with terminals.

Following the pioneering work of Buzen and Moore, there was an explosion in research results including many major advances in both the theory and practice of queueing network models. It is the purpose of this chapter to survey much of this work, although it is not possible to attempt anything close to a complete account of subsequent work in this introduction: this will occupy us for the entire chapter. Below we outline the topics that are covered in the remaining sections of the chapter and only briefly mention a few of the major contributions.

In the early to mid 1970s, as one might expect, there were many studies in building and validating queueing network models of computer systems. Some of these are reported in [9,13,54,69,99,143,162] At the same time, mainly in response to the short-

comings of the currently available results, there were advances in queueing network theory. In [12], the product form results of Jackson were extended to allow multiple types of customers (customer chains), to allow general service time distributions in certain cases and to allow a more general form of routing. (Customers could change class as they moved among the resources and, since transitions could depend on the class of a customer, the routing behavior could incorporate some dependence on past history.) Since customers with different behavior could now be included in a model, the model no longer had to be open or closed but rather each customer chain could be open or closed as illustrated in Figure 2. Open chains, e.g. chain A in Figure 2

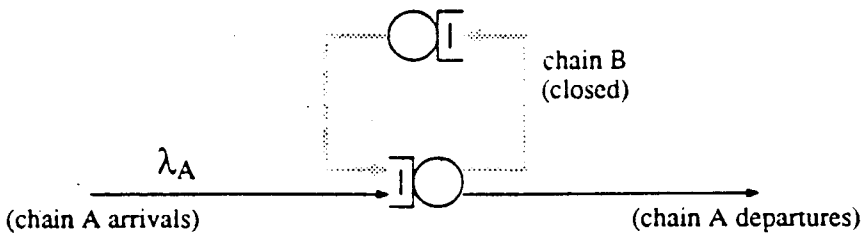

Figure 2: Queueing network with open and closed chains.

have external Poisson arrivals and departures from the network. Closed chains have neither arrivals nor departures, but rather the number of customers in the network remains constant. Open chains are more appropriate for modeling situations in which the customer population is very large and only a small fraction of the population is actually in the network model at any time. In this case, there is no significant negative feedback on the arrival rate as a function of the current population of the network. Closed chains are more appropriate when the population is considered constant, or nearly so. A common case is that of a multiprogramming system which is heavily loaded. Assume that the degree of multiprogramming is limited by memory to some fixed limit. Under heavy loading we assume that when one job finishes another will immediately take its place, and thus the number of customers (jobs) remains fixed.

One of the effects of [12] was to raise the question of what other types of queues and customer behavior could be represented in a queueing network while still retaining a product form solution. This is the *characterization problem.* Basically, the goal is to provide an effective test to determine if a queueing network model has product form. Initially the solutions that were available were all developed in an ad hoc manner, e.g. start with a solution for a simple case; then add some additional feature; solve a small network by hand; guess a generalization to the product form solution; and check the global balance equations for a general network. Several approaches were developed that provided more systematic formal development of product form results [29,96]. This issue is the main topic of Section 2.

The study of product form queueing networks has provided the stationary state probabilities for a very useful set of models. Many performance measures of interest are then, in theory, available as functions of these state probabilities. However, for closed queueing networks the state probabilities are not given directly, but rather a normalization constant must be computed to determine actual state probabilities. The cost of computing performance measures grows combinatorially with the number of

queues, the number of customers and the number of customer chains. In particular, as queueing network models began to be applied to communication networks and distributed systems, computational cost and numerical stability became an issue in the actual use of the formal results. The Mean Value Analysis Algorithm was a major advance in this area [149] which permits calculation of mean queue lengths and throughputs without calculation of the individual state probabilities. This has several advantages including the avoidance of numerical problems which can occur with the convolution algorithm. Other algorithms have been developed which are particularly efficient for models with a large number of chains but are "sparse" in that each chain visits only a small subset of queues [112]. These are common characteristics in communication network models. Recently we have seen the development of a new type of computational algorithm which is especially efficient for models with many chains [39,48]. Another approach to avoiding the high computational cost is to find efficient approximations, e.g. [129]. Computation algorithms for product form networks are the subject of Section 3.

In attempting to build and validate models of real computer and communications systems, it quickly became evident that there are many features that occur in real systems but which cannot be represented in product form queueing networks. Those features of real systems that have a first order effect on the performance measures of interest and which are often present in computer systems are candidates for special attention. For example, these include: memory contention, priorities, and synchronization between customers. Driven by the need to model such features, there has been extensive work on developing useful approximation methods. There are at least two ways to organize the work on approximations: one is to organize the material on the basis of the application and the other is to place the emphasis on the approximation technique. We have chosen to do both. Section 4 discusses approximation techniques with reference to some practical applications for each technique. Section 6 discusses application areas and often refers back to Section 4 for details of the appropriate approximation technique.

Most of the topics covered in Section 3 through Section 4 are concerned with analysis of a model rather than synthesis of a system. The emphasis in Section 5 is on the application of queueing network models in system optimization. Many varieties of optimization problems can be posed in terms of queueing network models. For example, in a computer system the allocation of files to secondary storage devices will determine the branching probabilities to those devices. The obvious question is how to allocate the files so that some measure, e.g. the total throughput of the system, is maximized. Optimal routing in a communications network is a similar problem. Many versions of the load balancing problem can also be posed as an optimization problem in which there are constraints on the routing probabilities. The above problems assume a fixed set of resources, and the goal is to control the transition behavior of customers to optimize some performance measure. Another optimization problem concerns allocation of capacity (in other words, money) to servers to optimize a performance measure subject to constraints on total investment. Theoretical results and applications of these problems are presented in Section 5.

Our goal in writing this chapter is to present an up-to-date survey of the major advances in queueing network modeling with particular emphasis on their application

to computer system and communication network performance evaluation. A brief summary has just been given of the chapter organization and what topics are covered in each section. In this conclusion to the introduction, it seems appropriate to mention several topics which we will not cover. Two major areas that are not touched on here are the subjects of two other chapters in this book: blocking and sojourn times.

An alternative approach to the analysis of queueing networks is that developed by Buzen and Denning [54], called operational analysis. The operational analysis approach has the advantage of providing a more easily accessible approach to many of the results than using more classical stochastic analysis. However, the results available from this approach are a subset of those available with the classical techniques.

A number of queueing network based tools have been built and many are commercially available, e.g. [1,2,3,4,153,192]. Due to lack of space we have chosen not to discuss this topic.

In addition to topics that were deliberately omitted, there may be topics or particular results that have been inadvertently overlooked. The authors take full responsibility for any such blunders. Now, we begin!

2 Product Form Queueing Networks.

The most general method of solution for a queueing network model is to generate and solve the associated Markov chain using numerical techniques. This is clearly too costly in space and time for any but the most trivial network models. Product form queueing networks are a special class of queueing networks which have a particularly simple form for the stationary state probabilities and which have a number of interesting properties. We begin with a definition.

A queueing network will be said to be *product form* if the stationary state probabilities have the following form:

$$P(S) = \prod_{i=1}^{J} P_i(S_i) \tag{1}$$

where:

J is the number of queues in the network.

$S_i \in \mathcal{S}_i$, where $\mathcal{S}_i$ is a set of "states" associated with the i^{th} queue, $1 \leq i \leq J$.

$S = (S_1, S_2, ..., S_J)$ is a state of the network.

$P_i(S_i)$ is a function only of the state of the i^{th} queue and in general may depend on the type of the i^{th} queue.

Product form queueing networks are the foundation upon which the queueing network area is built. Several major factors have contributed to the success that they have enjoyed in practical applications. First and most importantly, empirical evidence shows that these models are often sufficiently accurate for quantitative performance prediction in many applications [121]. Second, solutions exist for models with an arbitrary number of resources and flexible "topology", i.e customer routing. This allows the results to be applied easily to a variety of system configurations.

In many applications the product form assumptions are violated, and in this case approximations are often required. The exact results for product form networks are often useful in guiding the heuristics used in inexact methods and/or are useful in solving submodels that are part of these inexact methods. These issues will be discussed in more depth in Section 4 where approximation techniques are the main topic.

In this section we concentrate on the stationary state probabilities for product form queueing networks. We first briefly present the results which are most well known and useful for computer systems modeling. We then discuss extensions and generalizations of the basic results. Due to the importance of product form networks, it is clearly of interest to provide a characterization of this type of network model. By a characterization we mean a set of necessary and sufficient conditions for a queueing network to have a product form solution. This issue is discussed in the following subsection. In subsection 2.2 we discuss some additional properties of product form networks that are particularly useful in applications.

2.1 Basic Results

We consider queueing networks that have the following characteristics [12]:

- There can be different types of customers. Each distinct type will be called a *chain*. For each chain k, there is a routing probability matrix P_k for which $P_k[i,j]$ is the probability that a chain k customer departing from queue i will next visit queue j. This type of routing is referred to as Markovian, since the transition probabilities are not a function of past history.

- Each chain can be *open* or *closed*. For an open chain there is a Poisson arrival process of customers to the network and at some queues there is a non zero probability that the customer may depart from the network. For an open chain, the total number of customers in the network is variable. For a closed chain the number of customers is fixed. There are no arrivals from outside the network and no departures.

- The service disciplines at the queues can be FCFS, PS (processor sharing), IS (infinite servers or delay) or LCFS-preemptive.

- Except for delay servers, all queues are SSFR (single server fixed rate).

- Successive service times of chain k customers at queue i are i.i.d. and exponentially distributed.

The class of networks described above has product form and (due to the exponential service time distributions and the service disciplines) the "state" of a queue is the number of customers of each chain at that queue, denoted $\vec{n}$. Then:

$$P_i(\vec{n}) = \frac{|\vec{n}!|}{\prod_{m=1}^{|\vec{n}|} \mu_i(m)} \prod_{k=1}^{K} \frac{a_{ik}^{n_k}}{n_k!}$$

where:

$\mu_i(m) = 1$ or m for SSFR or IS centers, respectively.
$a_{ik}^{n_k}$ is the relative service demand at the i^{th} queue by chain k customers.
The above results can be generalized in a number of ways [12.96.195]:

- Phase type service time distributions (i.e. formed from exponential stages) can be accommodated for IS, PS and LCFS-preemptive queues. In stationary state probabilities for the number of customers at the queues is independent of the service time distribution for the queue types mentioned.

- Queue length dependent service rates that are a function of the number of customers queued can be represented without sacrificing the product form.

- The routing of customers can be generalized to allow a limited form of history dependent routing by introducing customer classes [12.96]. When a customer departs from one queue and joins another we allow the customer to change class. In this way the same customer may visit the same queue in different classes and its behavior at departure may depend on the class. An example is shown in Figure 3 in which customers follow a "figure eight" pattern. This is a single chain of customers but two classes ("a" and "b") are used to distinguish between customers that have arrived at queue 2 from queue 1 and those that have arrived from queue 3. As with the phase type service time distributions, it can be shown that the product form is independent of this generalized routing behavior [148]. For a_{ik} one simply uses the total load of chain k at queue i, i.e. summing the load over all classes of chain k that visit queue i.

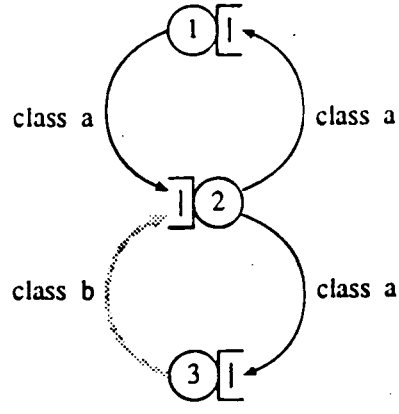

Figure 3: Queueing network with class changes.

- State dependent routing in which transition probabilities depend on the current congestion at the various queues is clearly of interest in modeling systems with dynamic routing, load balancing, etc. Unfortunately, except in special cases [83,179], including this feature generally results in a non-product form network.

Given the importance of product form networks, it is natural that considerable effort has been made to characterize precisely the class of queueing network models that admit a product form solution, i.e. determine a set of necessary and sufficient conditions for a queueing network model to be product form in the sense of Eq. (1).

The most useful characterizations are those that provide a means for easily testing a particular model to determine if it has product form. This "test" can be either in terms of the structure of the transition rate matrix or at the "system level" in terms of the queueing discipline, service rates, etc. The latter type of characterization is obviously more accessible and useful for systems designers, while the former generally leads to a more abstract and more general characterization. This section summarizes the major approaches and the results that have been achieved.

The class of models that have a product form solution has been difficult to characterize exactly. What is found in the literature generally are results that give sufficient conditions for product form solutions. The most general approach appears to be based on *quasi-reversibility*. An early algebraic treatment of quasi-reversibility (without that name) and its relationship to product form queueing networks can be found in [133]. An extensive development of these ideas and their application to queueing networks is due to Kelly [96]. Informally, a quasi-reversible queue is one for which the past departures, the current state and the future arrivals are mutually independent. A property that follows from this definition is that the arrival process and the departure process are Poisson.

The main result is a sufficient condition for a queueing network to be product form, namely: that the routing be Markovian and that each queue be quasi-reversible. When these conditions are met, then the network has a product form stationary distribution, and the factor $P_i(S_i)$, corresponding to queue i, is the marginal distribution of the the i^{th} queue when driven with a Poission arrival process of appropriate rate.

The set of queues that are quasi-reversible includes:

- FCFS with multiple classes. Each class must have the same exponential service time distribution.
- Processor sharing with multiple classes and arbitrary phase type service distributions.
- IS (delay server) with multiple classes and arbitrary phase type service distributions.
- Last-Come-First-Serve-preempt-resume with multiple classes and arbitrary phase type service distributions.

The quasi-reversibility results provide a sufficient condition for testing whether a new queueing model is one which can be incorporated into a network model and preserve product form. This characterization is useful for queueing theory enthusiasts but not for performance analysts who generally want to think in terms of queueing disciplines, number of servers, etc. and not in terms of transition rates or reversibility. We gave a short list of types of queues above which are quasi-reversible only to provide some evidence that it is a useful concept. The results on quasi-reversible queues can be translated into more generally accessible terms by considering a general class of parameterized queueing disciplines and determining the restrictions (if any) required for product form in terms of the parameters. For example, a rather general class of disciplines called *symmetric queues* have been shown to be quasi-reversible [96]. A symmetric queue is one with a Poisson arrival process for which the probability that

an arrival joins the i^{th} position in the queue when there are n customers queued is equal to the fraction of the service capacity that is given to the i^{th} queue position when there are $n+1$ customers queued. A processor sharing system, for example, can be described as a symmetric queue by defining that a customer that arrives when there are n customers in the queue have equal probability of entering any of the $n+1$ positions. Many queueing disciplines can be described in this framework, and it is known that all are quasi-reversible. With this characterization it is possible for a practitioner to determine that a given queueing discipline is one which can be included in a queueing network and retain product form. However, note that this is only a sufficient condition for a queue to be quasi-reversible (although a rather general one). It is also interesting to note that there can be multiple equivalent representations of the same queueing discipline, and while one of these may be a symmetric queue, others may not be. Thus the choice of representation rather than the intrinsic properties of the model may determine whether the queue is symmetric. The processor sharing queue is a simple example. Note that there is no compelling reason to have the arriving customers have equal probability of entering any queue position. However if, for example, the arriving customer joined the end of the queue, the queueing system would no longer be symmetric (although it is clearly still quasi-reversible).

The concept of symmetric queues is similar to the set of *station balanced* queueing disciplines that are described by Chandy and Martin in [29]. The work described in [29] takes a quite different approach to characterizing queueing networks with product form. They start with a (general but nevertheless constrained) class of queueing disciplines and a specific algebraic form for the product form solution, and so the results only apply in that restricted world. For example, one of the results is that a queueing model with non-exponential service time distribution will have product form if and only if it is *station balanced* and *balanced.* Thus both necessary and sufficient conditions are given in this case. This is unusual in that most results are only for sufficient conditions. The proof of necessity in the above result requires the assumption of the particular algebraic form for the solution. The algebraic form is one which is quite general, but none the less it is an assumption and does not follow from first principles.

Excellent treatments of the characterization of product form networks based on quasi-reversibility can be found in [96,195]. Other interesting work concerned with characterization of product form models can be found in [83,155,197,198].

2.2 Insensitivity/Robustness

The product form solutions presented above allow a broad range of customer behavior to be represented. It is interesting that some performance measures are *insensitive* to certain details of the model. These properties are interesting theoretically and also have practical significance, since they indicate when simpler models, often with fewer parameters, can be used. There are two consequences. First, the form of the solution is less complex and performance measures can be more efficiently calculated (see Section 3). Second, if the performance measures of interest are insensitive to certain parameters, then these parameters do not have to be provided (e.g., estimated from measurements).

The major result of this type concerns the insensitivity of the stationary distribution of the number of customers at queues in a product form network to any characteristic of the service time distribution except the mean (the service time distribution must be differentiable). This was mentioned above in connection with the station balanced queueing disciplines [29]. The most common disciplines that exhibit this property are processor sharing, delay (or infinite server) and LCFS-preemptive queues. Clearly then, if a product form network model is being used, only the mean service times need to be estimated. For more general and indepth treatments of insensitivity refer to [23,155,156,157,197,198]. An interesting recent application of these results to the analysis of circuit-switched networks can be found in [24,35].

Another insensitivity property concerns customer routing. The general model allows the definition of customer classes and class changes. Using this feature, the modeler can represent routing behavior that can depend on the customers past history of visits to queues. A simple example was discussed previously and illustrated in Figure 3. As previously mentioned, routing behavior that is introduced to represent "past path dependencies" can be suppressed and only the total relative visit service demands of a chain at a queue are required to obtain the queue length distributions. This set of parameters is often easier to obtain in practice. Note that the dynamic behavior of the model is a function of the detailed routing behavior; the claim is only that the stationary state distribution is not effected. Also, the distribution of customers at queues is obtained only on a per chain basis rather than by class (e.g., in the example in Figure 3 only the total demand by class a and class b customers need be known) but then only the distribution for the total number of customers at queue 2 is obtainable. This is sufficient for many applications.

In this section we have discussed some of the insensitivity results available for product form networks. We were concerned specifically with modifications to the structure of the model for which the stationary state distribution is invariant. A related topic is the sensitivity of performance measures to errors in parameter values. In this case one is concerned with metrics such as the percentage error in the performance measure as a function of the percentage error in the parameter. These topics are discussed in Section 5.

3 Computation Algorithms for Product Form Networks.

The stationary state probabilities for product form queueing networks provide the basis for the calculation of performance measures of these models. It is easy to see that most performance measures can be expressed as an *expected reward rate* as follows:

$$\mathcal{R} = \sum_{S_i \in \mathcal{S}} R(S_i) P(S_i)$$

where $\mathcal{S}$ is the set of states. For open networks these calculations often collapse to closed form expressions if the service capacities are of simple form, e.g. fixed rate servers, infinite servers, etc. [12]. For closed networks the situation is more complicated. Naive evaluation of performance measures by direct use of this equation

would imply a computation cost that is proportional to the number of states in the model, and this is often prohibitively expensive for realistic models. Therefore, it is important to develop algorithms for efficiently calculating performance measures of interest.

Considerable effort has been expended on developing algorithms for closed product form queueing networks. Over time, some algorithms have been subsumed by others that are uniformly superior (e.g., RECAL [39], and MVAC [37] have been subsumed by DAC [48]). A sampling of the most useful current algorithms is given below. These algorithms differ in terms of:

- The computational complexity:

  The computational complexity of the algorithm is usually given in terms of the number of centers, the number of chains and the number of customers in each chain. Often the complexity is not easily calculable in such terms. For example, the complexity of the MVA algorithm (see below) grows as 2^N where N is the number of state dependent servers. No closed form expression exists for the Tree Convolution Algorithm complexity which is highly dependent on the number of chains in the model and "topology" of the network (e.g., the fraction of centers visited by the average chain).

- The performance measures that can be calculated by the method:

  Queue length statistics (mean, distribution), throughputs and utilization are the most common measures of interest. However, joint queue length distributions, derivatives of measures with respect to parameters, etc. are sometimes required. The algorithms are often limited to particular measures, e.g., MVA does not yield queue length distributions.

The following notation is used throughout the chapter:

J	=	number of service centers.
K	=	total number of chains in the network.
N_k	=	population of chain k.
$\vec{N}$	=	population vector $= (N_1, \ldots, N_K)$.
$j(k)$	=	a specified service center visited by chain k.
θ_{jk}	=	visit ratio of a chain k customer to center j, scaled so that $\theta_{j(k)k} = 1$.
T_{jk}	=	mean service time of a chain k customer at center j.
a_{jk}	=	$\theta_{jk} \cdot T_{jk}$, relative utilization of a chain k customer at center j.
$\mu_j(n)$	=	service rate of service center j when there are n customers present, $\mu_j(1) = 1$.
n_{jk}	=	number of chain k customers at center j.
n_j	=	$\sum_{k=1}^{K} n_{jk}$ = number of customers at center j.
$\vec{n}_j$	=	$(n_{j1}, \ldots, n_{jK})$ = state of center j.
$\underline{\vec{n}}$	=	$(\vec{n}_1, \ldots, \vec{n}_J)$ = state of the network.
λ_{jt}	=	$\theta_{jt} \cdot \lambda_t$ = throughput of chain t customers at center j, where $\lambda_t = \lambda_{j(t)t}$.
L_{jt}	=	mean number of chain t customers at center j.

W_{jt}	=	mean waiting time (queueing time + service time) of a chain t customer at center j.
$\underline{S}$	=	$\{\vec{n} \mid \sum_{j=1}^{J} \vec{n}_j = \vec{N}\}$ = state space of the network.
$P(\vec{n})$	=	steady state probability that the network is in state $\vec{n} \in \underline{S}$.
$P_j(n)$	=	steady state probability that there are n customers at service center j.
$\vec{\epsilon}_k$	=	K-dimensional vector whose k-th element is one and whose other elements are zero.
FCFS	=	First Come First Served service centers.
SSFR	=	Single Server Fixed Rate service centers.
IS	=	Infinite Server service centers.
QLD	=	Queue Length Dependent service centers (the dependence is in the total queue length at a center).

The product form solution can be immediately used in the calculation of state probabilities of queue lengths. However, for closed networks, we need to evaluate the normalization constant which appears in the solution. For a product form single closed chain network with SSFR centers only, the equilibrium state probabilities are given by:

$$P(n_1, \ldots, n_J) = \frac{1}{G(N)} \prod_{j=1}^{J} a_j^{n_j} \tag{2}$$

and $G(N)$ is the normalization constant determined so that the state probabilities sum to one:

$$G(N) = \sum_{n_1 + \ldots + n_J = N} \prod_{j=1}^{J} a_j^{n_j} \tag{3}$$

Since the summation is over the set of all feasible states, it is easy to see that the computation of $G(N)$ requires $O\left[\binom{N+J-1}{J-1}\right]$ operations. However, Buzen [26] showed that there is a simple recursive algorithm which computes not only $G(N)$, but the set $G(1), \ldots, G(N)$ with much less cost than taking the above sum. Furthermore, after computing $G(n)$, $n = 1, \ldots, N$, it is necessary to determine the performance measures of interest. Calculation of common performance measures is discussed below based on the generating function approach introduced by Moore [132] and further extended by Williams and Bhandiwad [199].

Consider the following polynomial which is called the generating function for the network:

$$\begin{aligned} g_j(t) &= x_1(t) \cdot x_2(t) \ldots x_j(t) \\ &= (1 + a_1 t + a_1 t^2 + \ldots)(1 + a_2 t + a_2 t^2 + \ldots)(1 + a_j t + a_j t^2 + \ldots) \\ &\quad \textit{for } 1 \leq j \leq N \end{aligned} \tag{4}$$

Call the coefficient t^n in $g_j(t)$, $G_j(n)$. Note that the coefficient t^N in $g_J(t)$ is equal to the normalization constant $G(N)$ $(= G_J(N))$, since $G_J(N)$ is the result of the sum of terms $a_1^{n_1} a_2^{n_2} \ldots a_J^{n_J}$ such that $\sum_j n_j = N$. Furthermore, $G_j(n)$ is simply the convolution of $G_{j-1}(n)$ with the function $f_j(n) = a_j^n$,

$$G_j(n) = G_{j-1}(n) \otimes f_j(n) \stackrel{\text{def}}{=} \sum_{i=0}^{n} f_j(i) G_{j-1}(n-i) \qquad 0 \leq n \leq N$$

Using the property that:

$$x_i(t) = \frac{1}{1 - a_i t} \tag{5}$$

we have

$$g_j(t) = \frac{g_{j-1}(t)}{1 - a_j t}$$

and so,

$$g_j(t) = g_{j-1}(t) + a_j t g_j(t) \tag{6}$$

Now, noting that the coefficient t^n in $tg_j(t)$ is the coefficient t^{n-1} in $g_j(t)$, we have the following recursive formula:

$$G_j(n) = G_{j-1}(n) + a_j G_j(n-1) \tag{7}$$

where $G_j(0) = 1$. Clearly $G_J(N)$ can be evaluated in $O(NJ)$ operations. (This expression could also be obtained directly from the convolution equation, but the above development serves to introduce the generating function technique.)

From the values of $G_J(n)$, several performance measures can be evaluated. For instance, the utilization of device i is obtained from Eq. (2):

$$U_i = \sum_{\substack{n_1 + \ldots + n_J = N \\ n_i > 0}} P(n_1, \ldots, n_J) \tag{8}$$

If we multiply $g_J(t)$ by $a_i t$, the coefficient t^N in $g_J(t).a_i t$ $(a_i G_J(N-1))$ gives the sum of all terms of the form $a_1^{n_1} a_2^{n_2} \ldots a_J^{n_J}$ such that $\sum_j n_j = N$ and $n_i > 0$. Therefore, the utilization U_i of device i is given by:

$$U_i = \frac{a_i G_J(N-1)}{G_J(N)} \tag{9}$$

Other performance measures can be obtained in a similar way.

For multiple closed chain product form queueing networks with SSFR and QLD, a similar recursive formula is obtained [116,148]:

$$G_j(\vec{n}) = G_{j-1}(\vec{n}) \otimes f_j(\vec{n}) = \sum_{\vec{i}=\vec{0}}^{\vec{n}} f_j(\vec{n} - \vec{i}) G_{j-1}(\vec{i}) \qquad \vec{0} \le \vec{n} \le \vec{N} \tag{10}$$

where

$$f_j(\vec{n}) = \frac{|\vec{n}|!}{\prod_{m=1}^{|\vec{n}|} \mu_j(m)} \prod_{k=1}^{K} \frac{a_{jk}^{n_k}}{n_k!}$$

and $\mu_j(m) = 1$ or m for SSFR or IS centers, respectively. Eq. (10) simplifies to:

$$G_j(\vec{n}) = G_{j-1}(\vec{n}) + \sum_{k=1}^{K} a_{jk} G_{j-1}(\vec{n} - \vec{e}_k) \qquad \vec{e}_k \le \vec{n} \le \vec{N} \tag{11}$$

if the networks have SSFR and IS centers only. The normalization constant $G_J(\vec{N})$ is obtained in $O(J \prod_k \frac{(N_k+2)(N_k+1)}{2})$ or $O(JK \prod_k (N_k+1))$ operations if Eq. (10) or Eq. (11) is used, respectively.

An interesting result for product form queueing networks is the so called "Norton's theorem" [28]. For single chain networks, this theorem states that a subnetwork of resources of a product form queueing network model can be replaced by a QLD service center such that the queue length distributions for each resource in the remaining network are identical to those in the original network. The name "Norton's theorem" comes from the way the service rates of this equivalent center are obtained. The subnetwork is solved in isolation with the resources in the subnetwork's complement "shorted", i.e., the service rates of these resources are set to infinity. The service rate $\mu(n)$ is set equal to the throughput rate $\lambda(n)$, of this subnetwork with customer population n. As an example, consider Figure 4a where a subnetwork is formed by queues q_2 and q_3. In Figure 4b queue q_1 in the subnetwork's complement is "shorted"

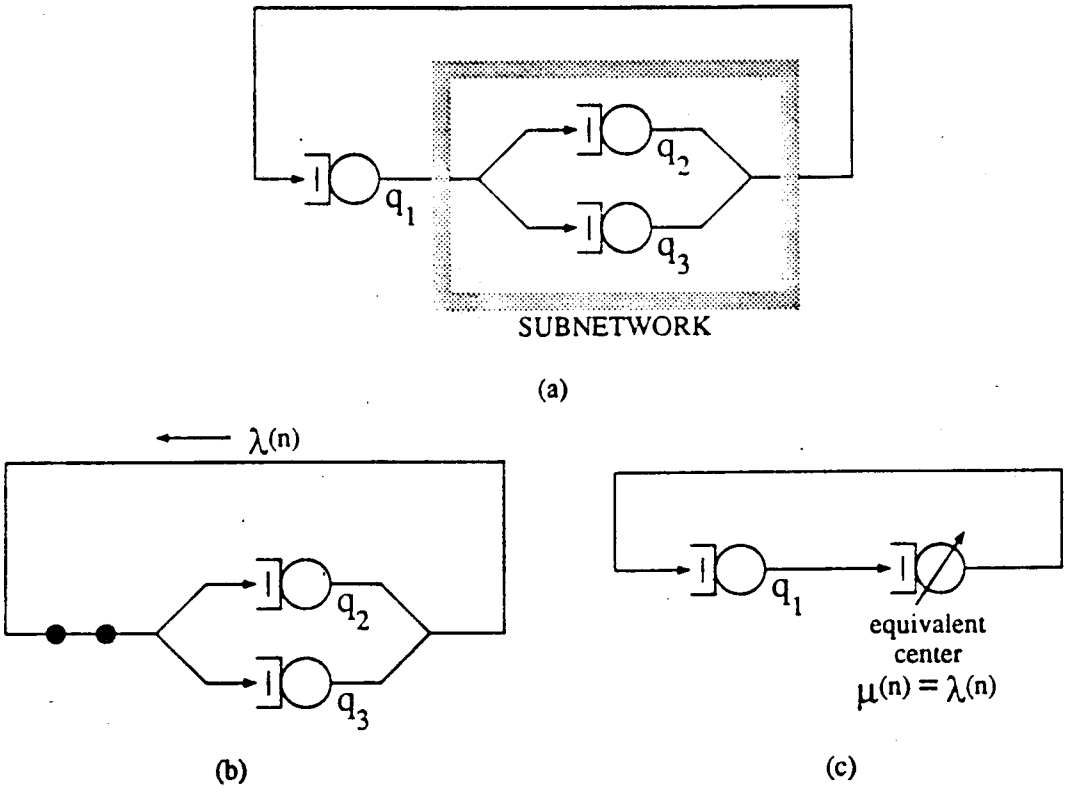

Figure 4: Norton's theorem.

in order to obtain the Norton equivalent QLD center. In this figure, $\lambda(n)$ is the throughput of the network when the chain population is n. Figure 4c shows the resulting network. This replacement is particularly useful for performing parametric analysis. In a parametric study, some subset of parameters is varied. Using Norton's theorem the part of the network model that is not being varied can be replaced by an equivalent QLD server and computational savings can be realized. It is interesting to note that replacing a subnetwork by its Norton's equivalent center corresponds to the convolution algorithm in which the queues in the subnetwork are first in the order of convolution. Perhaps the major use of Norton's theorem is as a heuristic to develop approximations for non-product form networks. The results are also valid for multiple chain networks [105].

The convolution algorithm represented a major step forward in queueing network modeling. However, it has the drawback that floating point overflow/underflow may

occur during the computation of the normalization constant. This problem was addressed by Lam [110] who proposed an approach which alleviates the numerical problem, but at the expense of increased computational cost. Other algorithms followed, such as LBANC [31], but with no significant advantages [116].

Reiser and Lavenberg [149] developed a new computational algorithm which they termed Mean Value Analysis (MVA), since it allows the computation of mean performance measures (throughputs, mean queue lengths, mean response times) without the need to compute the normalization constant. The computational cost is on the same order of magnitude as the convolution algorithm for SSFR and IS service centers, but it is higher than convolution for QLD centers. The algorithm does not suffer the numerical problems of convolution. Furthermore, the equations obtained are simple and have a physical interpretation which has made MVA the most popular algorithm for queueing networks. The physical interpretation has been the basis of a number of approximations as we will survey in the next section.

For SSFR and IS centers, the algorithm is based on three equations. The first is obtained by applying Little's result to a service center:

$$L_{jk}(\vec{N}) = \lambda_k(\vec{N}) a_{jk} W_{jk}(\vec{N}) \tag{12}$$

and the second by applying Little's result to the customers belonging to a routing chain:

$$N_k = \sum_{j=1}^{J} L_{jk}(\vec{N}) = \sum_{j=1}^{J} \lambda_k(\vec{N}) \theta_{jk} W_{jk}(\vec{N})$$

and so,

$$\lambda_k(\vec{N}) = \frac{N_k}{\sum_{j=1}^{J} \theta_{jk} W_{jk}(\vec{N})} \tag{13}$$

The last and key equation is obtained from the arrival theorem for closed queueing networks [96,118,163] which states that, for networks of the type given in [12], a customer arriving to a queue sees the distribution of the number of customers in that queue equal to the steady state distribution for the network with one less customer of that type. This implies that:

$$W_{jk}(\vec{N}) = \begin{cases} T_{jk}[1 + \sum_{k=1}^{K} L_{jk}(\vec{N} - \vec{e}_k)] & \text{for SSFR centers} \\ T_{jk} & \text{for IS centers} \end{cases} \tag{14}$$

These equations can be generalized to networks with QLD centers. Queue length dependent centers often occur and it is important to be able to include them in models. Common examples of such servers in computer systems modeling are I/O devices such as disks. Depending on the scheduler of a moving arm disk, the disk seek can differ significantly as a function of the number of requests queued. For QLD centers, the calculations must include expressions for the marginal state probabilities:

$$W_{jk}(\vec{N}) = T_{jk} \sum_{i=1}^{|\vec{N}|} \frac{i}{\mu_j(i)} P_j(i-1|\vec{N} - \vec{e}_k) \tag{15}$$

$$P_j(i|\vec{N}) = \frac{1}{\mu_j(i)} \sum_k a_{jk} \lambda_k(\vec{N}) P_j(i-1|\vec{N} - \vec{e}_k) \tag{16}$$

For numerical stability, $P_j(0|\vec{N})$ is calculated from the equation below:

$$P_j(0|\vec{N}) = \frac{\lambda_k(\vec{N})}{\lambda_k^{J-j}(\vec{N})} P_j(0|\vec{N} - \vec{e}_k) \tag{17}$$

which increases significantly the cost of the overall algorithm since $2^{\#\text{ load dependent centers}}$ networks have to be solved, each with some of the load dependent service centers removed from the network [116,187]. (In the equation above the superscript $J-j$ on the throughput is meant to indicate that the throughput is for a network which is identical to the original network except that center j is removed.) However, Heffes [76] has shown that, if the center service rate is given by a special function: $\mu(m) = \frac{k}{\alpha+\beta m}$ ($\alpha + \beta = 1$, $\beta \geq 0$), an efficient solution exists. By a proper choice of α and β, the modeler can try to match the queue dependent service rate of a device.

The MVA algorithm, as the name indicates, is appropriate for computing mean performance measures. However, it is possible to obtain similar recursions for moments of queue lengths. In [76], Heffes showed that moments of queue lengths can be obtained recursively in terms of lower moments using Eq. (16). De Souza e Silva and Muntz [53] showed that the recursion for calculating variance (and higher moments) can be obtained directly from the main MVA equation. For instance, a recursive expression for the variance $V_{jk}(\vec{N})$, of queue lengths of chain k customers at center j is obtained by: (a) taking partial derivatives of the performance measures of a slightly modified version of Eq. (14) with respect to input parameters (such as visit ratios), and (b) using simple relationships between these derivatives and mean queue lengths. One of these relationships is:

$$V_{jk}(\vec{N}) = \theta_{jk} \frac{\partial L_{jk}(\vec{N})}{\partial \theta_{jk}} \tag{18}$$

One then obtains:

$$\begin{aligned} V_{jk}(\vec{N}) = {} & \lambda_k(\vec{N}) a_{jk}[1 - L_{jk}(\vec{N}) + 2L_{jk}(\vec{N} - \vec{e}_k)] + \\ & \sum_{t=1}^{K} \lambda_t(\vec{N}) a_{jt}[(L_{jk}(\vec{N} - \vec{e}_t) - L_{jk}(\vec{N}))L_{jk}(\vec{N} - \vec{e}_t) + V_{jk}(\vec{N} - \vec{e}_t)] \end{aligned} \tag{19}$$

The relationships between partial derivatives of mean performance measures and moments of queue lengths have also been applied in optimization problems [46]. Similar results for moments of total queue lengths in a center were also presented in [169]. McKenna and Mitra [126] obtained a general expression which relates higher (> 1) factorial moments of queue lengths with higher order partial derivatives of the generating function with respect to relative utilizations. As a consequence, they obtained a recursive expression for calculating the derivatives in terms of lower order derivatives. The approach was used in conjunction with the asymptotic expansion method (to be surveyed in the next section) to obtain bounds for moments of queue lengths.

Many applications, such as communication networks, require models with many closed chains. Even for product form networks, an exact solution is not feasible due to the large computational costs of the algorithms above. Lam and Lien [112] observed that, in the case of communication networks and probably in other applications as

well, a given chain often visits only a small subset of the centers. With this motivation, they developed a convolution-based algorithm called *Tree Convolution* which takes advantage of this "sparseness" property when it exists.

Consider Eq. (10). Then:

$$G_J(\vec{n}) = f_1(\vec{n}) \circledcirc f_2(\vec{n}) \circledcirc \ldots \circledcirc f_J(\vec{n}) \tag{20}$$

Let $SUBNET_1$ be a subnet of m centers in the network (e.g., centers 1 to m) and $SUBNET_2$ the complementary subset (e.g., centers $m+1$ to J). From Eq. (20),

$$G_J(\vec{n}) = G_{SUBNET_1}(\vec{n}) \circledcirc G_{SUBNET_2}(\vec{n}) \tag{21}$$

where

$$\begin{aligned} G_{SUBNET_1}(\vec{n}) &= f_1(\vec{n}) \circledcirc \ldots \circledcirc f_m(\vec{n}) \\ G_{SUBNET_2}(\vec{n}) &= f_{m+1}(\vec{n}) \circledcirc \ldots \circledcirc f_J(\vec{n}) \end{aligned}$$

The key observation is to partition the set of chains in the network into three subsets, Φ_1, Φ_2 and Φ_3. Subset Φ_1 (Φ_2) contains all chains which visit only centers in $SUBNET_1$ ($SUBNET_2$) and subset Φ_3 contains all chains which visit centers in $SUBNET_1$ and $SUBNET_2$. Then, it is easy to see that if Φ_1 (Φ_2) is non-empty, $G_{SUBNET_1}(\vec{n})$ ($G_{SUBNET_2}(\vec{n})$) can be evaluated with less computational effort than if Φ_1 (Φ_2) was an empty set. This is because chains in Φ_2 (Φ_1) do not need to be considered when evaluating $G_{SUBSET_1}(\vec{n})$ ($G_{SUBSET_2}(\vec{n})$). Furthermore, $G_J(\vec{n})$ can also be evaluated with less computational effort if Φ_1 and Φ_2 are non-empty sets. The algorithm works by finding a sequence of subnets ($SUBNET_1$, $SUBNET_2$, ..., $SUBNET_i$, ...) such that Φ_i is non-empty and then merging these subsets into larger ones until all the network is covered. The tree approach can also be implemented using an MVA-based recursion. The details can be found in [84,188].

The computational requirements of both convolution and MVA algorithms grow exponentially with the number of closed chains in the network. Recently, Conway and Georganas [39] proposed a new algorithm called RECAL (Recursion by Chain Algorithm) with computational requirements which grow as a polynomial function of the number of closed chains in the network. RECAL is significantly less costly than either convolution or MVA for networks with many closed chains and few centers, but the opposite is true for networks with many service centers and a few closed chains. Similar to convolution, RECAL is based on a recursive computation of the normalization constant and suffers from the same potential overflow/underflow problems that are found with convolution. Subsequently, Conway *et al* [37] derived a new algorithm called MVAC (Mean Value Analysis by Chain) that is similar in form to RECAL's recursion (and has similar computational requirements) but avoids the computation of the normalization constant. Like MVA, MVAC computes only mean performance measures for networks with IS and SSFR centers. Furthermore the equations used have a pleasing probabilistic interpretation.

As we will discuss in following sections, there are applications which require the calculation of joint distributions of queue length. Motivated by such applications, de Souza e Silva and Lavenberg [48] developed a new algorithm called DAC (Distribution

Analysis by Chain) which has been shown to be more efficient than previous algorithms for that purpose, since the computational cost to evaluate joint queue length probabilities with DAC is of the same order as the number of such probabilities [48]. DAC can be applied to networks with IS, SSFR and QLD centers. Furthermore, DAC can be used to compute mean performance measures and marginal queue length probabilities more efficiently than RECAL or MVAC. Finally, the recursion is very simple, the equations obtained have a physical interpretation and the computations present no numerical problems.

To illustrate the DAC recursion, consider a network with QLD centers in which each chain with $N_k > 1$ customers is converted to N_k identical single customer chains (with the same visit ratios, etc. as the original chain). We define some additional notation in which superscript k denotes a quantity for a network when only chains $1, \ldots, k$ are present. Thus, the superscript K denotes the network with a full population.

$\vec{n}$	=	$(n_1, \ldots, n_J)$ = joint queue length.
$\vec{e}_j$	=	J-dimensional vector whose j-th element is one and whose other elements are zero.
$\vec{1}^k$	=	population vector = k-dimensional vector all of whose elements are one.
$\underline{S}^k$	=	$\{\vec{n} : \sum_{j=1}^{J} \vec{n}_j = \vec{1}^k\}$ = state space of the network.
$P^k(\vec{n})$	=	steady state probability that the network is in state $\vec{n} \in \underline{S}^k$.
S^k	=	$\{\vec{n} : \sum_{j=1}^{J} n_j = k,\ n_j$ is a nonnegative integer for all $j\ \}$ = set of possible queue lengths.
$P^k(\vec{n})$	=	steady state probability that the joint queue length is $\vec{n} \in S^k$.
$P_j^k(n)$	=	steady state probability that there are n customers at service center j.

The recursion is based on the key equation:

$$P^k(\vec{n}) = \lambda_k^k \sum_{j=1}^{J} a_{jk} \frac{n_j}{\mu_j(n_j)} P^{k-1}(\vec{n} - \vec{e}_j) \tag{22}$$

where,

$$\lambda_k^k = \frac{1}{\sum_{j=1}^{J} \theta_{jk} W_{jk}^k} \tag{23}$$

$$W_{jk}^k = T_{jk} \sum_{n=1}^{k} \frac{n}{\mu_j(n)} P_j^{k-1}(n-1) \tag{24}$$

Each step of the recusion corresponds to adding one customer to the network.

To allow easier interpretation of Eq. (22) we rewrite it as:

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k \left[\frac{n_j}{\mu_j(n_j)} \frac{T_{jk}}{W_{jk}^k} P^{k-1}(\vec{n} - \vec{e}_j) \right] \tag{25}$$

Since the network is composed of single customer closed chains, L_{jk}^k is the probability that the single customer from chain k is at center j, when k chains are present. It

can be shown (using similar arguments to those presented in appendix A of [37]) that the term in brackets in Eq.(25) is the expression for the steady state probability of a network with k chains being in state $\vec{n}$, conditioned that the customer from chain k is at center j. (Furthermore, this conditional probability is identical to the steady state unconditional probability of a network where the k^{th} customer is circulating at center j only.)

The recursion presented above calculates the joint queue length probabilities and average measures for the last chain added to the network. The mean performance measures for any other chain, say chain t, can be calculated by "backtracking", i.e., by removing the chain t customer from the network.

In [48] extensions to the algorithm were presented which reduce the computational requirements when only marginal probabilities (or joint probabilities at a subset of centers) are needed. Furthermore, a simple recursive expression is obtained which allows the calculation of any state probability in K steps only, if chain throughputs are previously calculated using any algorithm, say MVA or DAC.

Summary.

In this section, we presented a brief survey of computational algorithms for closed queueing networks. Mixed networks, i.e., networks with closed and open chains, offer no additional problem. The approach is simply to convert the mixed network into an equivalent network with only closed chains. For networks with IS and SSFR, the effect of open chains is to reduce the capacity of the service centers by a fixed amount [116]. Several books exist which include a detailed presentation of algorithms, such as [21,31,38,116].

We conclude this section with several observations. Although some of the exact algorithms have been applied in practice, they are not practical for solving large network models, mainly due to the large amount of storage required. (The issue of reducing space requirements of exact algorithms by implementation procedures have been addressed by several authors, e.g. [74,209].) Many applications require the solution of medium to large problems, and thus the modeler has to resort to approximation techniques. However, even in this case the exact algorithms play an important role, as will become evident in the next section. Furthermore, exact algorithms based on equations which have a probabilistic interpretation are particularly useful as a basis for developing heuristics for approximation algorithms. The prime example of this is MVA. It remains to be seen if DAC will be useful in a similar way. Finally, it is important to point out that care must be taken when one chooses a particular algorithm to solve a problem. It is common to see a reference to an exact algorithm as appropriate to solve "product form queueing network models". However, the correct phrase is "a class of product form queueing network models", since the equations obtained for the particular algorithm are often proven algebraically from expressions valid for a restricted class of product form models only. For example, recently a new kind of service center (of interest in modeling certain types of multiprocessor systems) was proven to satisfy the requirements for a product form solution [17]. However, the known algorithms cannot immediately be used to obtain a solution for models which include this new center, and new algorithms had to be developed [18,32].

4 Approximations.

4.1 Introduction.

As discussed in the previous section, open networks have a simple closed form solution. The exact analysis of general closed queueing networks is typically very costly due to the large state space. Closed networks require additional effort to obtain a normalized solution, and relatively efficient algorithms have been developed as were described. Unfortunately, these algorithms have computational requirements that grow combinatorially with the number of closed chains in the network or with the number of centers. For example, for the most common algorithms (convolution, MVA) networks with more than five chains with a few customers per chain (say ten) may be too costly to solve in a medium size computer. Of course increases in the speed and memory capacities of computer systems will expand the range of models that can be solved exactly but the exponential growth of complexity of the computations means that only marginally larger networks will come within reach. Networks with a larger number of chains can be solved exactly if the network has a special sparseness structure (e.g., with tree algorithms) or if it has only a few centers (e.g., with RECAL, MVAC or DAC). For networks which do not satisfy product form requirements, the only general solution method is to determine the transition rate matrix for the model and to solve numerically. Due to the state space explosion this is only feasible if the network has a very small number of service centers and jobs, i.e., a small state space. The high cost of existing algorithms for product form closed chain networks and the lack of efficient exact algorithms for non-product form networks are the main reasons to investigate accurate and cost effective methods for the approximate analysis of closed queueing networks.

Many methods have been proposed for the approximate analysis of queueing networks. Most do not give error bounds for the results obtained. In general, the accuracy of the method is evaluated empirically for a large number of networks by comparing the approximate solutions with results obtained by analyzing the same networks with an exact algorithm (if that is feasible) or by simulation.

We classify the techniques for the approximate analysis of queueing networks (both product form and non-product form networks) into non-iterative and iterative methods. Most of the non-iterative methods proposed in the literature are based on extensions to the MVA equations, hierarchical decomposition and/or Norton's theorem and asymptotic expansions of the normalization constant. Some of the non-iterative MVA based methods for non-product form networks require a recursive solution over all network population vectors, and so they have the same multiple chain computational limits as MVA. MVA based approximations for product form networks generally break the recursion of the MVA equations in order to obtain significant computational savings. Non-iterative methods based on hierarchical decomposition/Norton's theorem typically involve solving a subnetwork for all feasible population vectors (to obtain the service rates for a flow equivalent server) and then solving an aggregate network consisting of the flow equivalent server and the remainder of the network. These methods are more suited to single chain networks rather than multiple chain networks due to the cost of solving the subnetworks and the aggregated network. A

variety of iterative methods have been proposed in the literature. A survey can be found in [50]. Roughly, in some methods, sets of nonlinear equations are obtained by introducing approximate relationships between unknown parameters of the network. There may be two (or more) sets of equations and two (or more) corresponding sets of parameters (say, X_1 and X_2). The equations take the form: $X_1 = F_1(X_2)$ and $X_2 = F_2(X_1)$, which can be reduced to a single set of fixed point equations of the type $X_1 = F_1(F_2(X_1))$. In general, we may have N sets of equations of the form $X_i = F_i(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$. Others follow the classical iterative decomposition approach in which a model is decomposed into an interrelated collection of subnetworks. Each subnetwork is analyzed independently, with the remainder (or complement) of the subnetwork represented in a simplified way. The parameters of the complement of a subnetwork are obtained from the solution of other subnetworks. Each subnetwork is analyzed, one at a time, and the parameters in the complement are updated to reflect the latest subnetwork solution. Successive iterations refine previous estimates, until some convergence criterion is satisfied. Figure 5 below illustrates this iterative process. Besides the accuracy of the result, issues concerning

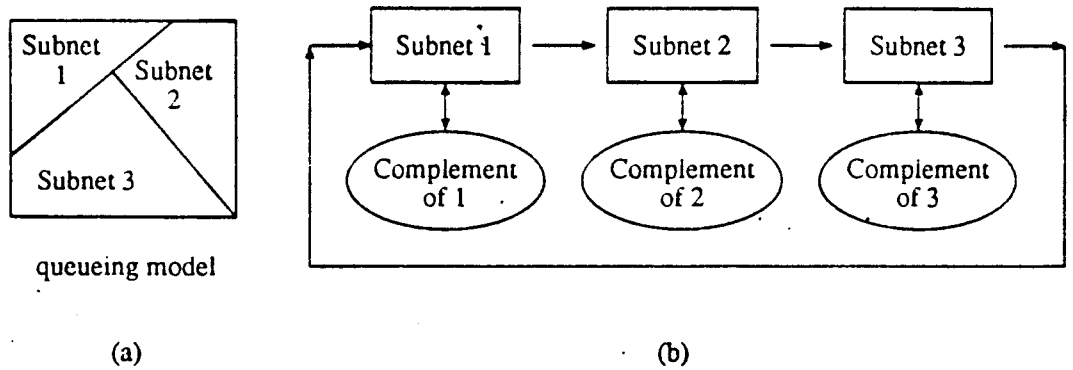

Figure 5: An iterative process.

convergence, existence and uniqueness of the final solution are important, but in general it is difficult to resolve any of these issues. Several researchers have addressed the problem, e.g. [6,50,139,206].

We have outlined one classification of approximation techniques. In this section we survey several specific approximations for product and non-product form networks and indicate where they apply. These approximations have been developed in response to needs in practical applications as will be discussed for each.

4.2 Decomposition.

Many approximations are based on hierarchical decomposition in which the network is decomposed into subnetworks which are solved independently. The results are then combined to obtain a solution for the original network. Decomposition is a useful technique in analyzing large complex systems consisting of weakly coupled subsystems. The basic theorems formalizing the approach for stochastic models are due to Simon and Ando [165]. Courtois made a seminal contribution in showing how the results apply to stochastic queueing networks and developed a comprehensive theory of nearly completely decomposable queueing networks. The development below

follows that of Courtois [40,41]. We briefly survey the application of the general technique to queueing networks.

Consider a queueing network model of a system with exponential servers. Assume that this model is composed of weakly coupled subsystems, i.e., the queues in any subsystem interact much more heavily with queues in the same subsystem than with queues in the remainder of the network. Then it may be possible to construct a transition rate matrix of the model such that it is nearly completely decomposable. This matrix consists of state aggregates. Each aggregate is such that the transition rates from states in one aggregate to another are small compared to the transition rates between states in the same aggregate. Courtois has shown that for such weakly coupled queueing network models the aggregates are the transition matrices of subnetworks of queues. This observation is of great importance in providing computational savings. The basic solution technique is as follows: each aggregate is solved in isolation, obtaining an approximate solution for the conditioned state probabilities of each; then the steady state probabilities for each aggregate are calculated by accounting for the interaction between aggregates.

In more detail, assume that the behavior of the system is represented by a continuous time Markov chain with transition rate matrix Q as shown in Figure 6. In that

$$Q = \begin{bmatrix} Q_1 & \epsilon & \cdots & & \epsilon \\ \epsilon & Q_2 & & & \vdots \\ \vdots & & \ddots & & \\ & & & Q_{N-1,N-1} & \epsilon \\ \epsilon & \cdots & & \epsilon & Q_N \end{bmatrix}$$

Figure 6: Transition rate matrix Q.

figure, Q_i is a submatrix consisting of the transition rates between states of aggregate i. The ϵ's indicate that the transition rates from a state in aggregate i to states in any other aggregate are small, in comparison to the transition rates between states in each aggregate (less than a value ϵ). It is easy to see that a matrix C can be chosen such that

$$Q = Q^* + \epsilon C \tag{26}$$

where Q^* is a completely decomposable matrix. π_i^*, the solution of $\pi_i^* Q_i^* = 0$, can be interpreted as an approximation to the steady state probability vector conditioned that the system is in aggregate i ($\pi_i^* \approx \pi_i^c$). In order to find the unconditioned steady state probabilities, we need to calculate the percentage of time the system is in any of the states of each aggregate. If we knew the correct π_i^c this could be easily done by *exactly aggregating* each submatrix Q_i into one state and constructing and solving a new matrix R, shown in Figure 7, with N states, where the r_{ij}'s are calculated from

$$r_{ij} = \pi_i^c Q_{ij} \vec{1} \tag{27}$$

where $\vec{1}$ is a column vector of 1's.

$$R = \begin{bmatrix} \bullet & r_{1,2} & r_{1,3} & \cdots & r_{1,N} \\ r_{2,1} & \bullet & & & \vdots \\ r_{3,1} & r_{3,2} & \bullet & & \\ \vdots & & & \ddots & \\ r_{N,1} & \cdots & & & \bullet \end{bmatrix}$$

Figure 7: Aggregate matrix R.

The π_i^e's are not known exactly, but they can be approximated by the π_i^* s. The solution of $\Psi = \Psi R$ ($\Psi = [\psi_1, \psi_2, \ldots, \psi_N]$) is used to obtain the final approximate solution π:

$$\pi = [\pi_1^* \psi_1, \pi_2^* \psi_2, \ldots, \pi_N^* \psi_N] \tag{28}$$

This solution can be shown to be $O(\epsilon)$[42].

In [190] Vantilborgh extended the above results and showed that the solution for the steady state probabilities can be bounded by $O(\epsilon^2)$ as follows. It is known that for a specific choice of the matrix C in Eq. (26), each π_i^* will correspond exactly to π_i^e. However, in order to make this exact aggregation, it is necessary to know the exact value of π. The basic idea is to estimate π, as above, and again use Eqs. (27) and (28) to obtain a new estimate. This new estimate can be shown to be $O(\epsilon^2)$ and successive estimates obtained by iterating the procedure will produce successively better bounds and will in fact, converge to the exact solution.

As an example of the application of decomposition to queueing networks, consider the model of a multiprogrammed computer system with memory constraints shown in Figure 8, where a memory partition has to be allocated to a job before it is allowed to compete for the CPU and disks. In the figure, the allocation of memory is repre-

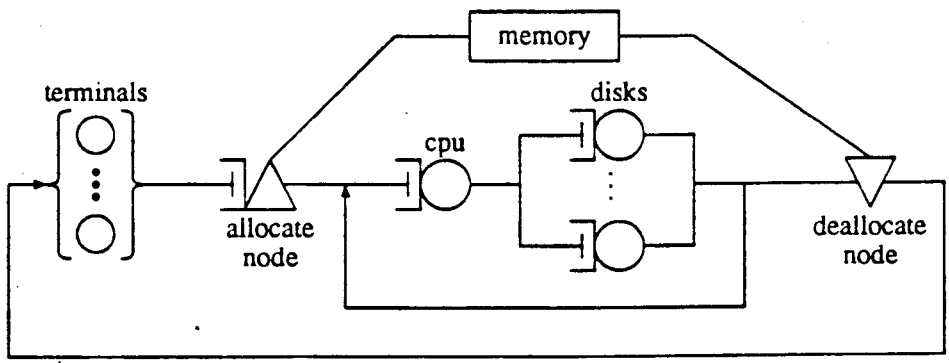

Figure 8: A central server model with memory constraint.

sented by an "allocate node" [116,152,154], and we assume that there are M memory partitions and $N > M$ terminal users. Let Q be the transition rate matrix for the system. Assume that the states are ordered in such a way that: (a) the i^{th} aggregate, for $i \leq M$, contains all states with i jobs in the CPU-I/O subnetwork and; (b) the j^{th} aggregate, $j > M$ contains all states with M jobs in the CPU-I/O subnetwork and

$j - M$ jobs waiting for a memory partition. Further assume that $N\lambda << \min\{\mu_c, \mu_d\}$, where $1/\lambda$ is the average user think time and μ_c and μ_d are the CPU and disks service rates, respectively. This assumption is reasonable for real system models. By properly choosing the matrix C, each aggregate Q_i^* will correspond to the subnetwork formed by the CPU and disks, when the population in this subnetwork is i, for $i \leq M$ or M for $i > M$. Therefore, the aggregates can be efficiently solved. Unfortunately, the approach in [190] destroys the physical interpretation above, since the submatrices used to obtain a new estimate of π do not necessarily represent networks of queues.

It is interesting to note that, based on decomposition, one can derive Norton's theorem for product form queueing network models [191]. As a final note, not only has decomposition been applied to queueing network models of weakly coupled subsystems but, as will be discussed next, the general idea of decomposing a system into an interrelated set of subnetworks can be used even when subnetworks are tightly coupled.

4.3 Approximations for Large Product Form Networks.

As mentioned in the introduction of this section, we can classify the majority of the methods that have been proposed for the approximate analysis of large product form queueing networks as either: (a) non-iterative methods based on extensions to the main MVA equations; (b) non-iterative asymptotic expansion methods; (c) iterative methods which involve the approximate solution of the whole network; (d) iterative methods based on hierarchical decomposition which involve the solution of subnetworks. Based on this classification, we survey below some of the methods that has been proposed.

4.3.1 Non-Iterative MVA Based Methods.

A simple example of a non-iterative approximation is the asymptotic bounds of Muntz and Wong [134] for single chain networks. The upper bound for utilizations and lower bound for mean response time for a class of queueing networks (single chain) applies more broadly than product form networks and includes arbitrary work conserving queueing disciplines, arbitrary service time distributions, etc. It can be shown that the increase of a device utilization with an increase in the number of customers is no greater than linear and that the ratio of the utilization of any two servers is equal to the ratio of their respective relative utilizations (for single server queues). These simple facts plus the observation that no utilization can exceed 1 lead to the following upper bound on device utilization:

$$U_i(N) \leq \min\left(\frac{a_i}{\sum_j a_j} N, \frac{a_i}{a_S}\right) \tag{29}$$

where $U_i(N)$ is the actual utilization of the i^{th} server when the network population is N, a_i is the relative utilization of the i^{th} server, and a_S is the highest relative utilization. An upper bound on response time can also be obtained by using Little's result. The bounds are also trivially extended for multiserver queues. The bounds above can be used as a rough guideline for system analysis. In [54], there are several examples which illustrate the application of asymptotic analysis.

Although the bounds above require few assumptions, a penalty is that they are not very tight. Zarhojan *et al* [208] developed a bounding technique called BJB (Balanced Job Bounds) which leads to more accurate bounds, but the method requires the network to be product form and only SSFR queues are permitted. It is shown that bounds on the throughput (response time) can be obtained by calculating the throughput (response time) of two related networks; one in which the relative utilizations of all devices are identical to the maximum relative utilization, and another in which all the relative utilizations are identical to the average of the relative utilization of all devices:

$$\frac{N}{\sum_j a_j + (N-1)a_S} \leq \lambda(N) \leq \frac{N}{\sum_j a_j + (N-1)a_a} \tag{30}$$

where $a_a = \frac{\sum_j a_j}{J}$. Extensions for networks with IS centers are also described in [208], but the solution is less efficient.

In [60,61] Eager and Sevcik proposed a bounding method which has advantages over previous techniques. This method can handle multiple chain networks with IS and SSFR centers, and one can trade off computation for accuracy, thus creating a hierarchy of bounds. The method is appropriately called Performance Bound Hierarchies (PBH). We briefly summarize the results. From the MVA equations for single chain networks we obtain:

$$W_j^{(i)}(N) = a_j \left[1 + \frac{W_j^{(i-1)}(N-1)}{Z + W^{(i-1)}(N-1)}(N-1) \right] \tag{31}$$

where Z is the sum of the relative utilizations in all IS centers, $W^{(i)}(N) = \sum_j W_j^{(i)}(N)$, and the superscript (i) indicates the order of the bound. If $W_j^{(i-1)}(N-1)$ is known exactly for all j, then Eq. (31) gives the exact value for $W_j(N) = W_j^{(i)}(N)$. However, if an upper (lower) bound is chosen for $W_j^{(0)}(N-i)$, then by applying Eq. (31) i times successively, $W_j^{(i)}(N)$ will also be an upper (lower) bound for $W_j(N)$, for any value of i. Also, the larger the value of i, the tighter the bounds. In [61] it is shown that, starting with Muntz and Wong's lower bounds for response time, a sequence of lower bounds is obtained. Similarly, a sequence of upper bounds is obtained when it is assumed that all customers are in the bottleneck center, i.e., $W_j^{(0)}(N-i) = N-i$ for $j = S$.

The basic idea of the algorithm remains the same for queueing networks with multiple closed chains. However, exact MVA equations are not used as in the single chain case, and new (MVA based) equations are employed. The computational cost of the PBH algorithm is $O\left(JK\binom{K+i-1}{i}\right)$. For a given value of i, bounds are less tight for less balanced fixed rate centers and more "dissimilar" chains. In [60] extensions are also developed for queueing networks with QLD centers, but the computational requirements grow combinatorially with the number of those centers.

Recently, Hsieh and Lam [88] proposed a bounding technique called Proportional Bounds, which is proven to be more accurate than BJB in [208]. The approach also enables the trading of computational time for accuracy and, for single chain networks, extends the technique in [61], since the recursion can be over an arbitrary

population sequence rather than $N-i$ through N. The method is based on the following observation: as the population of the network increases, the increase in the ratio L_i/L_j is greater than a linear increase with rate a_i/a_j, when $a_i > a_j$. If we assume that the rate of increase is linear (in particular, proportional to the ratio of relative visit ratios), then using the main MVA Eq. (14) a lower bound can be obtained for the total network delay. Similarly, an upper bound can be obtained by noting that $L_i(n)/L_j(n) < (a_i/a_j)^n$, for $a_i > a_j$, $n > 1$. For instance, bounds on $W(N)$ are given by (assuming the network has only SSFR centers):

$$\sum_{j=1}^{J} D_j^l(n_i) \leq W(n_i) \leq \sum_{j=1}^{J} D_j^u(n_i) \tag{32}$$

$$D_j^{l(u)}(n_i) = a_j[1 + r_j^{l(u)}(n_{i-1})(n_i - 1)]$$

$$r_j^{l(u)}(n_i) = \frac{D_j^{l(u)}(n_i)}{\sum_{m=1}^{J} D_m^{l(u)}(n_i)}$$

$$r_j^l(n_0) = \frac{a_j}{\sum_i a_i} n_0 = 1 \qquad r_j^u(n_0) = \left[\frac{a_j}{\sum_i a_i}\right]^{n_0} n_0 = n_1 - 1$$

where the superscript $l(u)$ represents the quantity for lower or upper bound, and $1 < n_1 < \ldots < n_S$ is any population sequence ($S \leq N$).

For multiple chain networks, the algorithm proposed in [88] is similar to the level 2 bounds in [60]. In [88] several examples are also presented which apply the multichain algorithm to medium size communication networks.

In a subsequent paper, Hsieh and Lam [87] proposed a simple approximation for multiple chain networks called PAM (Proportional Approximation Method). The method uses the MVA equations and estimates average queue lengths for population $(\vec{N} - \vec{e}_k)$ for $1 \leq k \leq K$ to obtain performance measures for population $\vec{N}$. The estimates are based on assuming the customers from each chain are distributed among the centers in the same proportion as the relative utilizations of the centers. A slight modification is also proposed in which the last two steps of the MVA recursion are executed (instead of just one step) to improve accuracy. In [87] the PAM algorithm was tested for chain throughputs only. The approximation does not give error bounds, and the results seem to indicate that the accuracy is not nearly as good as existing iterative methods. However, it is argued that the accuracy is sufficient for communication network optimization problems with the advantage that the solution is very fast. The emphasis is on low computation cost since, in communication network design, a large number of models must be solved during a search of the design space.

Other related work for obtaining hierarchies of bounds is presented in [98,106,168, 170].

4.3.2 Asymptotic Methods.

Most of the methods surveyed thus far are based on the MVA equations. A different approach is to obtain bounds on performance measures by performing asymptotic expansions on the normalization constant. Computational savings are obtained by truncating the higher order terms in the expansion. The earliest work that we are

aware of that took this approach is that by Ferdinand [62] for the machine interference (or machine repair) model. The idea of performing asymptotic expansions on the normalization constant was also used by McKenna and Mitra [128,129,130,144] who developed an extensive and novel methodology to handle large closed queueing networks with IS, SSFR and QLD centers. In their method, Euler's integral is substituted for terms in the expression for the normalization constant of a network. For example, for a single chain network with $J-1$ SSFR centers and a single IS center (server J) (see also [195] for details):

$$G(N) = \sum_{\underline{n} \in \underline{S}} \frac{a_J^{n_J}}{n_J!} \prod_{i=1}^{J-1} a_i^{n_i} \tag{33}$$

Recalling Euler's integral:

$$n! = \int_0^\infty e^{-t} t^n dt \qquad\qquad n \geq 0$$

it easily follows that,

$$a^n = \int_0^\infty e^{-t} \frac{(at)^n}{n!} dt \tag{34}$$

Assuming that $a_J = bN$ and $0 < a_i < b$ (for $1 \leq i \leq J-1$) (this is called the "normal usage" assumption), and using Eq. (34) to substitute for the $a_i^{n_i}$ in Eq. (33), after some algebraic manipulation and making an asymptotic expansion in one of the terms in $G(N)$, it is shown that $G(N)$ can be written as a linear combination of the normalization constant for hypothetical networks with SSFR centers called pseudo-networks, with relative utilization at the centers and chain population different that those for the original network. If higher order terms in the expansion are truncated, $G(N)$ can be estimated efficiently by solving the pseudo-networks with small population values for the pseudo-networks.

The computational cost of the algorithm (for multiple chain networks) is $O(JK^t)$, where t is the number of terms in the asymptotic expansion and is empirically chosen to be 4 [144] for accurate results. With this choice for t, the cost is similar to the cost of the PBH algorithm with bound level of 3 [61]. However, the asymptotic expansion algorithm requires that all chains visit an IS center and also imposes a limitation on the SSFR center utilizations relative to the total IS utilizations (e.g., for single chain networks, the normal usage assumption mentioned above requires that $U_i < U_J/N$). In practice, this limitation is translated into a restriction on the maximum utilization of the SSFR centers (say utilizations less than 0.85). Recently it was shown in [130] how the "normal usage" restriction can be relaxed if the centers at the pseudo-networks are treated as queue length dependent centers, even if the original network has no such centers. Finally, besides obtaining bounds for average quantities, bounds on second moments of queue lengths can also be obtained.

Other work related to asymptotic expansion techniques is found in [104] for solving a queueing network model of a multiprocessor system. For this simple system, approximate formulas are obtained for the performance measures of interest.

4.3.3 Iterative Methods.

The physical interpretation of the MVA equations has been the basis for a number of iterative approximations.

In some early work, Bard [11] proposed a very simple way to break the recursion in Eq. (14) by assuming:

$$L_{jk}(\vec{N} - \vec{e}_t) = L_{jk}(\vec{N}) \tag{35}$$

which has an intuitive appeal when the number of customers is large.

A more accurate and widely used approximation was proposed by Schweitzer [160] for product form networks with IS and SSFR centers:

$$L_{jk}(\vec{N} - \vec{e}_t) = \begin{cases} L_{jk}(\vec{N}) & k \neq t \\ \frac{N_k - 1}{N_k} L_{jk}(\vec{N}) & k = t \end{cases} \tag{36}$$

It is easy to see from Eq. (36) that the approximation comes from assuming that the fraction of each chain population at each center does not change if the network population is decreased by one customer. This approximation has computational costs which are $O(KN)$ and typically errors of less than 10% are obtained for throughputs and less than 20% for mean queue lengths and mean service times. However, large errors (up to more than 40% for average queue lengths) can occur [49,206]. This is due to the fact that the removal of a chain t customer can have nonnegligible effect on the average queue lengths of other chains, which is contrary to the assumption on which Eq. (36) is based.

In order to more accurately account for the effect of removing a job from the network, Chandy and Neuse [30] proposed a new algorithm called Linearizer. To illustrate the method, consider a single chain queueing network model. Define:

$$F_j(N) = \frac{L_j(N)}{N}$$

as the proportion of jobs at center j and

$$D_j(N) = F_j(N-1) - F_j(N)$$

as the difference in this proportion when the population increases by one job. Schweitzer's approximation assumes that the proportion of jobs at each center does not change from population $N-1$ to population N (i.e., $D_j(N) = 0$, for all j). (Note that this is true for balanced networks, i.e., networks in which all centers have the same relative utilization, and for unbalanced networks in the limit as $N \to \infty$ since almost all the customers will be in the bottleneck center.) Linearizer can be viewed as attempting to estimate the true value of $D_j(N)$. In order to break the recursion it is assumed that $D_j(N) = D_j(N-1)$. This is equivalent to assuming a linear change for $F_j(N)$ as N grows. Successive iterations are used to refine the value of $D_j(N)$. For multiple chain networks, it is necessary to estimate the difference $D_{kl_j}(\vec{N})$ in the proportion of chain k jobs at center j when the population of the network varies from $(\vec{N} - \vec{e}_l)$ to $(\vec{N})$. The basic approach remains the same, though. This algorithm is very accurate and extensive tests reported in the literature indicate that errors are typically less than one percent, and on the order of a few percent in the worst cases. Linearizer is

however, much more costly than Schweitzer's approximation. In [30] it was reported that the computational cost is $O(JK^3)$ and storage cost is $O(JK^2)$. In [52] it is shown that the computational cost of Linearizer can be reduced to $O(JK^2)$. Recently, an approximation based on Linearizer was proposed which has the same computational cost of $O(JK^2)$ but storage cost is reduced to $O(JK)$ [206]. Results in [206] indicate that the new approximation is only slightly worse than Linearizer. Usually, convergence is not a problem in these methods, but networks can be constructed such that convergence is slow [206].

Schweitzer's and Linearizer approximations cannot solve networks with queue length dependent centers. However, load dependent centers are important not only to model a congestion adaptive servers (e.g., disks), but they also occur in hierarchical modeling as a flow equivalent center for a subsystem. Zahorjan and Lazowska [207] proposed an approximate algorithm to solve networks with QLD centers. A simpler and apparently more accurate approximation was proposed by Krzesinski and Greyling [108]. The method is based on Linearizer and uses an additional approximation for QLD centers, namely:

$$P_j(i \mid \vec{N} - \vec{e}_k) \;=\; P_j(i \mid \vec{N}) \tag{37}$$

This approximation breaks the recursion of the MVA equation (Eq. (16)), and the estimates obtained for queue length distributions were found to be reasonably accurate. The computational complexity is $O(J(\sum_{k=1}^{K} N_k)K^2)$.

Combined iterative/decomposition methods.

The approximations above attain reduction in computational cost by replacing the MVA recursion by iteration. They do not attempt to decompose the model into smaller, more efficiently solved submodels. The main reason for partitioning a product form queueing network into subnetworks is to obtain subnetworks which are much less costly to analyze than the original network. The sum of the costs for solving each subnetwork times the number of iterations is usually significantly less than solving the whole network. There are many possibilities for choosing the subnetworks, for the representation for the complement for each chain in the original network, and how the parameters for the complement are determined. In [50,205] we find a detailed discussion of the possible choices.

Below is an example of an approximation which decomposes the original model into subnetworks and a simplified representation of the complement of each subnetwork. Successive iterations over the subnetworks refine the estimate of the parameters for the complement.

Models of local area distributed systems may require many closed chains (on the order of tens to hundreds). Briefly, in a local area distributed system, a job generated at one site may use resources from a foreign (i.e., remote) host. To represent the routing behavior of a job, at least one routing chain has to be associated with each site, and so models of LAN's with n sites will have at least n chains. Schweitzer's approximation is computationally practical and could be used to solve this kind of model. However, large errors may occur in some cases. Linearizer is very accurate, but it can be computationally too expensive for such models. An iterative approximation

method was proposed in [49] which is applicable to this problem. The basic idea is to subdivide the network resources into subnetworks (not necessarily disjoint) and, for each subnetwork, to identify the chains that have higher loadings at the subnetwork resources than at the resources in the complement. These chains will be called the local chains for that subnetwork, and their customers the local customers. All the other chains (and customers) will be called foreign (with respect to that subnetwork). For instance, in a local area distributed system, the subnetworks can be the computer sites, and the local chains for a site would most likely correspond to customers logged on at that site. Chains representing customers logged on at other sites would be the foreign chains. In this approximation, the effect of the complement in the subnetwork is represented in a different way for the two types of chains. For local chains the complement is represented by an IS center and for foreign chains the complement is represented by Poisson arrivals. As described in [49], this representation and the parameters for the complement are obtained by (a) preserving the recursion in Eq. (14) in a subnetwork when a local chain customer is removed, and (b) approximating the effect on the mean performance measures in a subnetwork (and the remainder of the network) when a foreign (local) chain customer is removed (for instance, Schweitzer's approximation is used to approximate the effect on the mean queue lengths of the resources in a subnetwork, when a foreign chain customer is removed). Figure 9 shows the representation of a subnetwork and its complement. The approach was

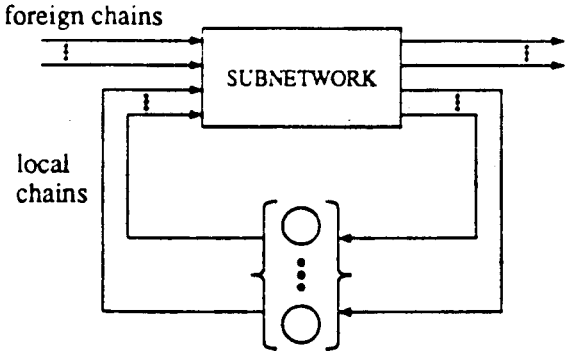

Figure 9: A subnetwork and its complement.

empirically found to be nearly as accurate as Linearizer but with a computational cost competitive with Schweitzer's approximation. It is interesting to note that any solution technique can be used to solve for the subnetworks, and if the original network contains only SSFR and IS centers and Schweitzer's approximation is used for the subnetworks, then the approach reduces to Schweitzer's approximation applied to the whole network. It can also be applied to networks with multiserver service centers. Furthermore, it is possible to trade off cost and accuracy by the choice of subnetworks and local chains. In [49] an example is also given which extends the method to non-product form networks.

Other iterative decomposition approaches are possible, and discussions can be found in [50,122,205].

4.4 Approximations for Non-Product Form Networks.

There are many applications that require the modeling of certain features which violate product form requirements when included in a queueing network model. Some examples include:

- FCFS queues with different service times for different routing chains. These queues appear in models of packet switching networks when packets in different virtual channels have different average lengths.
- In models of interactive computer systems, a job may be required to obtain a memory partition before competing for I/O resources. This is an example of the so called simultaneous resource possession problem. Other examples include the modeling of critical sections in a computer system.
- Queues with priority scheduling may appear both in models of data networks (e.g., control packets may have higher priority than data packets) or in a computer system model (e.g., jobs may have different priorities according to their class).
- Blocking. Blocking may occur when jobs contend for resources with finite waiting room. For instance, in data networks, a node with no free buffers available may block the traffic on incoming channels to that node.
- Queues with fork/join. Fork/join constructs are important to model the behavior of programs which can spawn tasks to be executed in parallel and need resynchronization at some points in time during execution.

Most of the solution methods proposed in the literature are tailored for a particular class of non-product form networks, and do not produce error bounds. An exception is the work of Courtois [40,41] which can be used in a more general framework and allows the bounding of the solution. However, it is limited to small models since it deals with the Markov chain representation of the system. Recently though, Courtois and Semal [43,45] have shown that queueing models with a very large number of states can be handled, by bounding performance measures of interest which are computable from the conditional steady state probability of certain subsets of states. This approach may considerably extend the size of models which can be analyzed. Another bounding methodology was recently proposed by van Dijk [189] and will be discussed below.

The majority of the methods for the approximate analysis of non-product form queueing network models can be subdivided into non-iterative and iterative methods based on heuristic extensions to the MVA equations, hierarchical decomposition and/or Norton's theorem.

4.4.1 MVA Based Methods.

FCFS centers with relaxed service time distribution constraints.

It is known that for FCFS service centers the product form solution holds only if all jobs belonging to different chains have the same exponential distribution. Even with

the usual "independence assumption" [102] models of packet switched communication networks with different packet lengths for different virtual circuits would violate product form. Reiser [147] proposed a heuristic modification to the MVA equation for the mean waiting times (Eq. (14)) for FCFS service centers with different service demand distributions for different chains. Based on Eq. (14), the waiting time observed by a packet of a chain k at service center j has three components, namely: (1) the mean service time of the arriving job at service center j, T_{jk}; (2) the mean backlog of jobs waiting in queue at arrival; (3) the residual service time of the job in service upon arrival of a chain k packet. The main assumption is that the arrival theorem holds for this kind of non-product form network. From the above assumption and observations the following equation is obtained:

$$W_{jk} = T_{jk} + \sum_{l=1}^{K} T_{jk} q_{jl}(\vec{N} - \vec{e}_k) + PB_j(\vec{N} - \vec{e}_k) \times \sum_{l=1}^{K} \frac{\lambda_l(\vec{N} - \vec{e}_k) a_{jl}}{\sum_{t=1}^{K} \lambda_t(\vec{N} - \vec{e}_k) a_{jt}} \frac{\overline{T_{jl}^2}}{2T_{jl}} \quad (38)$$

where $PB_j(\vec{N})$ is the probability that the center j server is busy when the population is $\vec{N}$. Eq. (38) simplifies to

$$W_{jk}(\vec{N}) = T_{jk} + \sum_{l=1}^{K} T_{jl} L_{jl}(\vec{N} - \vec{e}_k) \quad (39)$$

for exponential service time distributions.

The above results apply to single server FCFS centers, but cannot be easily extended to multiple server centers. In [51] an approximation is developed to handle non-product form FCFS multiple server centers. The approach is also based on the assumption that the arrival theorem applies to the network model under consideration. Furthermore, an additional assumption is used to relate: (a) the probability mass function that, at arrival times, the server is serving $\vec{n}$ customers (given that all servers are busy), and (b) the network flows. Experimentation has shown that the results are accurate. In [36] a linearizer algorithm based on this method was developed with good results.

Priorities.

Priority scheduling disciplines are frequently used in computer systems [102]. For instance, jobs with different resource requirements may be assigned different priority levels at the CPU. In packet switching networks, control packets usually have higher priority than short message packets, which in turn may have higher priority than long messages.

The shadow approximation [164] is an early iterative technique proposed for preempt-resume priority systems, but large errors have been reported. More recently, Bryant *et al* [22] proposed an MVA-based approximation for queueing networks (open, closed or mixed) containing preempt-resume and head-of-the-line priority service centers, which appears to have better accuracy than previously published results. Similar to Reiser's approximation for FCFS centers, it is assumed that the arrival theorem holds for non-product form networks with priority queues. In light of Eq. (14), the expected sojourn time of a tagged job in a preempt-resume priority queue is given

by three components: (1) the mean service time of the tagged job at the center: (2) the mean backlog of jobs in the queue with equal or higher priority than the tagged job; and (3) the delay due to higher priority jobs which arrive while the tagged job is queued. A new equation for the mean waiting time of a job with a given priority is obtained, and an MVA like recursion is used to solve for the mean performance measures. A similar development applies to head-of-the-line priority queues. [22] reports on extensive experiments to evaluate the accuracy of the approximation.

The approximation above requires a recursive solution over all network populations and thus has the same computational cost as MVA. But the work reported in [59] showed that a Linearizer type algorithm can be used to alleviate complexity without sacrificing much accuracy. Other related work appeared in [16] where the authors tried to overcome potential sources of errors in previous approximations (such as "synchronization error"). The modifications proposed seem to improve the accuracy for some cases studied.

Bard's Approximation.

An early example of an iterative method is found in [10] where Bard proposed the use of Eq. (40) below instead of the MVA equation (Eq. (14)):

$$W_{jk}(N) = F_{jk}(\lambda_1, \ldots, \lambda_j, L_1, \ldots, L_J) \tag{40}$$

to model several system characteristics including some which lead to non-product form networks. F_{jk} gives the proper approximate solution in terms of parameters of the model; the exact form of F_{jk} depends on the type of the j^{th} service center. Among the models considered were: tightly and weakly coupled multiprocessors, CPU priorities, software subsystems, distributed systems, etc.

4.4.2 Methods Based on Hierarchical Decomposition.

The approximations above are examples of MVA based techniques. In methods based on hierarchical decomposition, the network is decomposed into subnetworks which are solved independently. Then the results are aggregated to obtain the solution for the original network [40,41].

Several decomposition methods are based on a heuristic use of Norton's theorem for non-product form networks. These approaches are based on weakly coupled subnetworks [116].

FCFS centers with relaxed service distribution constraints.

Marie [125] developed an iterative algorithm for single chain networks with non-product form FCFS centers. It has been extended for multiple chains by Neuse and Chandy [135]. Recently, Akyldiz and Sieber [7] proposed an extension to handle QLD centers in single chain networks.

In Marie's method, each iteration involves analyzing a set of subnetworks and their complements. For the sake of exposition, consider a network that has two non-product

form FCFS centers (e.g., non-exponential service time distributions), say center i and j, as shown in Figure 10a. It is assumed that Norton's theorem is valid for non-product networks. A subnetwork is formed consisting of center j and the Norton's flow equivalent of the remainder of the network, as shown in Figure 10e. The flow equivalent server for the remainder of the network is obtained by substituting for center i an "equivalent" exponential server and applying Norton's theorem, as shown in steps (b) through (d) in Figure 10. (In that figure, $equiv_i$ is the "equivalent" server for center i.) This subnetwork is not product form (since center j is FCFS with general service time distribution) but it is an M/G/1 queueing model with state dependent arrival rates, and can be efficiently solved. After solving for the subnetwork containing center j, the parameters of the an "equivalent" exponential center j are determined. (In the equivalent center, the service rate for a given queue length is set equal to the mean departure rate at that queue length.) The same steps are followed substituting center j by i (Figure 10g), and iterating until a convergence criteria is satisfied. The accuracy of the method appears to be good, but it is clearly very expensive for multiple chain networks, unless the number of chains and the number of non-product form FCFS centers is small.

Simultaneous resource possession.

The approaches based on decomposition and Norton's theorem have also been successfully used to model a class of queueing networks with simultaneous resource possession [151]. A typical example is shown in Figure 8. Other examples include the representation of software critical sections. These are examples of models which include passive resources and are also called models with simultaneous possession of resources.

The basic approach used to solve central server models with memory constraints is to isolate the CPU-I/O subsystem, and to compute its Norton's equivalent server. The final solution is obtained by solving a network with the terminals and the equivalent server with state dependent service rate equal to $\mu(m)$, $0 \leq m \leq M$, where M is the number of memory partitions. Sauer [151] has studied the accuracy of this decomposition approach and has shown that it is effective for single chain networks. The method can also be extended to multiple chain networks which do not share the same passive resources. In [151] a solution was proposed to solve networks where jobs in different chains share the same passive resource, but it is computationally expensive.

Several iterative methods have been proposed for solving the simultaneous resource possession problem. They follow the classical iterative decomposition approximation technique as described in the introduction to this section. Each of the examples below employs a different method for choosing the subnetworks, its complement, and the functions which relate the parameters for the complement of a subnetwork to the solution of other subnetworks.

In the method of "surrogate delays" [92], a system is partitioned into a set of resources which are held alone (without any other resource being simultaneously held), a set of primary resources and a set of *secondary* resources. The primary resources are those which are held while a customer requests service in the secondary subsys-

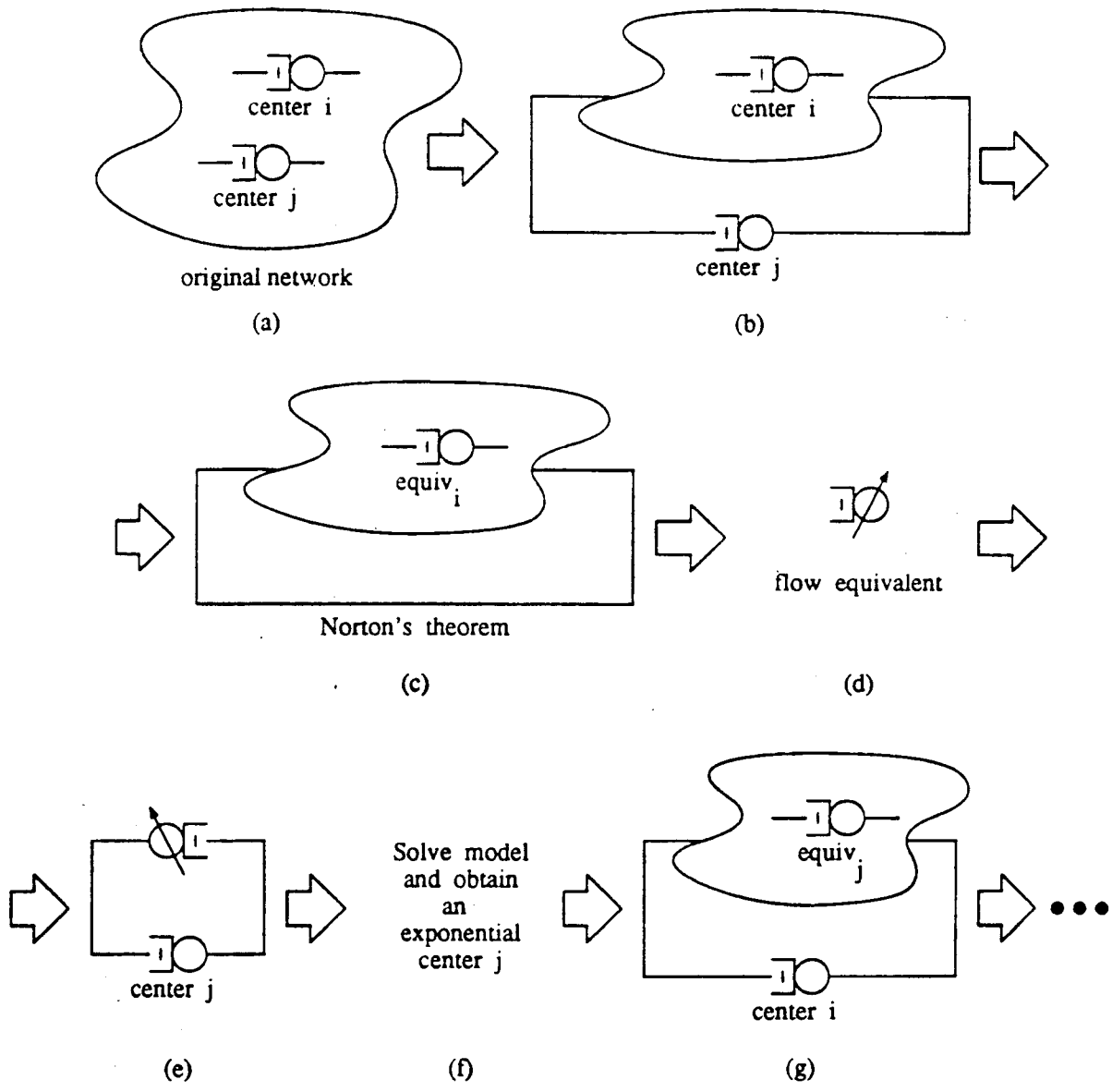

Figure 10: Marie's method.

tem. In the surrogate delay approximation, two submodels are constructed: the first represents explicitly the queueing delay for the primary resources, and the second the queueing delay due to congestion in the secondary subsystem. In the example of Figure 8, the primary resources are the memory partitions, and the secondary subsystem is formed by the I/O disks and the CPU. The first submodel contains an IS center for the terminals and an M server center (M is the number of memory partitions) representing the queue for primary resources, as shown in Figure 11a. The service

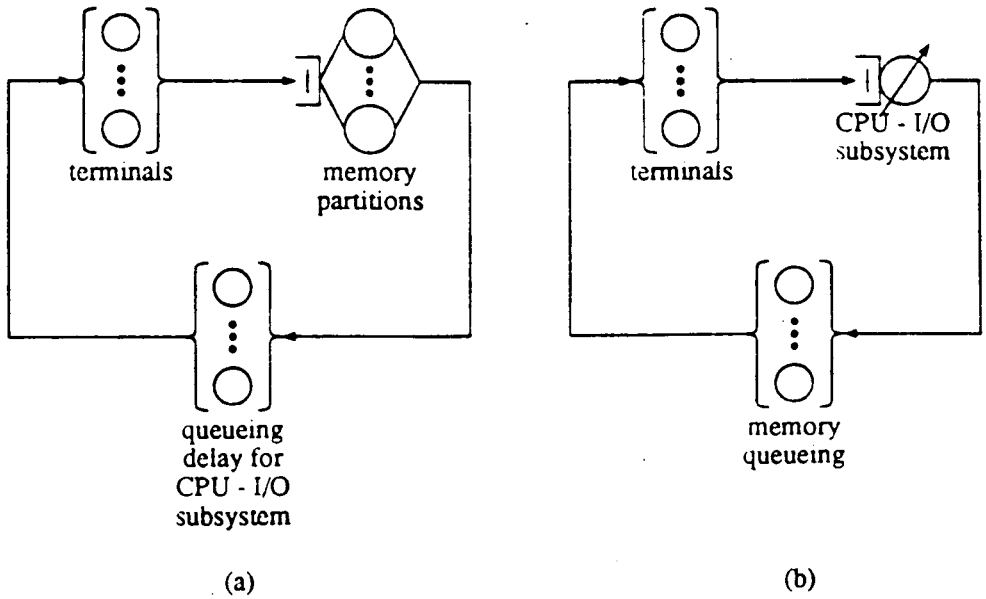

Figure 11: The method of surrogate delays.

time of this last center is equal to the response time in the subsystem, assuming that no contention occurs. The complement of this submodel is represented by an IS center which acts as a surrogate for the queueing delay (excluding service) in the CPU-I/O subsystem. The second submodel represents the terminals explicitly and includes a flow equivalent center for the CPU-I/O subsystem (obtained from Norton's theorem), as shown in Figure 11b. The complement is represented by an IS center, representing the queueing delay (excluding service) due to contention for memory. The solution is obtained by iterating between the two submodels. The method of surrogate delays is not readily extendible to multiple chain networks [92]. Other interesting application examples are also presented in [92].

In another method proposed by Jacobsen and Lazowska [94], simultaneous resource possession is represented as a population constraint in a subnetwork. The approach, applied to the network of Figure 8, also requires an iteration between two product form subnetworks. The first includes an IS for the terminals and a multiple server center with M servers representing the memory partition resources. The complement is represented by the average service time of this multiple server center, which is equal to the average time spent in the CPU-I/O subsystem. The second subnetwork explicitly includes the CPU-I/O subsystem. The complement is represented by the population size which is chosen equal to the mean number of memory partitions in use. This quantity is obtained from the solution of the first submodel. Since this number may not be an integer, an approximation is used to solve for the second subnetwork. Several example applications were presented in [94] which demonstrate that method

can be quite accurate. Although the method was also intended for solving multiple chain networks sharing the same passive resources, additional approximations would be required and no extensions were presented in [94].

In [120], another approximation was proposed to solve multiple chain queueing models in which each chain has an independent memory constraint. We refer again to Figure 8, but now there are K chains and K independent sets of memory partitions. The basic idea is to solve K single chain models, each one representing the terminals, with a flow equivalent server modeling the memory constraint for a given chain (say chain k). The flow equivalent server is obtained from the solution of a multiple chain network by varying the population of chain k and fixing the other chain populations to the average number of jobs of the corresponding chain in the I/O subsystem. Note that each iteration requires the solution of K single chain models with a QLD center and the solution of a multiple chain model for all populations of each chain with memory constraints. In [120] this approximation technique is also proposed as an approximation for shared memory constraints, but in this case, the results are not too accurate.

Krzesinski and Teunissen [109] extended the previous approach to solve models with multiple chains sharing the same passive resource. In this method, each chain visiting a shared passive resource is modeled as a chain visiting a dedicated passive resource subjected to a constraint which is calculated from the average number of customers of the other chains in the constrained subnetwork. As in [120], each iteration requires: (a) the solution of K single chain subnetworks with a QLD center, where K is the total number of chains sharing the same passive resource (MVA was used for the solution); and (b) the solution of a multiple chain subnetwork as in [120], where the population of a chain is allowed to vary and the other chains are maintained fixed at the average population. Since Linearizer was used to solve for the multiple chain model, it has to be invoked $O(\sum_{i=1}^{K} N_k)$ times, in order to obtain the results for all possible populations. The examples presented in [109] showed the accuracy of the approach, but the computational cost can be high.

De Souza e Silva and Muntz [51] also addressed the simultaneous possession problem. Like [109], the approach is applicable to networks with dedicated and/or shared passive resources. Furthermore, the use of passive resources can be nested. In this method, two or more subnetworks are constructed. The effect of the complement of a subnetwork for a chain is represented differently for different chains, depending on whether the chain visits a passive resource in the subnetwork and/or in the complement. Some of the subnetworks may not be product form due to multiple server FCFS centers (with different service times for different chains visiting the center) which result after applying the decomposition procedure, and an additional approximation was developed to handle this case. To illustrate the method, we again refer to Figure 8. Assuming that there are K chains sharing the same passive resource (memory partitions), two subnetworks are built. The first represents the terminals and the memory passive queue. The complement of the subnetwork for the K chains is represented by an IS center which collapses to one FCFS multiple server center due to the memory constraint. The mean service time of jobs from each chain at this center is given by an equation relating parameters of the other subnetwork. Figure 12a shows the first network. Note that this subnetwork is not product form due to

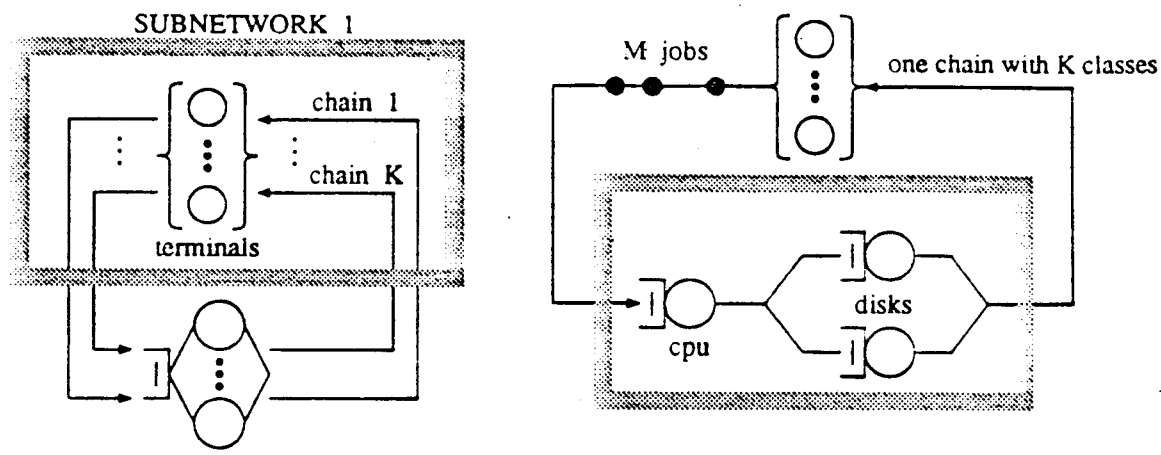

Figure 12: An approximation method for simulataneous resouce possession.

the multiple server FCFS center with different service times for different chains. The second subnetwork represents the CPU and the disks, as shown in Figure 12b. The complement of the subnetwork for the chains is represented by: (a) a single chain with M jobs (recall that M is the number of memory partitions) and K classes, each one corresponding to a chain of the original model; and (b) an IS center in which the mean delay is a function of performance measures of the first subnetwork. Note that each iteration involves only: (a) the solution of a K chain network (an MVA based approximation algorithm was used, but a Linearizer based approximation [36] may also be employed), and (b) the solution of a single chain network (MVA can be used with little cost).

In [51], an example was also presented for a model which represents contention for peripheral processors in addition to memory. The model contains nested passive resources. Several examples demonstrate the accuracy of the approach. Some of these examples also appear in [109]. For those, the overall accuracy seems to be comparable, but the maximum error in [51] was smaller than the maximum error in [109]. However, no extensive systematic comparisons have been carried out for the various approaches we have described.

Other approximations exist for the memory partition (passive resource) and related problems. Some of these can be found in [5,19,20,175].

4.4.3 Other Approximations.

van Dijk's Bounding Methodology.

Van Dijk [189] has proposed a methodology for obtaining bounds for non-product form networks. The basic idea is to modify the original queueing network problem so that it satisfies product form requirements and so that a performance measure obtained with the solution of this network bounds the corresponding measure of the original network. The modification proposed is such that the resulting network satisfies job-local-balance (i.e., "for any state and any job, the rate into that state due to a particular job is equal to the rate out of that state due to the same job"). In several systems, it is relatively easy to recognize the causes for violation of job-local-balance

at the system level (i.e., in terms of queues, service disciplines, etc. rather than in terms of the Markov chain description of the model) and to make the necessary modifications. Application of the methodology to some non-product form models such as overflow models, systems with breakdown, etc., are presented in [189] (and the references therein). It is argued that the bounds obtained are sufficiently closed to be useful to obtain insights. As with other approximation methods (e.g., MVA-based methods), some ingenuity is required to apply the approach to each new model.

Modeling Program Behavior: Fork/Join Constructs.

Heidelberger and Trivedi [81] proposed an iterative approach to solve models of parallel processing systems where a job subdivides into two or more tasks after visiting a service center (these centers are called "split nodes"). These models arise, for example, in some interactive systems where the user submits a batch job while continuing to execute interactive tasks. In their method, the split node is replaced by an "equivalent" product form service center. Interactive jobs are modeled by a closed chain as usual. The secondary (split) tasks are modeled by a Poisson source with arrival rate dependent on the throughput of the corresponding closed chain. Thus the approximate model is mixed. Iteration is used to match the closed chain throughput with the Poisson rate.

In [80], Heidelberger and Trivedi modeled the behavior of a class of programs which allow concurrent execution of some of the tasks within a job. A central server model with terminals was used to represent the system resources. In their model, a job consists of a primary task and two or more secondary tasks. A primary task executes using the system resources and can spawn two or more secondary tasks. The secondary tasks execute independently using the system resources, while the primary task waits. Once all spawned secondary tasks finish, the primary task resumes execution.

Consider that each task spawned from a job belongs to a different class c_i, $i = 1, \ldots, T$ (where T is the total number of tasks spawned by a primary task). c_0 is the class of the primary task. Consider a stochastic process in which each state represents a particular number of tasks of each class in execution. The basic assumption is that this process evolves as a Markov chain. It is also assumed that the time between transitions is sufficiently large in comparison to changes in the queue lengths at the service centers, so that the queue length distributions converge to the steady state distribution prior to the next change in the number of executing tasks. This is identical to the assumption of weakly coupled systems used in decomposition. The transition probabilities between states are calculated from: (a) the solution of a central server model with $T+1$ closed chains, each one corresponding to a task class c_i and with the number of customers at each chain given by the corresponding state of the Markov chain (note that if MVA is used only one pass is needed for all population vectors); and (b) additional approximations for computing the conditional probabilities that a class c_i task ($i = 1, \ldots, T$), which has just completed execution, finds that all its siblings have already completed executing, given the current number of active tasks of each class. The solution method was shown to be very accurate. However, only models with one type of job was considered, and the solution is very costly if jobs spawn several (say, more than three) tasks. Another, less costly but less accurate

approximation was also proposed in [80].

5 Decision Support.

Queueing networks have been employed as a basic tool for formulating and solving a number of optimization problems. These optimization problems can occur during any of the system life cycle phases. For instance, in system sizing the modeler may want to select the capacity of system resources which have to be allocated, such that some system measure is optimized subject to certain constraints, such as maximum cost and/or minimum throughput. In other problems, the goal is to distribute the resources among the customers (jobs, packets, etc) so that some performance measure is optimized subject to constraints. It can be useful to compare a performance measure obtained from data measurements taken from the system during operation with the "optimal" performance measure obtained from a model. If the measured performance is much worse than the estimated optimal, the analyst may want to "tune" the system by altering some parameter values in order to improve performance. In this case, sensitivity analysis plays an important role to determine the influence of different parameters on the measure to be improved.

5.1 Optimization Problems.

The set of decision problems can be organized according to:

- which model parameters can be controlled;
- the measure to be optimized;
- the set of constraints.

For computer/communication systems, the most common model parameters to be controlled are:

- routing;
- service capacities;
- rate of service demands;
- scheduling disciplines.

Usually, only one of the model parameters is optimized with the others fixed a priori, but in general a subset of the parameters can be optimized simultaneously.

The general routing problem deals with the assignment of customers' demand for service to resources that can provide the service such that some performance measure is optimized subject to specified constraints. For instance, in the "classical" routing problem in long-haul computer networks, the resources are the channels and the customers are the packets traveling through the network. The problem is how to choose the set of channels (the route) to be allocated to the packets traveling from

a source to a given destination. Load balancing problems can also be put in this general framework. In these problems, the resources are the disks, file servers, or computer sites in a local network environment. The users are the jobs to be executed and the problem is to "route" the jobs to the resources in order to balance resource utilizations. File allocation problems are another variation of the general routing problem. The issue is how to distribute the files among disks to optimize some performance measure. In several studies, this problem is reduced to determining optimal routing probabilities.

In the capacity assignment problem, the capacities of the resources are chosen in order to optimize a performance measure. For instance, in long-haul computer communication networks, the issue is to find the optimal capacities of the links to maximize throughput (or minimize the network average delay) subject to cost or fairness constraints. For a single computer system, the issue may be to choose the capacity of the CPU and/or the disks subject to cost constraints. It is interesting to observe that it may be possible to relate the routing problem with the capacity assignment problem if a product form queueing network is used for the system model. In this case, the model solution is affected only by the relative utilization of the resources. The capacity c_i of a device i can be computed from the relative utilization a_i, the visit ratio θ_i and the relative service time $\tau_i (= T_i/c_i)$, as $c_i = \theta_i \tau_i / a_i$. Therefore, by calculating the relative utilizations which optimize a performance measure of interest, the capacities can be obtained if θ_i and τ_i are fixed. Similarly, the routing in terms of the θ_i can be obtained if c_i and τ_i are fixed.

Examples of optimization problems for which the rates of service demands are the decision variables are: (a) selecting the throughput rates of jobs in a computer system in order to optimize the overall mean throughput subject to a cost constraint; (b) selecting the window sizes in a computer communication network with window flow control.

The influence of different scheduling disciplines on the performance measures of a queueing network has not been addressed much in the literature. It is, nevertheless, an interesting and difficult problem if the queueing network is not product form. For instance, it is not difficult to construct examples in which jobs from different chains have different bottlenecks which vary from a center to another according to the scheduling discipline of the centers.

The most common performance measures to be optimized are: response time, queue size, throughput and cost. Among the constraints we mention: total server capacity, total cost, minimum throughput (perhaps per chain), limited use of resources (not all resources can be used by all customers).

Below we survey examples from computer/communication systems which employ queueing network models. The examples are organized according to the particular application area. Other examples are presented in Section 6.

Routing in computer networks.

One of the earliest applications of queueing network models in computer communication network optimization problems was the routing assignment problem [14,102]. An open queueing network was used in these studies to model the contention for link capacities. The input traffic is assumed to be Poisson and λ_{ij} is the rate of traffic

entering at site i and destined for site j. The problem is to split the traffic among the possible paths from origin to destination such that the network delay is minimized. The routing obtained with this procedure is often referred to as probabilistic static routing, since the network flows are divided probabilistically among several paths. The optimal routing solution was formulated as a non-linear, multicommodity flow problem. Since the objective function is convex with respect to the flows, a downhill search method called Flow Deviation [64] was employed.

Open queueing network models do not accurately model networks with window flow control mechanisms, as we will describe in the next section in more detail. For networks which use window flow control in a virtual channel, a closed queueing network is more appropriate to model the level 3 protocol of packet switching networks [147]. Kobayashi and Gerla [103] proposed an algorithm based on the Flow Deviation method for finding the optimal routing of traffic in closed queueing networks. The direction of steepest descent is found by taking the derivative of the delay function with respect to the network flows. The derivative is shown to be a function of expected queue lengths, which can be easily obtained from the MVA algorithm. For single chain networks, the global minimum is obtained, since the objective function can be shown to be convex with respect to the set of routing patterns. For multiple chain networks, it is easy to show by counter example that the objective function is not convex. Therefore, in this case, the algorithm only finds a local minimum. Note that in [103] the routing is probabilistic, and several paths can be used between a source destination pair. However, in networks which employ virtual circuits, only one route is chosen when a session is set up between a pair of communicating sites. In [67], Gavish and Hantler describe a routing selection method for non-bifurcated routing. The approach is iterative and uses a modified version of the Flow Deviation method to make iterative improvements on an initial route assignment. At each step of the iteration, the error between the current solution and the optimal solution is bounded using a Lagrangian relaxation technique.

An example of optimizing routing and capacity.

Due to its special structure and wide applicability in computer system modeling, the central server model (Figure 1) has been used as the basic model for different optimization problems in the literature. We briefly survey some of these studies in this section.

Trivedi *et al* [186] studied the problem of determining the CPU speed, the capacity of the disks (in number of blocks of words) and the file assignment which maximize system throughput subject to a cost constraint. (The CPU cost is assumed to be a power function of its speed and a disk cost is assumed to be a linear function of its capacity.) A single closed chain central server model was used to represent the computer system. In their model, all files are broken into fixed size blocks, (which are stored in blocks of identical size in the devices) and each block can be independently stored in any secondary storage device. It is also assumed that the number of file blocks which are stored on a device and the number of blocks a device can store (the capacity of the device) are continuous variables. Furthermore, uniform probability of access to any block in a file is assumed. With these assumptions, the unconditional

probability of a request to a particular disk, after a CPU burst, is a simple function of the parameters: (a) the probability of referencing file i after a CPU burst and: (b) the number of file blocks stored on disk j. It was shown that the reciprocal of the throughput is convex, and that the set of feasible solutions is also convex. Together these guarantee that any local minimumn is a global minimum. The paper also includes a proof of an interesting property of the optimal solution, namely that if files are ordered in nonincreasing order of activity, then files should be assigned to disks in this order, filling the disks with highest cost per block first. This property was used to reduce the number of decision variables of the optimization problem. Finally, the solution to this nonlinear constrained optimization problem was obtained by: (a) first converting it to a nonlinear unconstrained optimization problem and then (b) solving it via a standard quasi-Newton minimization technique. It is also worth mentioning that this problem reduces to the file assignment problem if the CPU and disk capacities are fixed a priori.

Capacity optimization.

In [184] Trivedi and Kinicki were concerned with the selection of CPU speed and secondary memory device speeds such that the throughput is maximized subject to a cost constraint. The system is modeled by a central server model with a single closed chain. Device speeds are assumed to be continuous variables. (In a previous paper, Trivedi and Wagner [185] had shown that for this optimization problem any local minimum is a global minimum.) The Lagrange multiplier method was used to solve this constrained nonlinear optimization problem. For this method, the partial derivatives of the CPU utilization with respect to the relative utilization of the devices have to be found. From Section 5.2 we can see that this has a simple form in terms of mean queue lengths.

It is interesting to compare the routing problem of Kobayashi and Gerla [103] with the above problem. For single chain networks, if cost constraints are relaxed, these are equivalent problems and the main difference is the solution method employed.

Optimization of rates of service demands.

A simple example of a problem in optimizing service demand rates is that of choosing the optimum degree of multiprogramming together with device speeds such that the throughput in a multiprogrammed computer system is maximized subject to cost constraints. This can be easily accomplished by using the approach in [184] with a fixed degree of multiprogramming, and then performing a search for the highest throughput, as suggested in [184]. (In other words, several models must be solved, each with a fixed degree of multiprogramming.)

Heiss and Totzauer [82] used a central server model with open chains to find the open chain throughputs which maximize the utilization of the devices such that the response time of different types of users (modeled by different open chains) do not exceed given limits. More specifically, the objective function is a linear combination of device utilizations. The decision variables are the open chain throughputs. The constraints are upper and lower bounds on the open chain throughputs and an upper

bound on the response time of each open chain.

Load balancing.

A load balancing policy chooses the resources that should be used to run a job (from the available resources that are capable of executing the particular job) in order to optimize a given performance measure. As mentioned in the introduction to this section, this is similar to determining an optimum routing policy for communication networks, but there are some significant differences. In the routing problem, a set of source-destination pairs, the traffic for each pair and cost/performance constraints are specified. The resources (communication channels) are all identical in function. In the load balancing problem, there is no notion of source-destination traffic. Instead, there are collections of one or more resources which can perform a certain type of work and which we might call *functionally equivalent subsystems*. During execution, a job can choose (or be assigned) to access resources in a particular subsystem to obtain a certain type of service. Usually, the routing of jobs to the subsystem is not an issue. In some systems, jobs are grouped into classes and, for each class, resources are classified as either *local* or *foreign*. If the dispatcher chooses to execute a job at a foreign resource, a penalty is paid (e.g., extra processing is needed) to dispatch the job. An important property of a load balancing strategy is fairness of service, i.e., the system should operate in such a way that all jobs, regardless of their class, should be provided with specified *acceptable* levels of performance. As an example of a "fairness problem" that may occur, consider a central server model representing a computer site in a local area network. This site has a CPU (with PS scheduling) and a disk (with FCFS scheduling). Assume that there are two types of jobs: type a with very long CPU mean service times, and type b with relatively small CPU mean service times. Assume also that the CPU is the bottleneck. In an attempt to improve the response time of both jobs, type a jobs are sent to execute in a "foreign" site which has more CPU capacity available. Both types of jobs, however, continue to use the local disk (i.e., files are not transferred to the foreign site). It is easy to construct a case in which the bottleneck is transferred from the CPU to the local disk, and the response time of type b jobs can increase significantly. This is essentially the same problem reported in [79] in a different context.

A load balancing policy may be source-initiated or server-initiated, depending on whether the source site makes the decision where to execute the job or the server sites search for work when they become idle [196]. Furthermore, load balancing can also be static or dynamic. Static load balancing algorithms generally base their decision on the average behavior of the system. Therefore, static load balancing aims at obtaining optimal mean performance at steady state with no dynamic control based on the current system state. A drawback of static algorithms is that they do not adapt to short-term changes in the load. Dynamic strategies, on the other hand, base their decision on current information about the state of the network. A taxonomy of load balancing algorithms and a comparison of several policies can be found in [196].

The usefulness of static algorithms might be questioned, since operational systems often implement dynamic control procedures. However, both types of algorithms are important since they play different roles. Static models are useful for system sizing

(e.g., allocation of resources, identification of bottlenecks, sensitivity studies, etc.). Note that some decisions are inherently static, e.g., file allocation in most applications. Static algorithms can sometimes be exploited for dynamic load balancing (e.g., some routing algorithms in computer networks [14]). Dynamic models are harder to solve analytically, and solutions often deal only with very small systems and/or are based on simplifying assumptions such as system homogeneity.

Many papers that deal with load balancing algorithms model the distributed system being analyzed as a system of M parallel queues as illustrated in Figure 13. The

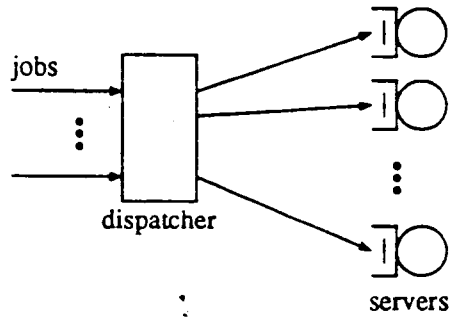

Figure 13: A queueing model for load balancing.

queues represent the resources in the network subject to load balancing (e.g., processors, computer servers, etc.). There is a dispatcher which is in charge of distributing the load among these queues (e.g., [33,136,203]). The overhead incurred in transferring a job from its "local" site (i.e., origin) to a "foreign" site is sometimes included in the model (e.g., [58,66,123]). In [46,47,172], a more detailed representation of the network is considered. Below, we briefly review some of the results in the literature.

Chow and Kohler [33] studied three policies based on immediate information about the state of the system. In the first policy, the dispatcher transfers a job to queue k if $R_k(n_k+1) = \min_l\{R_l(n_l+1)\}$ where $R_l(n_l) = n_l/\mu_l$, n_l is the current number of jobs in queue l and μ_l is the service rate of queue l. This policy is aimed at minimizing the expected response time of an arriving job. In the second policy, queue k is chosen to receive an arriving job if this choice minimizes the expected time to complete service of all the jobs in system assuming no further arrivals. (It is assumed that all servers are exponential which facilitates this calculation.) The third policy assigns an arriving job to queue k if this choice maximizes the expected throughput of the system during the next arrival period. A solution is obtained using numerical techniques and only for a system with two queues.

Yum and Lim [203] studied systems in which jobs are restricted to run in a subset of the resources, according to their types. Different policies are compared (including one of the policies of [33]). Numerical solution methods were employed and only systems with three queues were studied.

Ni and Hwang [136] obtained the optimal assignment probabilities which minimize the expected system delay for a static load balancing policy. The system is modeled by a set of parallel M/M/1 queues (processors) where each job type i is constrained to run only in a subset $\mathcal{P}_i$ of the processors. The problem is formulated as a non-linear

programming problem with constraints. In order to obtain the final solution, the authors first solve the simpler problem of finding the optimal routing probabilities for one class of jobs in a system with two classes only, in which the second class has its rate already preassigned. The solution to the original problem is obtained (roughly) as follows. All classes which are restricted to run in a single processor are combined into one class with preassigned rates. All the remaining classes (say classes $u, \ldots, v$) are combined into a single class where the jobs can run in any processor belonging to the set $\mathcal{P}_u \bigcup, \ldots, \bigcup \mathcal{P}_v$. This "simpler" optimization problem is solved and the optimal throughput calculated for each processor (say δ_j for processor j) is used to obtain a set of rates r_{ij} for each original class $i = u, \ldots, v$. These assigned rates r_{ij} are such that they satisfy as close as possible the optimal rate δ_j for all j. The rates r_{ij} are made into preassigned rates. The remaining job rates are combined into a single class and the "simpler" optimization problem is solved again. This process is repeated, recursively, until the final solution is obtained.

Eager *et al* [58] included in their model of dynamic policies the processor overhead spent in transferring a job from its local site to a foreign host. As in previous work, hosts are modeled by single queues, but here there is a dispatcher in each host which makes the decision of whether to transfer a job or not. The decision to transfer a job to another host is made based only on local information. More specifically, a job should be transferred if the local queue length exceeds a given threshold ("threshold policy"). However, the choice of the foreign site to receive the transferred job may or may not be state dependent. Three policies are studied: (a) a random policy in which a site is chosen at random. Since new arrivals and transferred jobs are treated equally, a transferred job may be retransferred if it arrives at a host with queue length exceeding the threshold value. To avoid instability, a limit is imposed on the number of transfers a job may experience; (b) a threshold policy where a site is selected at random and then its queue length is probed to determine if it is below the threshold. If so, the job is transferred. If not, another site is probed up to some specified maximum number of probes. A job is processed locally if the probing limit is exceeded; (c) a shortest queue policy, where a given number of sites are chosen at random and their queue lengths are obtained. The site with the shortest queue length is chosen to receive the job, if its queue is less than a given threshold. Otherwise, the job is processed locally.

A model is developed for a homogeneous system. The key assumption made to obtain an approximate solution is that each site behaves independently of other sites in the network. Thus each site can be analyzed in isolation. The effect of the remainder of the network on a site is modeled by a Poisson arrival process of transferred jobs. Since all sites are identical, information about the current state is obtained from the solution of a single site.

Lee and Towsley [123] also considered dynamic threshold policies. The model and main assumptions are similar to those in [58]. The choice of a foreign site is always made at random and no retransfers are possible. The optimum threshold is obtained numerically by exhaustive search. As in [58], only homogeneous systems are considered. Unlike [58], there is a communication delay whenever a job is transferred to a foreign site, but no extra processing at the local site. Furthermore, the effect of assigning different priorities for local and transferred jobs is studied.

The models of static and dynamic policies presented above are simplistic in that the hosts in the distributed system are represented by a single (fixed rate) queue and/or only homogeneous systems are considered. The models discussed next use a more complex representation for a site. For example, a host can be represented by a central server subnetwork or by any other product form queueing network model. Complex job behavior can also be represented, and the distributed system may be heterogeneous. However, only static load balancing is considered.

In the model of Tantawi and Towsley [172], the distributed system consists of a set of sites connected by a communication network. Sites are represented by a number of resources, and different sites may have different configurations and resources with different processing rates. Jobs arrive at each site according to a Poisson process with possibly different rates for each site. The communication network may be represented by one or more resources. The model is required to be a product form queueing network.

The optimal static load balancing problem is formulated as a non-linear optimization problem. If the communication delay is such that it satisfies the so called "triangle inequality property" (i.e., the communication delay of transferring a job from site i to j is never greater than the delay of transferring the job to a site k and then to site j), the authors were able to show that sites are partitioned into three categories: they are either sending jobs for foreign processing but not receiving jobs from foreign sites; or only receiving jobs from other sites but not sending any; or only processing local jobs. It was also assumed that the network communication delay is a function of the total traffic in the network and is identical for all source-destination pairs. Under this assumption, a set of relations between partial derivatives of the response time of jobs processed in a site with respect to the job completion rate of that site and, partial derivatives of communication network delay with respect to the total traffic through the communication network is found using the Lagrange multiplier method. These relations must be satisfied by the optimal solution. The partitioning of sites and the relations are used in an efficient algorithm for determining the optimal solution. Furthermore, an algorithm (appropriate for performing parametric studies) was developed to find the average values of the communication delay at which the partitioning of sites will change, i.e., sites will move from one category to another.

This "direct" solution method in [172], albeit elegant, requires the assumption of a single chain network and does not handle cases where jobs from one site are restricted to run in a subset of sites in the network. For these cases, however, a downhill search method, such as the Flow Deviation method [64] developed for product form open queueing networks can be employed.

De Souza e Silva and Gerla [46] also modeled a distributed system as a collection of central server models (with terminals), one for each site. In this model, multiple classes of jobs with different demands are allowed. More specifically, it is assumed that there are local application programs running at each site. These programs are modeled as closed chains and are not subject to load balancing. Other types of jobs are assumed to come from a large number of terminals (such as in some transaction processing systems) and are modeled as open chains. These interactive jobs may run locally or may be transferred to a foreign site via a communication network. It is further required that the network of queues representing the distributed system has

a product form solution. There can be multiple open chains with different resource loadings to represent different types of interactive work. Some of these classes may be restricted to run in a subset of sites in the network. The objective function to be minimized is the overall mean system delay where weighting factors are parameters to be chosen by the designer to avoid unfairness to different classes.

The load balancing problem is formulated as a nonlinear optimization problem. The solution method chosen is a downhill technique based on the Flow Deviation method [64]. In order to compute the steepest descent direction for the downhill technique, partial derivatives of the objective function with respect to the decision variables (open chain flows at each service center) must be computed. It is shown (see also [53]) that the derivatives can be obtained from means and variances of queue lengths which can be easily computed using MVA (for calculating the means) and new equations using an MVA-like recursion (for calculating the variances).

In a subsequent paper, de Souza e Silva and Gerla [47] extended the above technique to allow: (a) multitasking within each job and (b) jobs with spawned tasks. In the first extension, a job may require the execution of several tasks in sequence, before completion. Each task is subject to site constraints, i.e., it is allowed to run only on a subset of sites in the system. A motivation for this model arises in distributed database applications in which the database is partitioned (and replicated) among several sites. A job can be initiated on any site, and consists of a sequence of tasks that are processed on the site where the referenced data is located. The task sequence is modeled by introducing class changes [12] in the chain which represents a particular type of job. The algorithm takes into account all the possible assignments of tasks to sites.

The second extension is motivated by distributed system applications where interactive users may submit batch jobs to be executed. These batch jobs can be assigned to a set of sites in the system (e.g., server sites) and run until completion in the selected site. (For instance, the batch jobs could represent the compilation of a program, text processing, etc.) Interactive jobs (such as text editing) are modeled as closed chains. Spawned jobs are approximately modeled as open chains with rates dependent on the related closed chain jobs, following the approach of [81].

File allocation.

The file allocation problem can be formulated as a variation to the load balancing problem. (For surveys on this problem see, for example, [55,194].) The placement of files on the disks can be interpreted as determining the routing probabilities to the disks after a CPU access. Many solution techniques are based on the following two step procedure: a) find a solution for the optimum branching probabilities to the disks and then b) from the file access statistics determine a file allocation that best approximates the optimum branching probabilities. An example of this approach can be found in [186]. Another example is found in [202] which is discussed below.

In [202], Woodside and Tripathi considered a model of a local area network with J identical SSFR centers representing file servers and an IS center representing workstations (see Figure 14). There are K single customer chains, each one representing the behavior of a workstation user. A user cycles between accessing its one workstation

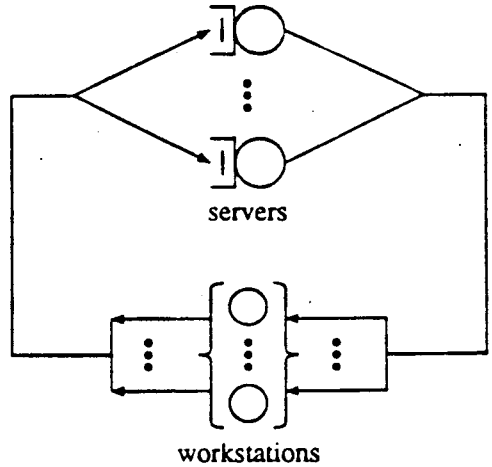

Figure 14: A queueing network model of a local area network.

and accessing a file server. Files of each workstation may be allocated in one or more file servers. As in previous work, the allocation of files in different servers can be characterized by the routing probabilities from the center representing the workstations to the centers representing the file servers. The problem studied was the optimal allocation of files to the file servers in order to maximize the total utilization of the file servers. In the model this reduces to optimally choosing the routing probabilities. It is shown that each workstation should access a single file server and that the load of the file servers should be balanced.

In a subsequent paper [182], a more general model was considered by the authors. Servers to be allocated may be load dependent and need not be identical. Furthermore, the servers are divided into subsets such that a server belongs to a subset k if it can execute work of type k. Some of the chains may have their demands (characterized by the routing probabilities) already fixed, while the routing probabilities of the other chains can be varied. Let p_{rki} be the probability that a request for a service of type k from the chain r customer, is served at center i. The key result indicates that, to maximize a weighted sum of the chain throughputs, the routing probabilities p_{rki} are such that for each chain r and service type k, p_{rki} is 1 for some value of i (and 0 for all others). In other words, there is an optimal solution in which each customer always goes to the same server for a particular type of request. The optimal solution is thus at a vertex of the space of feasible values for the transition probabilities. The result, albeit very interesting, does not solve the optimization problem since explicit values for p_{rki} are not obtained. However, it restricts the form of the optimal solution and may be useful for pruning the search space.

In [181], an extension of the model in [202] was studied. In this new model, workstations may share files, and the allocation of shared files to servers is assumed to be given. The problem is to find the optimal allocation of the non-shared files. As in [182,202], the optimal solution is also shown to be at a vertex of the space of feasible solutions, but the optimization problem, unlike [202], has no known efficient solution. Therefore, the authors proposed several heuristics to obtain a suboptimal solution for the placement of files at the servers.

It is worthwhile to note the difference between the solution obtained with the above model and the model considered in [186]. In the former, each user behaves independently of the others. (There is a distinct routing chain for each user.) In the latter model, customers are statistically identical, and only one routing chain is used to model customer behavior. The optimal solution to the first model is at a vertex of the solution space. The solution to the second model is not, in general, at a vertex of the space of feasible solutions.

Finally, it is interesting to observe that the results in [182] can be used to find a suboptimal solution to the allocation problem as follows. Assume that each workstation is activated one at a time. Since the model is characterized by many single customer chains and a few service centers, the DAC algorithm is particularly suited to this problem. Furthermore, from the results of a network with r users, it is very easy to obtain the results for the network with one more customer, and it is efficient to compute the results when all possible values of $p_{rki} \in \{0, 1\}$ are tried. This process is similar to the one used in [68] for finding the window sizes in a computer network, where a solution close to the optimal is obtained. Clearly, the order in which chains are added to the network will affect the final solution, but since DAC permits backtracking, it is easy to evaluate different orderings.

In Wolf [200] we find an example of an approach applied successfully to IBM mainframe systems. The objective function is the mean I/O response time. Constraints can be imposed on the relative disk access rates. An open queueing network model was used to obtain the relative disk access rates, and heuristics were developed for the final solution. The approach was used to assign files to disks in two IBM systems: an IBM 3081 CPU with 15 channels, 15 controllers and 144 disks, and an IBM 3090 with 40 channels, 42 controllers and 168 disks. Significant performance gains were reported when files were placed according to the new approach in comparison with the previously utilized allocation schemes.

Summary.

In this subsection we surveyed various optimization problems applied to computer/communication systems. We have argued that many problems can be cast as a form of the "routing problem", although there are also some key differences.

Queueing optimization problems have been successfully applied in the analysis and design of data communication networks. The file allocation and load balancing problems have become increasingly important as the number of distributed system applications proliferate. We have discussed a number of studies concerned with analyzing and comparing load balancing strategies. Unfortunately, the more interesting case of dynamic load balancing is more difficult to analyze, and only simple models have been studied. Static models, on the other hand allow important characteristics to be modeled, such as complex site models, site constraints, jobs with multiple tasks, etc.

In the solution of an optimization problem, it is often useful to know certain properties of the measure to be optimized. For example, it may be useful to know if the throughput as a function of a certain model parameter is convex or not. Some of the papers referenced above have addressed this problem. Additional work can be

found in [97,167]. A particularly important class of properties is the sensitivity of the performance measures of a system to its parameters. This is the subject of the next section.

5.2 Sensitivity Analysis and Workload Characterization.

Sensitivity analysis plays an important role in system design. It allows the modeler to evaluate the impact of small changes in system parameters for a specific measure. As an example, this information can be useful in determining: (a) the most critical parameters to be changed in order to improve performance; (b) the influence of errors in estimating parameter values. In this sense, sensitivity analysis is related to workload characterization for queueing networks.

The sensitivity of a measure with respect to a parameter is defined in terms of the derivative of the measure with respect to that parameter. Let $\mathcal{M}$ be a specific measure and $\mathcal{M}_{\%}$ the percent change in $\mathcal{M}$ when the value p of a parameter $\mathcal{P}$ varies by $\mathcal{P}_{\%}$ percent. A first order expansion on $\mathcal{M}$ around the point $\mathcal{P} = p$ yields:

$$\mathcal{M}_{\%} = \left[\frac{p}{M}\frac{\partial \mathcal{M}}{\partial \mathcal{P}}\right]\mathcal{P}_{\%}$$

where M is the value of $\mathcal{M}$ for $\mathcal{P} = p$. The term in brackets in the expression above is the sensitivity of $\mathcal{M}$ with respect to $\mathcal{P}$ at the point $\mathcal{P} = p$. Clearly, a higher absolute value of the sensitivity with respect to a parameter indicates that a fractional change in the parameter value has more influence on the measure.

For product form queueing network models, expressions for performance measures (e.g., throughput, expected queue length) typically involve the normalization constant (see Section 3). Therefore, the evaluation of the partial derivative of a measure with respect to a model parameter will usually require the calculation of the partial derivative of the normalization constant with respect to that parameter. The partial derivative of $G(\vec{N})$ with respect to the visit ratio θ_{jk} is given by [53,126]:

$$\frac{\partial G(\vec{N})}{\partial \theta_{jk}} = \frac{G(\vec{N})}{\theta_{jk}} L_{jk}(\vec{N}) \tag{41}$$

Williams and Bhandiwad [199] were probably the first to obtain an expression similar to the above for single chain queueing network models. They found an expression for the sensitivity of throughput by differentiating both sides of the equation $\lambda(N) = G(N-1)/G(N)$, and by then applying Eq. (41).

In [53,169], simple expressions relating first moments of queue lengths and partial derivatives of queue lengths with respect to input parameters (such as visit ratios and service times) are obtained. It is also shown that first moments can be easily obtained using recursive expressions similar to MVA. Thus, sensitivity measures can be easily evaluated. (The approach generalizes to higher moments of queue lengths and higher order partial derivatives.) It should be emphasized that partial derivatives of queueing measures with respect to input parameters are important not only in sensitivity analysis but also in optimization problems as well. For example, several studies discussed earlier used expressions similar to Eq. (41) in solving optimization

problems (e.g., [103.184]). In [46], expressions relating partial derivatives of queue lengths and their first moments were used in a load balancing optimization problem. The approach in [53] can be generalized to obtain recursive expressions for calculating partial derivatives of joint queue length probabilities which are necessary to evaluate the sensitivity of availability measures, as will be discussed in the Section 6.

Other studies of sensitivity include the work of Gordon and Dowdy [71] which generalizes the results of [199] to multiple chain networks and also considers the magnitude of errors introduced by "large" errors in parameter values, i.e., that are too large to use the first order approximation implicit in using the first derivative. Tay and Suri [173] using a different approach (operational analysis) consider the error in performance measures introduced by (a) errors in estimates of the parameters and (b) errors due to random fluctuations in the values of those parameters observed in a particular observation period.

Suri [171] studied the sensitivity of queueing network measures with respect to deviations from the "homogeneous service time assumption". This assumption states that the service rate at a station is independent of the number of jobs at that station. For single chain queueing networks, he obtained an expression relating the partial derivative of the throughput with respect to $\mu_M(n)$ (for station M) and marginal queue length probabilities:

$$\frac{\partial \lambda}{\partial \mu_M(n)} = \frac{1}{a_M} P[n_M \geq n+1] - \lambda P[n_M \geq n] \qquad (42)$$

A similar expression was obtained for the partial derivative of queue lengths with respect to $\mu_M(n)$. Eq. (42) was used to obtain bounds for the throughput when the variation in $\mu_M(n)$ $\forall n \leq N$ is bounded. (Bounds were obtained for average queue lengths for a balanced system, i.e., a system in which all centers have the same relative utilization.) The bounds obtained have shown the relative insensitivity of the throughput with respect to the homogeneous service time assumption.

The above discussion has concentrated on the sensitivity of performance measures to estimates of customers' mean service demands at resources. Other parameters that have to be specified for a queueing network model include the number of customer chains and the population of each. There has been much less formal analysis of these issues. However recent work by Dowdy, Krantz and Tripathi [56] shows that the representation of a multi-chain queueing network by a single chain network pessimistically bounds the throughput for the multi-chain network. The resource loadings for the single chain are the mean loadings in the multi-chain model. Other results pertaining to these issues can be found in [204]. General discussions of the issues related to the workload characterization can be found in [63], and particularly with respect to the design and parameterization of queueing network models in [180].

6 Application Areas

6.1 Brief Summary

Queueing networks are particularly useful models in situations where customers sequentially demand service from a set of resources and the major effect on performance

is captured in the contention for these resources. While there are limitations to the kinds of models that are tractable (even with approximations), there are a number of reasons for the popularity of queueing networks, most of which we have touched upon in the previous sections. These include:

- ability to model systems with sufficient realism.
- availability of efficient solution methods.
- insensitivity of the results to small variations in the parameters.
- empirically verified accuracy of the results.
- approximation techniques to solve larger and/or more realistic models based on key exact equations.

There are of course alternatives to analytic models: namely simulation and benchmarking. Often the alternatives can be more accurate but have their own limitations; benchmarking requires the existence and ability to control the target system and simulation is often to costly. In some cases, e.g., during design, parameter values may not be known with sufficient accuracy to warrant the cost of simulation.

To summarize the major uses of analytic models are:

- design phase; to choose between several design alternatives.
- system sizing; comparing systems or selecting the configuration.
- system operation; for selecting tuning parameters.
- planning for the future; selecting new hardware/software configuration, usually due to evolving workload.

In the following sections, we discuss the application of queueing network models to a number of particular computer performance problems. Our treatment is necessarily brief in all cases but some preference is given to those areas that are perhaps less familiar or are of more recent vintage (e.g., interconnection networks and synchronization in parallel systems). The more traditional applications (that seems appropriate) such as capacity planning and packet switched networks are more briefly covered.

For a more general discussion of computer system performance evaluation see [78]. [117] is a survey that covers analytic methods and [122] concentrates on the application of queueing networks. [114,146,201] are concerned with communication network modeling. [114,201] are specifically limited to queueing network models while [146] is more general.

6.2 Capacity Planning.

Capacity planning is concerned with the determination of future system needs based on the predicted workload grow. For determining future system needs, the modeler generally must: (a) construct a model of the current system and (b) validate it by

comparing measures obtained in the real system with results obtained from the model. During the model construction, the modeler estimates the values of the parameters to be used in the model. This estimation may require measures taken carefully from the system. Once the analyst is confident of his model, the effect of changes in workload can be studied, and the effect of changing the hardware configuration can be evaluated. In [121] a detailed discussion of all steps that should be taken for predicting the behavior of evolving systems is presented.

Capacity planning refers to the evaluation and selection of computer system configuration changes, e.g., processor upgrade, number and/or organization of I/O devices. Queueing networks have proven highly successful as a modeling tool for capacity planning for computer mainframe installations. As evidence, there are a number of commercial tools and companies devoted to this activity [1,2,3].

Capacity planning involves three basic activities: measurement of the current system, model construction and validation and performance projections. Details of these activities with particular emphasis on the use of queueing networks can be found in [116,121]. As might be expected, product form queueing networks are often not general enough to represent many features of real computer systems including for example, memory contention, priority scheduling and simultaneous resource possession. In some applications these are first order effects and must be represented in the model. We have discussed approximate solutions for queueing networks with these features in Section 4. More detail on the application of these solutions for computer system capacity planning can be found in [116,121].

Obtaining parameter values for a queuing network model of a computer system can be challenging if the right measurement facilities are not available. For example, it can be difficult to attribute operating system processing time or paging activity to a particular job. In addition, there is the issue of deciding on the number of chains required to represent the workload. Assuming that detailed measurements are available on individual jobs, one would have the problem of deciding how to cluster them into chains. When detailed measurements are not available it may be necessary to use only a few chains due to the inability to distinguish individual job's resource usage. These and other issues are discussed in [121,180]. The theoretical results on the sensitivity of product form network results that were discussed in Section 2 give some indication, perhaps, of why queueing network models have been proved useful even though many of the assumptions are untrue. Ultimately, however, the justification is via empirical evidence.

6.3 Packet Switching Networks.

Queueing networks have been widely used to model packet switching networks, mainly at the network level in the OSI reference model. Basically, a store-and-forward packet switching network is a collection of resources (e.g., data links) shared by the data packets that travel through the network. The key feature in the model is the network topology and the routing of data packets, which is defined by level 3 in the OSI reference model.

In order to obtain an analytic solution for the network model, several simplifying assumptions are usually made. A key assumption is the "independence assumption"

[102] which states that the transmission times of a packet at each channel it visits are independent random variables. Obviously this is not true, since the length of a packet does not change as it travels through the network. However, simulation studies have shown that the assumption is quite good when several packet streams are merged on a transmission line, but it is known to fail in tandem queues with constant packet lengths [150].

In [114.201] a survey of models used in the analysis of packet switching networks is presented. Schwartz [159] has a good discussion of the analysis of flow control mechanisms. Reiser [146] also presents a survey of models used for packet switching networks. Below we summarize some of the models proposed in the literature.

The basic model used in early studies assumed that the packet sources, which represent protocols of layers 4, 3, 2 are Poisson sources. Furthermore, all packets are assumed to have the same average length and packets traveling from a given source to destination follow either a fixed route or stochastic routing (state dependent routing cannot be represented except in very special cases). Therefore, at least one routing chain is needed to model the routing behavior. Finally, processing time at the switching nodes and propagation delays are ignored (although they can be heuristically incorporated [102]).

The set of assumptions stated above are the main assumptions which ensure that the model possesses a product form solution (other assumptions include no time-out errors, infinite buffers at the queues, etc.). The queueing network model is open, and therefore the number of routing chains does not add computational complexity to the final solution.

With this basic model, a number of quantities of interest can be obtained [102,201] including: packet delay averaged over the whole network, mean packet delay for a source destination pair, average buffer occupancy at the nodes. For fixed routing, if the network has special characteristics called *overtake-free conditions* [195], then the distribution of end-to-end delays can also be calculated. The overtake-free conditions can be stated roughly as: (a) a packet belonging to a routing chain cannot overtake another packet of the same chain (e.g., when the discipline at the link queues is FCFS) and; (b) if two routing chains k_1 and k_2 share a link l_i, the "influences" generated by a packet P_{k_2} of chain k_2 behind packet P_{k_1} of chain k_1 cannot overtake packet P_{k_1} in another link l_j. For a typical example of an "influence", assume that packet P_{k_2} after leaving link l_i visits link l_m. Also assume that link l_m is shared with packets from chain k_3. Note that a packet from k_3 can be "influenced" by P_{k_2} and may visit link l_j and "influence" P_{k_1}. The distribution can be calculated exactly for packets belonging to chains which satisfy the overtake-free conditions. But it may also be possible to use the equations obtained as an approximation when the overtake-free conditions are violated. For a detailed discussion on sojourn times in queueing networks, we refer to the chapter by Boxma in this book.

Using the basic model a number of optimization problems can be formulated and solved, including [102,201]:

(a) Capacity assignment. The capacity of each link is selected such that the average packet delay is minimized subject to a constraint on maximum cost.

(b) Optimal routing. The stochastic (static) routing which minimizes the average

delay is obtained under the constraint of given link capacities. Although stochastic routing is generally not used in computer networks, since realistic networks implement some form of dynamic routing strategy, stochastic routing plays an important role in system network design. Note that static models are relatively more efficient to analyze than dynamic models, and can be used effectively for identification of bottlenecks, sensitivity studies, topological design, etc.

The general design problem in computer networks in which the topology, link capacities and stochastic routing are to be selected to minimize the overall average packet delay under a cost constraint is too complex to be solved exactly. Instead, a combination of the optimization methods described above is used to find an approximate solution [14.102].

Buffer management may have a significant impact on the performance of data networks. The number of buffers allocated to different queues is an important design issue. Lam and Reiser [113] developed a model based on previous work by Lam and Schweitzer [111,161] for evaluating the impact of input buffer limits on performance. In general, there is no exact analytic solution for queueing networks with blocking and an iterative approximation technique was developed based on an estimation of blocking probabilities due to buffer limits. From the model, throughput versus input load curves are obtained which show the decrease in throughput with increase in load due to buffer limitations.

There have been a number of studies analyzing window flow control mechanisms [14,159]. A key assumption made is that packets are lost when they arrive at the source of a flow controlled connection between two communicating processes and the window is closed (i.e., the maximum number of packets traveling on the connection is achieved). (In the following, we assume a virtual circuit connection.) This ("lost packet") assumption together with the independence assumption allows a tractable analytic model to be obtained. With the lost packet assumption (and other minor simplifying assumptions), a closed queueing network model with one chain for each virtual circuit in the network is developed [147]. If it is further assumed that packets from different virtual circuits have the same average length, then the model has a product form solution. Even with these simplifying assumptions, networks with many virtual circuits will have many distinct closed chains. The computational cost precludes the use of exact solution techniques, and approximation methods have to be used. (Real networks may have hundreds to thousands of virtual circuits open at a time.) A simple approach is to consider in detail a chosen virtual circuit and to model the effect of other virtual circuits by reducing the capacity of the links used by the given virtual circuit [140]. Other approximation techniques can be used as well. In real networks, however, each virtual circuit uses only a subset of the links in the network. In this case, medium size networks (say, on the order of 50 chains) can be solved exactly and efficiently by using tree algorithms [84,112,188].

If packets from different virtual circuits have different mean lengths, then the model described above does not have a product form solution and approximations have to be used. An accurate approximation to solve this problem was proposed by Reiser [147], based on the arrival theorem.

A number of optimization problems have been studied in addition to those mentioned above. For instance, Lazar [119] studied the following problem. Consider a

closed queueing network representing a packet switching network with virtual circuits. Before transmission, each packet belonging to a virtual circuit is fed into a controller which determines the rate of sending packets into the network. This rate is a function of the number of outstanding packets in the network. The optimization problem is to find the rates which maximize the overall throughput, subject to a constraint on the overall delay function and a constraint on the maximum control rate of each virtual circuit. It is shown that the optimal control policy is a window flow control mechanism for each virtual circuit. A variation of this problem is to find the control rate for optimizing the virtual circuit throughput subject to a constraint on the network delay for that virtual circuit [85]. To find the optimal window size for each virtual circuit requires the application of Norton's theorem for the network, which may be impractical. In [86], a simple example was presented to find the window sizes. In the example, it is assumed that a set of virtual circuits have a bottleneck link, modeled as an exponential FCFS server. Clearly, this assumption significantly reduces the complexity of the problem.

Kobayashi and Gerla [103] used the flow deviation method to solve the optimum routing problem in a network model with virtual circuits. The solution involves the application of MVA for each iteration of the algorithm (which may not be practical for large networks).

Gerla and Chan [68] studied the problem of finding the window size for a virtual circuit network which minimizes a measure of fairness P. P is defined as $P = \sum_{i=1}^{R} \frac{d_i - \lambda_i}{\lambda_i}$ where d_i is the traffic demand for virtual circuit i, i.e., the maximum throughput rate for virtual circuit i assuming no interference from other traffic. The minimization is subject to a constraint on the overall network delay. The proposed solution is heuristic. Starting from a unit window assignment for all virtual circuits, the incremental improvement of fairness per increment of average delay ($\Delta P_i/\Delta T_i$) is computed for an increment in the window size of each virtual circuit i. Then virtual circuit l such that $\Delta P_l/\Delta T_l = \min_i\{\Delta P_i/\Delta T_i\}$ is chosen, and its window size is incremented. In order to calculate $\Delta P_i/\Delta T_i$, Schweitzer's approximation is used. Numerical results were presented in [68] which suggest the approach is viable.

6.4 Circuit Switching Networks.

In a circuit switching network, in order to establish a communication path between a pair of nodes, the source node generates a *call* which sets up a circuit linking the source and destination nodes. The circuit consists of a set of transmission lines or links, connected by switches. Unlike a packet switching network, when a communication path is set up between a pair of nodes, a fixed amount of the total capacity of each link in the path is allocated exclusively to the connection, and released only when the communication is completed. (For instance, the capacity of each link could be divided into *time slots* in a time division multiplexed system, or into *frequency bands* in a frequency division multiplexed system. During the set up of a circuit, one or more slots or frequency bands from each link in the path are allocated to the call.)

Consider a call k generated by node u to establish a circuit between nodes u and v. Assume that the call requests c_k units of capacity to be allocated between u and v. The call is *blocked* if a path between u and v cannot be found in which all links

have at least c_k units of capacity available to be allocated to the call. One important measure for such networks is the probability of blocking.

For the networks considered immediately below some addditional assumptions are made. We assume that there are K types of connections. Distinct types represent connections between distinct pairs of nodes. Furthermore, the path used by a type of connection is fixed and unique. The arrival process of type k calls is assumed to be Poisson with rate λ_k. The duration (holding time) of a type k call is a random variable with arbitrary continuous density with mean $1/\mu_k$. The arrival processes and call holding times are assumed to be independent, and call set up and call release are instantaneous. It is also assumed that all calls require the same amount of link capacity and this is taken as the unit of link capacity, i.e., $c_k = 1 for all k$.

Let $n_k(t)$ be the number of calls of type k in progress at time t. If the call holding times are exponentially distributed, then the system evolves according to a continuous time Markov process $\mathcal{N} = \{n_k(t) : k = 0, \ldots, K\}$, with discrete state space $\mathcal{S} = \{\vec{n} : \vec{n} = (n_1, \ldots, n_K), n_k \geq 0, \text{ integer}\}$. Let $P(\vec{n})$ be the stationary state distribution of the process. Burman *et al* [24] have shown that $P(\vec{n})$ has product form:

$$P(\vec{n}) = \frac{1}{G} \prod_{i=1}^{K} \frac{1}{n_i!} \left(\frac{\lambda_i}{\mu_i}\right)^{n_i} \tag{43}$$

where G is the normalization constant of the network satisfying:

$$G = \sum_{\vec{n} \in \mathcal{S}} \prod_{i=1}^{K} P(\vec{n}) \tag{44}$$

and $\mathcal{S}$ is determined by the routing of the calls, and is such that, for all links j:

$$\sum_{i=1}^{K} n_i I_{ij} \leq C_j \tag{45}$$

$$I_{ij} = \begin{cases} 1 & \text{if type } i \text{ call is routed via link } j \\ 0 & \text{otherwise} \end{cases}$$

The proof of Eq. (43), presented in [24], is simple and servers as an illustration of the application of concepts presented in Section 2. If we assume infinite capacity for all the links in the network, then the model reduces to an IS service center with K types of customers and exponential service times. It is easy to show that this new process $\mathcal{N}'$ (with state space $\mathcal{S}'$) is reversible and satisfies Eq. (43). The state space of the original process $\mathcal{N}$ is $\mathcal{S}'$ (the state space of process $\mathcal{N}'$) truncated to the set $\mathcal{S} \subset \mathcal{S}'$. Thus, from [96], this process is reversible and the equilibrium distribution is the probability that $\mathcal{N}'$ is in state $\vec{n}'$ conditioned that $\vec{n}' \in \mathcal{S}$. Burman *et al* [24] also show that Eq. (43) is valid for general holding time distributions.

The normalization constant for this problem cannot be calculated from the standard algorithms for product form queueing networks. Dziong and Roberts [57] obtained recursive expressions which allow the calculation of the blocking probabilities for a network. However, the calculation is costly and practical only for small networks. In [57], approximations for calculating the blocking probabilities are proposed.

The above model can be extended to allow state dependent arrival rates and different type of calls requiring different amounts of link capacity [57]. In [35], Conway

also proposed extensions to the basic model in which links may fail and are repaired independently of each other. The repair time of each link is given by a continuous (but otherwise arbitrary) density. It is assumed that calls routed through failed links continue to hold the capacity allocated to them in other links, and resume transmission immediately after all failed links in their path are repaired. Another extension in [35] allows random selection of a path at call set up time, from the set of possible paths between the source and destination.

6.5 Availability.

Obtaining availability measures of repairable computer systems often requires numerical solution of a Markov chain model of the system [183]. However, in certain cases, a queueing network can be used to model the failure/repair behavior of the system [73]. If certain restrictions are imposed, (e.g., the failure of a component cannot affect another component) the queueing network will have a product form solution. In this case, availability measures can be evaluated in a much less costly manner (in terms of computation time and storage requirements) than conventional numerical solutions. The queueing network obtained typically has one closed chain for each type of component in the system [73]. A component cycles between a queue, which models the failure rate of the component, and one or more queues representing the repair center.

Consider a set of N components of the same type (same failure/repair behavior) each with failure rate λ. If none of them is a spare, then the composite failure rate is $n\lambda$, whenever there are n components working. Obviously, this behavior can be represented by an IS queue. Another representation is used when some of the components are spares which fail with a different rate than that of the working component (λ_s, possibly zero). In this case, if there are n components working but at most m of them can work in parallel, then $\max(n-m,0)$ are spares and the total failure rate is $\lambda \times \min(m,n) + \lambda_s \times \max(n-m,0)$. A queue dependent service center can be used to represent this failure rate.

The queueing discipline of the repair queue may be any of the well known disciplines which lead to product form. Among the most useful for availability modeling are: IS, LCFSPR, FCFS, "random order of service preemptive resume" (with resampling at arrival times) [73]. (This last discipline is an example of a case where it is useful to have the product form characterization results to determine if a new discipline will preserve product form.)

As a specific example, consider a model of a distributed architecture database system. In this model, there are three processors (one of them is a spare), two disk controllers and three clusters of disks, each containing four dual ported disks. In each disk cluster data is replicated such that all data is accessible if any subset of three disks is accessible. Thus, any disk unit in a cluster may fail without the system losing access to any data. If a processor fails, the spare immediately replaces the failed unit. If two processors have failed, the system is still operational but in a degraded mode. Each processor may fail in two modes: a soft failure mode and a hard failure mode. In the first mode the system can be restarted automatically (if there is a spare processor, the spare substitutes for the failed unit during the restart). The restart process takes an average of $1/\mu_s$ hours. The second mode requires the intervention of a repairman.

The processors are linked to both disk controllers. Each disk controller is linked to all disks, and therefore, one disk can fail without affecting data access. The failure rate of an active processor, a spare processor, a controller and a disk unit of cluster i is $\lambda_p, \lambda_s, \lambda_c$ and λ_{d_i}, respectively. The probability of a processor failing in soft (hard) mode is p $((1-p))$. There is only one repairman who repairs units in FCFS order, and it takes an average if $1/\mu_r$ hours to repair a unit. We also assume independent exponential failure times.

Figures 15 and 16 show the architecture and queueing model for this example. There are 5 chains, one for each set of components with the same behavior, and 3

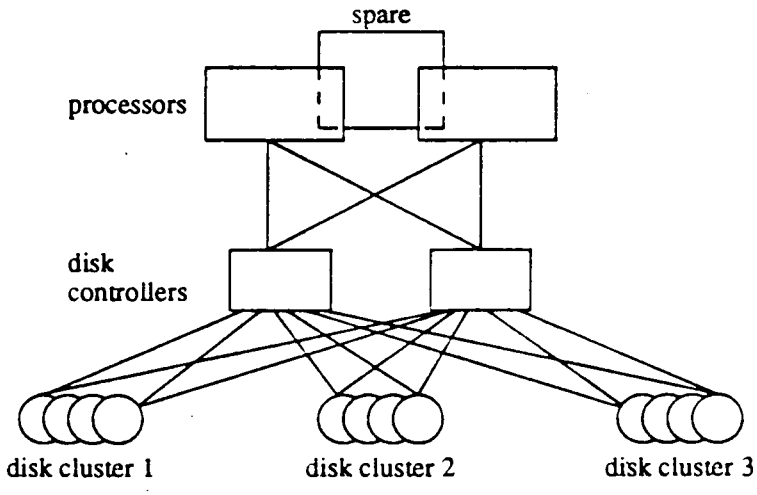

Figure 15: A distributed architecture for a database system.

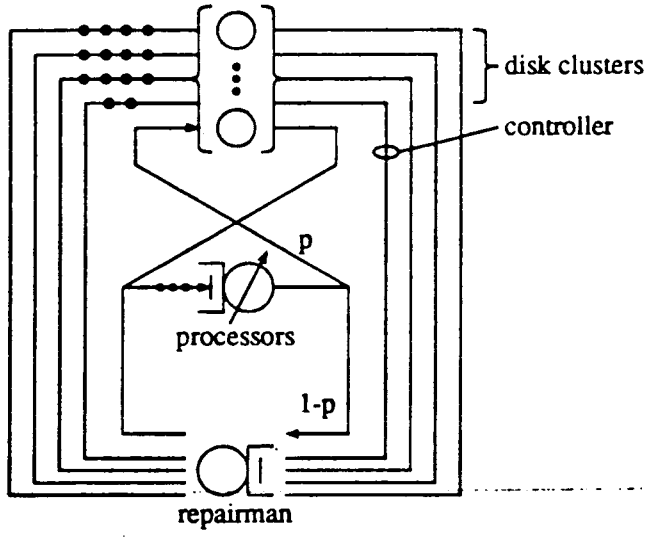

Figure 16: A queueing model for the distributed database system.

centers. The IS center models the failure behavior of the disk clusters, disk controllers and the restart process of soft failures of a processor. The center labeled "processors"

models the failure behavior of the processors. Note that the service rate of this center changes according to the number of spare and active modules, i.e., $\mu_p(3)/a_{pk} = \lambda_s + 2\lambda_p$; $\mu_p(2)/a_{pk} = 2\lambda_p$; $\mu_p(1)/a_{pk} = \lambda_p$ (where p and k are the indices of the center and chain for the processors, respectively). The center labeled "repairman" models the repair facility.

In order to calculate availability measures, we need to obtain joint queue length probabilities (see example below) and possibly state probabilities as well. The DAC algorithm described in Section 4 is particularly efficient for these calculations. In the example above, the percentage of time the system operates with all components working is simply $P^{17}(14;3;0)$ ($n = (14;3;0) \in S^K$), which is readily available from the basic algorithm. On the other hand, a small subset of state probabilities also needs to be calculated in order to obtain the steady state availability. These state probabilities can also be efficiently calculated using DAC.

In availability modeling, it is important to calculate the sensitivity of availability measures with respect to input parameters such as repair and failure rates. To evaluate these sensitivity measures, we need to obtain the partial derivatives of $P^K(\vec{n})$ with respect to the input parameter of interest. In most cases, partial derivatives can be easily calculated using the same approach as in [53,169].

6.6 Multicomputer Systems.

6.6.1 Database Systems/Machines.

Database systems represent one major application of computer systems. They also represent a number of challenges in modeling.

Recently Heidelberger and Lakshmi [77] used queueing network models to compare the performance of two database system architectures: a mainframe database and a database machine architecture based on multiple microcomputers. This work is an excellent example of how simple queueing network models can provide useful information about the expected behavior of systems and can help the analyst to make important decisions.

The architecture of the mainframe database management system consists of a tightly coupled multiprocessor running a simple operating system and a number of disk drivers. The architecture of the database machine consists of a front-end (assumed to be a tightly coupled multiprocessor) for terminal management, application processing, etc.; query processors which are responsible for parsing, optimization, translation of queries and dispatching them for the access processors, maintaining dictionary data, etc.; access processors for accessing the relations (a relational database is assumed), performing concurrency control and recovery functions. The relations in the database are distributed across the disks. The query and access processors are assumed to be microprocessors and to have only local memory.

The main assumption is that the queueing model satisfies product form requirements (contention for locks is negligible, commit processing is performed sequentially, etc.). Other simplifying assumptions are also made in comparing architectures. Among them we mention equal memory hierarchy performance for the two architectures; the given speed ratings (MIPS rating) for the mainframe processors and

microprocessors of the database machine are considered equivalent.

The queueing network models for the two systems are shown in Figures 17 and 18. Both models have only a single open chain representing the transactions which

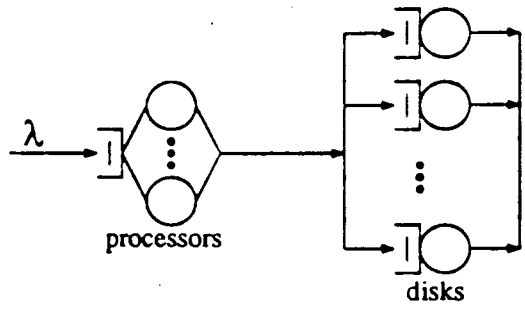

Figure 17: A queueing network model for a mainframe database management system.

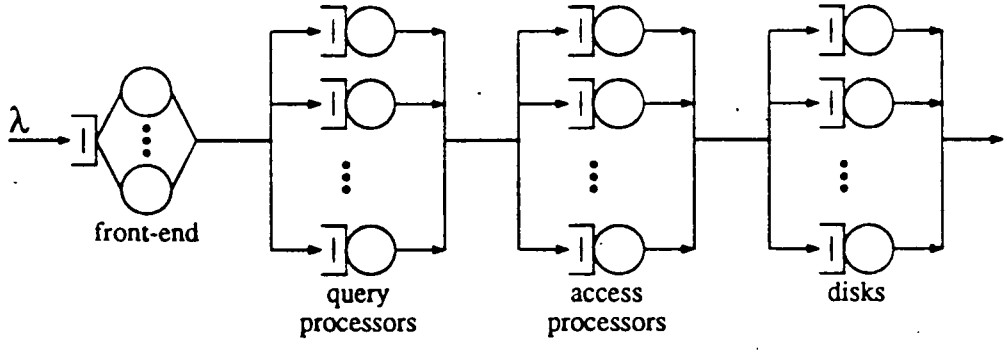

Figure 18: A queueing network model for a database machine.

visit the processors and disks a number of times before leaving. The multiserver queue of Figure 17 represents the tightly coupled multiprocessor architecture of the mainframe. The disk subsystem is modeled by a number of SSFR queues. The relative utilization for each resource in Figure (17) is obtained as a function of the average number of database system calls in the transaction, average number of I/O's per call, average number of I/O's during commit, average number of instructions executed by the transaction, main memory buffer's miss ratio, etc.

In Figure 18, the multi-server queue represents the front-end host which is also a tightly coupled multiprocessor. SSFR queues represent the query processors, access processors and the disks. The relative utilizations of the system resources are determined as in the mainframe model.

For comparison purposes, a measure called MIPS penalty is defined as the sum of the microprocessors MIPS of the database machine divided by the difference between the mainframe MIPS and the MIPS of the database machine front-end. Note that, if the MIPS penalty is greater than 1, then the total number of MIPS of the database machine (front-end plus multiprocessors) is greater than the total number of MIPS of

the mainframe. This measure captures the effect that in general the database machine requires more total MIPS than the mainframe to achive a given response time goal to execute identical tasks. Several performance studies were carried out using this basic model. For instance, studies to compare the mean response time of the two architectures as a function of the throughput for database machines with different MIPS penalties; studies to verify the sensitivity of the architectures with respect to distributed processing overhead, buffer miss ratio and other input parameters.

6.6.2 Multiprocessors and Interconnection Networks.

In the past few years a number of papers devoted to the analysis of multiprocessors and their interconnection network have appeared. Roughly, a multiprocessor is a collection of processor/memory units which cooperate by sending/receiving data directly among themselves or by accessing global shared memory units. In both cases, data is sent through an interconnection network which links the cooperating units. The type of interconnection network may vary from simple shared busses to more complex omega networks. In general, the modeler is concerned with the interference which occurs when the interconnection network and global memory units are accessed. Below we survey some solution techniques that have been employed to analyze these systems. Some of the work is based on solutions surveyed in previous sections.

As an example, consider a simple multiprocessor system with P processing units with local, non-shared memory, B global busses and M shared memory modules. In order to execute a task, a processing unit accesses data in its local memory and periodically, accesses data in any of the shared memory units. For data to be accessed, any of the B busses must be held, together with a memory unit which stores the data. Note that if $B \geq M$, there is no possibility of contention for the busses, and memory interference can be modeled as M independent queues. When $B = 1$, then the bus/memory subsystem behaves as a single server queue. The interesting case occurs when $B < M$ and $B > 1$. Irani and Onyuksel [89] obtained a closed form expression for a symmetric system with exponential service times. Le Boudec [17] was able to show that the bus/memory subsystem can be modeled by a special station, which leads to product form solution when used in a BCMP-type [12] network, provided that the following properties hold for the station:

(a) there are B identical exponential servers;

(b) there are M classes of users;

(c) a service request from a tagged customer of class m is granted if: (1) there is a server available and; (2) there is no other class m customer being served and; (3) all customers still in queue which arrive before the tagged customer cannot be served (i.e., they belong to any of the classes of customers in service).

It is easy to see that this station represents the contention for bus/memory, if there are as many servers as busses, as many classes as global memory units and the memory access time is the server service time. Furthermore, it is interesting to note that the bus/memory subsystem model yields a simultaneous resource possession problem and it is surprising that the model satisfies product form requirements. Due to the

solution complexity of this special station, the product form algorithms surveyed previously cannot be used to solve networks with these stations, and new algorithms were developed in [18,32].

Harrison and Patel [75] studied the contention for unbuffered omega networks. In the model, there are P parallel queues, each one representing a data transfer device, in which requests are queued. A transfer occurs if a path in the interconnection network can be found between a source and a destination. For the analysis, the P queues are collapsed into a flow equivalent queue where the state represents the number of devices that have a request to transmit. Given that n devices are busy, the probability that m requests are satisfied is calculated based on the topology of the network and simple combinatorial arguments. From the solution of the flow equivalent server and the above probabilities, the effective throughput rate of the network is calculated. In [138] the effect of non-uniform requests in the network is studied.

Kruskal *et al* [107] consider buffered synchronous (discrete time) omega networks. Their analysis is derived for the first stage only, and it reduces to the analysis of a single discrete time queue. The results are then extended by approximation for the remaining stages. Christos *et al* [34] develop bounds for the mean delays for the second and subsequent stages, also based on single queue analysis.

Fredericks [65] analyzes a simple buffered circuit switching network with two stages (input ports and output ports), where both input and output servers must be held simultaneously in order to start transmission. The solution is similar to the ones surveyed in Section 4.4.2 for the simultaneous resource possession problem, with minor variations.

Reed [145] studies the performance of microcomputers linked through an interconnection network. He develops a technique to calculate the visit ratios of messages through the communication links. This technique avoids the solution of a large number of linear equations that would result in a large model, and yet preserves some information about the structure of the network. With these visit ratios and given service requirements, he uses bottleneck analysis [134] to obtain bounds on performance measures. Furthermore, trade-offs between different topologies are studied using Balanced Job Bound analysis [208].

In [124] Leutenegger and Vernon give an interesting example of the use of simple mean value arguments to study the performance of a multiprocessor system with a complex structure: the Wisconsin Multicube. The system studied consists of processor units (with cache memories) connected via a grid network. Shared global memories are distributed across column busses. Interference for cache and global memory are assumed to have secondary effects in comparison to bus interference, and it is ignored. To obtain the solution, it is assumed that the resulting queueing network model satisfies product form requirements. Furthermore, Reiser's approximation (see Section 4.4.1) is used for FCFS centers. The model allows the evaluation of several design trade-offs such as the effect of cache block sizes and the speed up as a function of the number of processors.

Norton and Pfister [137] also analyzed a multiprocessor machine: the IBM RP3. The model solution is an example of hierarchical decomposition surveyed in Section 4.3.3. The overall model is decomposed into two submodels: one for the processor and one which represents the contention for the network/memory subnetwork. (The

effects of the respective complements are represented in a simplified way.) The two sets of equations obtained are reduced to a single point equation.

6.7 Distributed System Modeling.

The term "distributed system" has been used to describe a variety of different system architectures. In this subsection we restrict our discussion to local area network of computer systems which runs a single distributed operating system. The operating system is assumed to support a number of features, such as transparent access to data through the network, transparent distributed process execution and automatic replication of storage. (The UCLA-LOCUS system [142] is an example of such a distributed system.) Many interesting performance issues arise in such systems. For instance, how much can the system grow without degradation in overall performance (i.e. how well does it scale)? Replicating files through different sites to improve file access reliability may have an adverse effect on performance, since changes in a copy must be propagated to other copies. What is the "best" replication factor? Should requests to access remote resources be given priority over local requests?

One of the first distributed system models was presented in [70]. In that paper, each site in the network is represented by a number of queues which model the resources of the site (e.g., CPU, disks). In general, the resources of one site are accessible not only to jobs generated by users logged on to that site, but also to jobs generated remotely as well. Therefore, when a user submits a job, besides accessing the local processor and disks, the job may need a remote ("foreign") resource. It then "visits" the network and service centers on the foreign site. Through the communication network, it sends request messages, perhaps some data necessary to run the request remotely, and the response messages are returned. In order to represent the routing behavior of a job in the system, at least one routing chain is used for jobs generated by users logged on to a particular site. Figure 19 illustrates the queueing network model obtained. In this figure, a central server model is used to represent the resources at each site, a single closed chain is used to model the routing of jobs generated at a site and a single queue represents the contention for the communication network.

In [70], the effect of updating files replicated through the network is studied. Whenever a file is closed by a job, all modifications are propagated through the system, and the job continues to run asynchronously with the updates. To represent this effect, the model in [81] which was discussed in Section 4.4.3 is used. In [70], the performance measures predicted by the model are compared with measurements made in a system with four Vax computers running LOCUS in which user behavior is simulated by a synthetic benchmark. The results reported are encouragingly accurate.

The above work ignores the priority given to file updates which run at a foreign site. In [15], these priorities are taken into account. The model is essentially the one in [70], but priority servers are included. In order to solve the model, the approach of [22] is used. It is shown that the model with priorities predicts much more accurately the system response time than the model which ignores priorities. In [15], the effect of increasing the degree of file replication is also studied.

Both models discussed above use a single FCFS center to represent an Ethernet communication network channel. In [49] the use of a more accurate representation

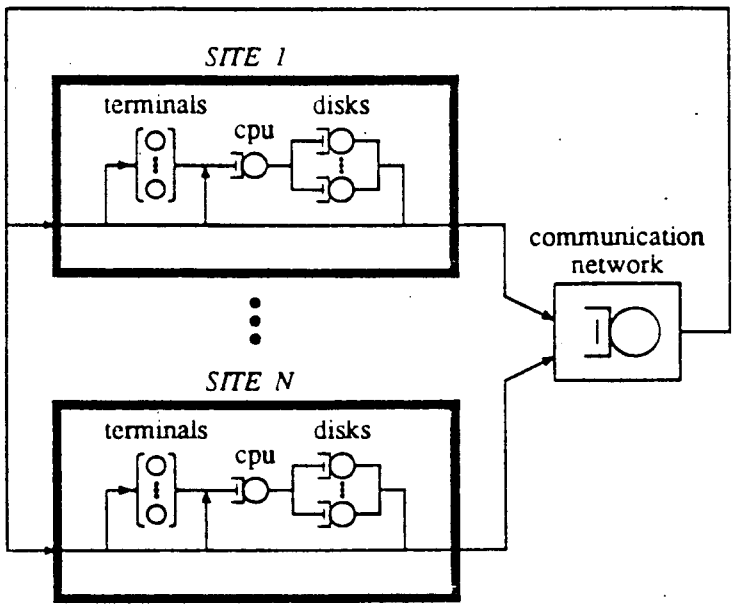

Figure 19: A queueing network model of a distributed system.

of the Ethernet channel embedded in the overall system model is proposed. The approach uses the clustering approximation technique described in Section 4.3.3. In [49], it is concluded that, for very light load in the channel, the FCFS center provides a good approximation, but when the number of sites increases, the channel becomes more utilized and influences the overall response time of jobs. Comparisons with simulation demonstrate that the results obtained from the analytical model are accurate.

As a final note, the basic model of Figure 19 was used in optimization problems as mentioned in the previous section.

6.8 Synchronization in Parallel Systems.

Except for a few special cases, product form networks require that customers can only occupy one resource at a time and that the only interaction between customers is due to contention for these resources. For many applications, other forms of customer interaction are first order effects and must be represented to obtain accurate models. This is particularly true in modeling the computation structure of interacting concurrent programs. Interactions between programs (customers) manifest themselves, for example, in the requirements for mutual exclusion and/or precedence relations among subtasks. The task spawning and the "spawn-join" behavior [80,81] that were discussed earlier are particular cases of this type of interaction.

The following are some of the major types of concurrency and synchronization features that have been studied.

1. spawning independent subtasks.

   In this case the model can be viewed [81] as having a set of primary customers. These primary customers occasionally spawn additional customers.

The spawned customers, like those of an open chain, have transition probabilities such that they eventually depart from the network with probability one. This type of non-product form behavior is not unusual in computer systems. For example, an interactive system may respond to the user while still doing some post processing of the previous request. Note that this type of behavior does not involve synchronization with the completion of the spawned tasks, i.e., the "parent" tasks and the spawned tasks are all mutually independent except for contention for resources.

2. task synchronization

   The feature that distinguishes this case from the previous one is more complex synchronization between task completions and initiations. For example, in [80], a parent task is blocked until the completion of all of the spawned subtasks. Many other forms of synchronization between tasks are possible. In [176,177], a computation is described as a partially ordered set of tasks where the partial ordering is a precedence relation. In this model a task is initiated at the instant when all of its immediate predecessors have completed. One may be interested in such models for computations that execute and then, when all task have completed, the job departs. An alternative is a model in which the job is immediately reinitiated when it "completes". Note that the partial order of tasks can be considered as a macro level model. Each task may have a "micro" behavioral description which describes the load on physical resources. In [178], a more general computation graph structure is used to specify CPU-I/O and I/O:I/O concurrency within a job. [8] and [193] suggest that some combination of timed Petri nets and queueing networks be used for model specification. [193] discusses the trade-off between the modeling capability of timed Petri nets and the efficient solutions available for product form queueing networks.

3. serialization delays

   This is a special form of contention which arises in computer systems due to shared resources that must be used in a mutually exclusive manner, i.e., by only one job at a time. Shared data structures are the most common example. When a job requires access to the resource and the resource is in use, then the requesting job must be blocked. We can therefore associate a queue with each such shared resource. Serialization delays are a special case of simultaneous resource possession. In this case there is a single "resource" analogous to a single server queue. For simultaneous resource possession in general, there may be a pool of identical resources (e.g., memory partitions) which is analogous to a multiple server queue.

   There are also special cases of this type of problem in terms of the (secondary) resources that are accessed while holding the primary resource. In the simplest case, the set of secondary resources associated with the primary resources are mutually disjoint. This is, for example, assumed in the surrogate delay method in [93]. In the case of serialization delays, this is often not a tenable assumption. For example, whether accessing a shared data structure, or not in any critical section, computations will be accessing shared physical resources (CPU, disks, etc.).

It is easy to see that the types of interaction mentioned above restrict the execution of jobs and can have a significant effect on performance. For performance prediction, these features may have to be represented to obtain accurate results. Models incorporating these features are also useful in evaluating design alternatives. As an example, the number of distinct serialization delays is often a design variable. It is possible to treat several shared data structures as one for the purpose of achieving mutual exclusion. At the extreme, there could be only one critical section for all shared data. With less critical sections there is less overhead due to requesting and releasing access to shared data; there is however, additional serialization delay. The tradeoffs can be evaluated using models incorporating these features [5,166,174]. Another issue is the effect on performance of more or less parallelism within a computation. The ability to represent concurrency and synchronization enables the evaluation of design alternatives involving these features, e.g., the effect of executing portions of a computation in parallel. [178] explores the effect of CPU-I/O concurrency within a computation and also of I/O-I/O concurrency. Heidelberger and Trivedi [80,81], as discussed previously, study models that incorporate spawning independent tasks and also the spawning of subtasks that then must synchronize with the parent task at completion. Thomasian and Bay [176,177] model computations as partially ordered tasks.

There are two main approaches to solving models that incorporate concurrency and synchronization features in a model. These are: iterative approximations and decomposition. Iterative approximations have been described in Section 4. Examples of this approach as applied to concurrent computations can be found in [5,80,81,177].

The second approach is the use of decomposition. In this approach, the synchronization between jobs is used to define the aggregation of states of the model. For example, consider a model in which all jobs are statistically identical, and there are K critical sections. A detailed state for such a model will have to include the queue at each critical section and also the distribution of active customers among the physical resources. We can, however, consider aggregation of all states in which the same number of customers are queued at each critical section, i.e., an aggregate state is defined by $(n_0, n_1, n_2, ..., n_K)$, where n_0 is the number of customers that are not queued or accessing any shared data structure and n_i is the number queued for critical section i. Thomasian [174] refers to the aggregate states as the "system states". The assumption that lock requests and releases are relatively infrequent relative to the transitions among the physical resources suggests that decomposition can be applied. Closed queueing network results may be used to analyze behavior within an aggregate state and to obtain the transition rates between aggregates. In cases where both iteration and decomposition have been considered and compared [80,81,174], it appears that decomposition is generally more accurate. However, it can also be considerably more expensive since many subnetworks may have to be solved, and the aggregate level model is generally not product form and must be solved numerically.

In [174] the main emphasis is on representing precedence relations between tasks (represented as a partial order). It is shown that the top level Markov chain has a special form and can be solved efficiently. In [95] the same type of aggregation is applied to models of parallel computation. However, in this work the resulting Markov chain was considered to be potentially too large to be efficiently solvable, and

a "fluid flow" approximation was developed for solution of the top level model.

7 Summary.

It is evident that queueing networks have been found to be useful in many areas of computer system performance evaluation. These models are also being used extensively in other areas such as flexible manufacturing [25]. The application areas in which queueing networks have proven to be effective were surveyed in some detail in Section 6. These will continue to be important applications and one can expect refinements in these areas. There is, however, a serious issue as to what role queueing networks will play in the future as highly parallel computer system architectures become common and parallel and distributed processing are the major concerns. While there has been some work on modeling concurrency and synchronization, it remains to be seen how accurately these systems can be modeled and exactly what role queueing network technology will play.

It seems clear that without major breakthroughs exact results will not be generalized significantly. There are then two evident directions for research: look for those breakthroughs or concentrate on bounds and approximations. The former is perhaps more exciting and holds more promise in the long range. The latter is more likely to produce useful results in the short term. Both research directions are important. With respect to bounds and approximations, we believe that formal analysis of bounds and approximations should be emphasized. There are several approaches that are of current research interest. The work of Courtois and Semal on extracting bounds from large Markov models [44.45] is an interesting and promising direction. This is an approach to finding bounds on performance measures without actually generating the entire transition rate matrix. The hope is that it will be generally applicable to Markov models since it is based on numeric solution but will conquer the state space problem by obtaining bounds from a representation of only a small portion of the Markov chain. The bounding methodology of van Dijk [189], the asymptotic methods [104,127,130,141], etc. are examples of other approximation methods currently being explored.

The accomplishments to date have been impressive. Queueing networks has emerged as a new field and enjoyed considerable success both in practice and in theory. A large part of the success is due to the symbiotic relationship of practical application and theory. Often the theory has developed in response to the needs of the applications. It remains to be seen if advances in queueing network theory can match the demands of the complex, highly parallel computer systems of the 1990's.

Acknowledgements. We would like to thank the following people for careful reading and thoughtful comments that were of great value to us: Bill Cheng, Richard Gail, Tom Page, and Alex Thomasian.

References

[1] BEST/1 user guide. BGS Systems, Inc. Waltham, MA, 1982.

[2] MAP user guide. Quantitative System Performance, Inc. Seattle, WA, 1982.

[3] PAWS/A user guide. Information Research Associates, Austin, TX, 1983.

[4] Q+: the AT&T performance analysis workstation: user guide and reference manual. AT&T Bell Laboratories, Holmdel, NJ 1988.

[5] S.C. Agrawal and J.P. Buzen. The aggregate server method for analyzing serialization delays in computer systems. *ACM Trans. on Computer Systems*, 1:116–143, 1983.

[6] S.C. Agrawal and P.J. Denning. *Convergence and Stability of Iterative Queueing Network Models*. Technical Report, RIACS, July 1984.

[7] I.F. Akyildiz and A. Sieber. Approximate analysis of load dependent general queueing networks. *IEEE Trans. on Computers*, 14:1537–1545, 1988.

[8] G. Balbo, S. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena. *IEEE Transactions on Software Engineering*, SE-12:561–576, 1986.

[9] Y. Bard. An analytic model of the VM/370 system. *IBM Journal of Research and Development*, 22:498–508, 1978.

[10] Y. Bard. A simple approach to system modeling. *Performance Evaluation*, 1:225–248, 1981.

[11] Y. Bard. Some extensions to multiclass queueing network analysis. In M. Arato, A. Butrimenko, and E. Gelenbe, editors, *Performance of Computer Systems*, pages 51–61, North-Holland, 1979.

[12] F. Baskett, 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*, 22:248–260, 1975.

[13] F. Baskett and F.P. Gomez. Processor sharing in a central server queueing model of multiprogramming with applications. *Proc. Sixth Annual Princeton Conf. Information Sciences and Systems*, 598–603, 1972.

[14] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, New Jersey, 1987.

[15] J. Betser, M. Gerla, and G. Popek. A dual priority MVA model for a large distributed system. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 51–66, 1984.

[16] A.B. Bondi and Y.M. Chuang. A new MVA-based approximation for closed queueing networks with a preemptive priority server. *Performance Evaluation*, 8:195–221, 1988.

[17] J.Y. Le Boudec. A BCMP extension to multiserver stations with concurrent classes of customers. In *Proc. Performance '86 and 1986 ACM SIGMETRICS Conf.*, pages 78–91, 1986.

[18] J.Y. Le Boudec. The multibus algorithm. *Performance Evaluation*, 8:1–18, 1988.

[19] A. Brandwajn. Fast approximate solution of multiprogramming models. In *Proc. 1982 ACM SIGMETRICS Conf.*, pages 141–149, 1982.

[20] A. Brandwajn and W.M. McCormack. Efficient approximation for models of multiprogramming with shared domains. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 186–194, 1984.

[21] S.C. Bruell and G. Balbo. *Computational Algorithms for Single and Multiple Class Closed Queueing Network Models.* North Holland, 1980.

[22] R.M. Bryant, A.E. Krzesinski, M.S. Lakshmi, and K.M. Chandy. The MVA priority approximation. *ACM Trans. on Computer Systems*, 2:335–359, 1984.

[23] D. Burman. Insensitivity in queueing systems. *Adv. App. Prob.*, 13:846–859, 1981.

[24] D. Burman, J.P. Lehoczky, and Y. Lim. Insensitivity of blocking probabilities in a circuit-switching network. *J. App. Prob.*, 21:850–859, 1984.

[25] J.A. Buzacott and D.D. Yao. On queueing network models of flexible manufacturing systems. *Queueing Systems*, 1:5–28, 1986.

[26] J.P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16:527–531, 1973.

[27] J.P. Buzen. *Queueing Network Models of Multiprogramming.* PhD thesis. Div. Engineering and Applied Phys., Harward University, Cambridge, MA, 1971.

[28] K.M. Chandy, U. Herzog, and L.S. Woo. Parametric analysis of queueing networks. *IBM Journal of Research and Development*, 19:43–49, 1975.

[29] K.M. Chandy and A.J. Martin. A characterization of product-form queueing networks. *Journal of the ACM*, 30:286–299, 1983.

[30] K.M. Chandy and D. Neuse. Linearizer: a heuristic algorithm for queueing network models of computer systems. *Communications of the ACM*, 25:126–134, 1982.

[31] K.M. Chandy and C.H. Sauer. Computational algorithms for product form queueing networks. *Communications of the ACM*, 23:573–583, 1980.

[32] G. Chiola, M.A. Marsan, and G. Balbo. Product-form solution techniques for the performance analysis of multiple-bus multiprocessor systems with nonuniform memory references. *IEEE Trans. on Computers*, 37:532–540, 1988.

[33] Y.C. Chow and W.H. Kohler. Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Trans. on Computers*, 28:354–361, 1979.

[34] B. Christos, G. John S. Paul, and T. Vassilis. Queueing delays in buffered multistage interconnection networks. In *Proc. 1987 ACM SIGMETRICS Conf.*, pages 111–121, 1987.

[35] A. Conway. Product-form and insensitivity in circuit-switched networks with failing links. *Performance Evaluation*, 9:209–215, 1989.

[36] A.E. Conway. Fast approximate solution of queueing networks with multi-server chain-dependent FCFS queues. In *4th Inter. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation*, Spain, September 1988.

[37] A.E. Conway, E. de Souza e Silva, and S.S. Lavenberg. Mean value analysis by chain of product form queueing networks. *IEEE Trans. on Computers*, C-38:432–442, 1989.

[38] A.E. Conway and N.D. Georganas. *Queueing Networks-- Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation.* MIT Press, 1989.

[39] A.E. Conway and N.D. Georganas. RECAL - a new efficient algorithm for the exact analysis of multiple-chain closed queueing networks. *Journal of the ACM*, 33:768–791, 1986.

[40] P.J. Courtois. Decomposability, instabilities and saturation in multiprogram-

ming systems. *Communications of the ACM*, 18:371–377, 1975.

[41] P.J. Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, 1977.

[42] P.J. Courtois. Error analysis in nearly completely decomposable stochastic systems. *Econometrica*, 43:691–709, 1975.

[43] P.J. Courtois and P. Semal. Bounds for the positive eigenvectors of nonnegative matrices and for their approximations. *Journal of the ACM*, 31:804– 825, 1984.

[44] P.J. Courtois and P. Semal. Bounds on conditional steady-state distributions in large Markovian and queueing models. In J.W. Cohen O.J. Boxma and H.C. Tijms, editors, *Teletraffic Analysis and Computer Performance Evaluation*, North Holland, 1986.

[45] P.J. Courtois and P. Semal. Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models. *IEEE Journal on Selected Areas in Communications*, SAC-4:926–937, 1986.

[46] E. de Souza e Silva and M. Gerla. Load balancing in distributed systems with multiple classes and site constraints. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 17–33, 1984.

[47] E. de Souza e Silva and M. Gerla. *Queueing Network Models for Load Balancing in Distributed Systems*. Technical Report CSD-870069, UCLA Computer Science Department, December 1987.

[48] E. de Souza e Silva and S.S. Lavenberg. Calculating joint queue length distributions in product form queueing networks. *Journal of the ACM*, 36:194–207, 1989.

[49] E. de Souza e Silva, S.S. Lavenberg, and R.R. Muntz. A clustering approximation technique for queueing network models with a large number of chains. *IEEE Trans. on Computers*, C-35:419–430, 1986.

[50] E. de Souza e Silva, S.S. Lavenberg, and R.R. Muntz. A perspective on iterative methods for the approximate analysis of closed queueing networks. In G. Iazeola, P.J. Courtois, and A. Hordijk, editors, *Mathematical Computer Performance and Reliability*, pages 225–244, North Holland, 1984.

[51] E. de Souza e Silva and R.R. Muntz. Approximate solutions for a class of non-product form queueing network models. *Performance Evaluation*, 7:221–242, 1987.

[52] E. de Souza e Silva and R.R. Muntz. A note on the computational cost of the linearizer algorithm for queueing networks. *to appear, IEEE Transactions on Computers*, 1990.

[53] E. de Souza e Silva and R.R. Muntz. Simple relationships among moments of queueing lengths in product form queueing networks. *IEEE Trans. on Computers*, 37:1125–1129, 1988.

[54] P.J. Denning and J.P. Buzen. The operational analysis of queueing network models. *ACM Computing Surveys*, 10:225–261, 1978.

[55] L.W. Dowdy and D.V. Foster. Comparative models of the file assignment problem. *ACM Computing Surveys*, 14:287–313, 1982.

[56] L.W. Dowdy, A.T. Krantz, and S.K. Tripathi. *Single Class Bounds of Multi-Class Queuing Networks*. Technical Report, Department of Computer Science, Vandrbilt Univ., 1989.

[57] Z. Dziong and J.W. Roberts. Congestion probabilities in a circuit-switched

integrated services network. *Performance Evaluation*, 7:267–284, 1987.

[58] D.L. Eager, E.D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12:662–675, 1986.

[59] D.L. Eager and J.N. Lipscomb. The AMVA priority approximation. *Performance Evaluation*, 8:173–193, 1988.

[60] D.L. Eager and K.C. Sevcik. Bound hierarchies for multiple-class queueing networks. *Journal of the ACM*, 33:179–206, 1986.

[61] D.L. Eager and K.C. Sevcik. Performance bounds hierarchies for queueing networks. *ACM Trans. on Computer Systems*, 1:99–115, 1983.

[62] A.E. Ferdinand. An analysis of the machine interference model. *IBM Systems Journal*, 10:129–142, 1971.

[63] D. Ferrari. On the foundations of artificial workload design. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 8–14, 1984.

[64] L. Fratta, M. Gerla, and L. Kleinrock. The flow deviation method - an approach to store-and-forward communication network design. *Networks*, 3:97–133, 1973.

[65] A.A. Fredericks. An approximation method for analyzing a virtual circuit switch based LAN - solving the simultaneous resource possession problem. In *Teletraffic Analysis and Computer Performance Evaluation Proceedings*, pages 63–73, North-Holland, 1986.

[66] C. Gao, J.W.S. Liu, and M. Railey. Load balancing algorithms in homogeneous distributed systems. In *Proceedings of the 1984 International Conference on Parallel Processing*, pages 302–306, 1984.

[67] B. Gavish and S.L. Hantler. An algorithm for optimal routing selection in SNA networks. *IEEE Trans. on Computers*, 31:1154–1161, 1983.

[68] M. Gerla and H.W. Chan. Window selection in flow controlled networks. *Proceedings of the 9th Data Communication Symposium*, 84–92, September 1985.

[69] T. Giammo. Validation of a computer performance model of the exponential queueing network family. *Proc. ACM Int. Symp. Computer Performance Modeling, Measurement and Evaluation*, 44–58, 1976.

[70] A. Goldberg, G. Popek, and S.S. Lavenberg. A validated distributed system performance model. In *Proc. Performance '83*, pages 251–268, 1983.

[71] K.D. Gordon and L.W. Dowdy. The impact of certain parameter estimation errors in queueing network models. In *Proc. 1980 ACM SIGMETRICS Conf.*, pages 3–9, 1980.

[72] W.J. Gordon and G.F. Newell. Closed queueing systems with exponential servers. *Operations Research*, 15:254–265, 1967.

[73] A. Goyal, S.S. Lavenberg, and K.S. Trivedi. Probabilistic modeling of computer system availability. *Ann. of Operations Research*, 8:285–306, 1986.

[74] A.G. Greenberg and J. McKenna. Solution of closed, product form, queueing networks via the RECAL and tree-RECAL methods on a shared memory multiprocessor. In *Proc. Performance '89 and 1989 ACM SIGMETRICS Conf.*, pages 127–135, 1989.

[75] P.G. Harrison and N.M. Patel. The representation of switching networks in queueing models of parallel systems. In *Proc. Performance '87*, pages 497–512, 1987.

[76] H. Heffes. Moment formulae for a class of mixed multi-job-type queueing net-

works. *Bell System Technical Journal*, 61:709–745, 1982.

[77] P. Heidelberger and M.S. Lakshmi. A performance comparison of multimicro and mainframe database architectures. *IEEE Transactions on Software Engineering*, 14:522–531, 1988.

[78] P. Heidelberger and S.S. Lavenberg. Computer performance evaluation methodology. *IEEE Trans. on Computers*, C-33:1195–1220, 1984.

[79] P. Heidelberger, R.D. Nelson, and P.D. Welch. An application of interactive analysis and computer graphics in stochastic modeling. *The Annals of Operations Research*, 8:256–264, 1987.

[80] P. Heidelberger and K.S. Trivedi. Analytic queueing models for programs with internal concurrency. *IEEE Trans. on Computers*, 32:73–82, 1983.

[81] P. Heidelberger and K.S. Trivedi. Queueing network models for parallel processing with asynchronous tasks. *IEEE Trans. on Computers*, 31:1099–1108, 1982.

[82] H.U. Heiss and G. Totzauer. Optimizing utilization under response time constraints. *Computing*, 35:1–12, 1985.

[83] A. Hordijk and N. van Dijk. Networks of queues, part 1: job-local-balance and the adjoint process; part 2: general routing and service characteristics. In *Proceedings International Seminar on Modeling and Performance Evaluation Methodology*, pages 79–135, 1983.

[84] K.P. Hoyme, S.C. Bruell, P.V. Afshari, and R.Y. Kain. A tree-structured mean value analysis algorithm. *ACM Trans. on Computer Systems*, 4:178–185, 1986.

[85] M.T.T. Hsiao and A.A. Lazar. A game theoretic approach to decentralized flow control of Markovian queueing networks. In *Proc. Performance '87*, pages 55–73, 1987.

[86] M.T.T. Hsiao and A.A. Lazar. Optimal decentralized flow control of Markovian queueing networks with multiple controllers, part I: the team decision problem. In *Proceedings of the Third International Conference on Data Communication Systems and their Performance*, pages 357–372, North-Holland, 1988.

[87] C.T. Hsieh and S.S. Lam. Pam - a noniterative approximate solution method for closed multichain queueing networks. *Performance Evaluation*, 9:119–133, 1989.

[88] C.T. Hsieh and S.S. Lam. Two classes of performance bounds for closed queueing networks. *Performance Evaluation*, 7:3–30, 1987.

[89] K.B. Irani and I.H. Onyuksel. A closed form solution for the performance analysis of multiple bus multiprocessor systems. *IEEE Trans. on Computers*, 33:1004–1012, 1984.

[90] J.R. Jackson. Jobshop-like queueing systems. *Management Science*, 10:131–142, 1963.

[91] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5, 1957.

[92] P.A. Jacobson and E.D. Lazowska. Analyzing queueing networks with simultaneous resource possession. *Communications of the ACM*, 25:142–151, 1982.

[93] P.A. Jacobson and E.D. Lazowska. *The Method of Surrogate Delays: Simultaneous Resource Possession in Analytic Models of Computer Systems.* Technical Report, University of Washington,Seattle, December 1980.

[94] P.A. Jacobson and E.D. Lazowska. A reduction technique for evaluating queueing networks with serialization delays. In *Proc. Performance '83*, pages 45–59,

1983.

[95] A. Kapelnikov, R.R. Muntz, and M. Ercegovac. A methodology for the performance evaluation of concurrent systems and computations. *Journal of Parallel and Distributed Computing*, 6:568–597, 1989.

[96] F.P. Kelly. *Reversibility and Stochastic Networks*. Wiley, 1979.

[97] J.R. Kenevan and A.K. von Mayrhauser. Convexity and concavity properties of analytic queueing models for computer systems. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 361–375, 1984.

[98] T. Kerola. The composite bound method for computing throughput bounds in multiple class environments. *Performance Evaluation*, 6:1–9, 1986.

[99] M.G. Kienzle. *Measurements of Computer Systems for Queueing Network Models*. Technical Report, Univ. of Toronto, 1977.

[100] L. Kleinrock. Analysis of a timed-shared processor. *Naval Research Logistics Quarterly*, 11:59–73, 1964.

[101] L. Kleinrock. *Communication Nets*. McGraw Hill, 1964.

[102] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. Wiley-Interscience, 1976.

[103] H. Kobayashi and M. Gerla. Optimal routing in closed queueing networks. *ACM Trans. on Computer Systems*, 1:294–310, 1983.

[104] Y.A. Kogan and L.B. Boguslavsky. Asymptotic analysis of memory interference in multiprocessors with private cache memories. *Performance Evaluation*, 5:97–104, 1985.

[105] P.S. Kritzinger, S. van Wyk, and A.E. Krzesinsk. A generalization of Norton's theorem for multiclass queueing networks. *Performance Evaluation*, 2:98–107, 1982.

[106] J. Kriz. Two classes of performance bounds for closed queueing networks. *Performance Evaluation*, 4:1–10, 1984.

[107] C.P. Kruskal, M. Snir, and A. Weiss. The distribution of waiting times in clocked multistage interconnection networks. In *Proceedings of the International Conference on Parallel Processing*, pages 12–19, 1986.

[108] A. Krzesinski and J. Greyling. Improved linearizer methods for queueing networks with queue dependent centers. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 41–51, 1984.

[109] A. Krzesinski and P. Teunissen. Multiclass queueing networks with population constrained subnetworks. In *Proc. 1985 ACM SIGMETRICS Conf.*, pages 128–138, 1985.

[110] S.S. Lam. Dynamic scaling and growth behavior of queueing normalization constants. *Journal of the ACM*, 29:492–513, 1982.

[111] S.S. Lam. Store-and-forward buffer requirements in a packet switching network. *IEEE Trans. on Communications*, 24:394–403, 1976.

[112] S.S. Lam and Y.L. Lien. A tree convolution algorithm for the solution of queueing networks. *Communications of the ACM*, 26:203–215, 1983.

[113] S.S. Lam and M. Reiser. Congestion control of store-and-forward networks by input buffer limits - an analysis. *IEEE Trans. on Communications*, 27:127–134, 1979.

[114] S.S. Lam and J.W. Wong. Queueing network models of packet switching networks, part 2: networks with population size constraints. *Performance Evalu-*

ation, 2:161–180, 1982.

[115] E.R. Lassettre and A.L. Scherr. Modeling the performance of OS/360 time sharing option(TSO). In W. Freiberger, editor, *Statistical Performance Evaluation*, pages 57–72, Academic Press, 1972.

[116] S.S. Lavenberg, editor. *Computer Performance Modeling Handbook*. Academic Press, 1983.

[117] S.S. Lavenberg. A perspective on queueing models of computer performance. In *Queueing Theory and Its Applications- Liber Amicorum for J.W. Cohen*, Amsterdam, North-Holland, 1988.

[118] S.S. Lavenberg and M. Reiser. Stationary state probabilities of arrival instants for closed queueing networks with multiple types of customers. *J. App. Prob.*, 17:1048–1061, 1980.

[119] A.A. Lazar. Optimal flow control of a class of queueing networks in equilibrium. *IEEE Trans. on Automatic Control*, 28:1001–1007, 1983.

[120] E.D. Lazowska and J. Zahorjan. Multiple class memory constrained queueing networks. In *Proc. 1982 ACM SIGMETRICS Conf.*, pages 130–140, 1982.

[121] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models*. Prentice-Hall, 1984.

[122] E.D. Lazowska, J. Zahorjan, and K.C. Sevcik. Computer system performance evaluation using queueing network models. In *Annual Review of Computer Science*, pages 107–138, Annual Reviews Inc., 1986.

[123] K.J. Lee and D. Towsley. A comparison of priority-based decentralized load balancing policies. In *Proc. Performance '86 and 1986 ACM SIGMETRICS Conf.*, pages 70–77, 1986.

[124] S.T. Leutenegger and M.K. Vernon. A mean-value performance analysis of a new multiprocessor architecture. In *Proc. 1988 ACM SIGMETRICS Conf.*, pages 167–176, 1988.

[125] R.A. Marie. An approximate analytical method for general queueing networks. *IEEE Transactions on Software Engineering*, 5:530–538, 1979.

[126] J. McKenna and D. Mitra. Asymptotic expansions and integral representations of moments of queue lengths in closed Markovian networks. *Journal of the ACM*, 31:346–360, 1984.

[127] J. McKenna and D. Mitra. Asymptotic expansions and integral representations of moments of queue lengths in closed Markovian networks. *Int. Seminar on Modeling and Performance Evaluation Methodology*, 1983.

[128] J. McKenna and D. Mitra. Integral representations and asymptotic expansions for closed Markovian queueing networks: normal usage. *Bell System Technical Journal*, 61:661–683, 1982.

[129] J. McKenna, D. Mitra, and K.G. Ramakrishnan. A class of closed Markovian queueing networks: integral representations, asymptotic expansions, and generalizations. *Bell System Technical Journal*, 60:599–641, 1981.

[130] D. Mitra and J. McKenna. Asymptotic expansions for closed Markovian networks with state dependent service rates. *Journal of the ACM*, 33:568–592, 1986.

[131] C.G. Moore. *Network Models for Large-scale Time-sharing Systems*. Technical Report, Dept. of Ind. Eng., Univ. of Michigan, Ann Arbor, 1971.

[132] F.R. Moore. Computational model of a closed queueing network with exponential servers. *IBM Journal of Research and Development*, 16:567–572, 1962.

[133] R.R. Muntz. *Poisson Departure Processes and Queueing Networks.* Technical Report RC 4145, IBM Thomas J. Watson research Center, Yorktown Heights, N.Y., 1972.

[134] R.R. Muntz and J.W. Wong. Asymptotic properties of closed queueing network models. In *Proceedings of the Eighth Annual Princeton Conference on Information Systems*, 1974.

[135] D. Neuse and K.M. Chandy. HAM: the heuristic aggregation method for solving general closed queueing network models of computer systems. *Proc. 1982 ACM SIGMETRICS Conf.*, 195–212, 1982.

[136] L.M. Ni and K. Hwang. Optimal load balancing in a multiple processor system with many job classes. *IEEE Transactions on Software Engineering*, 11:491–496, 1985.

[137] A. Norton and G.P. Pfister. A methodology for predicting multiprocessor performance. In *Proceedings of the International Conference on Parallel Processing*, pages 772–780, 1985.

[138] N.M. Patel and P.G. Harrison. On hot-spot contention in interconnection networks. In *Proc. 1988 ACM SIGMETRICS Conf.*, pages 114–122, 1988.

[139] K.R. Pattipati, M.M. Kostreva, and J.L. Teele. On the properties of approximate mean value analysis algorithms for queueing networks. In *Proc. 1988 ACM SIGMETRICS Conf.*, pages 244–252, 1988.

[140] M.C. Pennoti and M. Schwartz. Congestion control in store and forward tandem links. *IEEE Trans. on Communications*, 23:1434–1443, 1975.

[141] E. Pinsky. *Canonical Approximation in the Performance Analysis of Distributed Systems.* PhD thesis, Computer Science Department, Columbia University, New York, 1986.

[142] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel. Locus: a network transparent, high reliability distributed system. In *Proceedings of the Eighth Symposium on Operating Systems Principles*, pages 169–177, California, December 1981.

[143] T.G. Price. A comparison of queueing network models and mesurements of a multiprogrammed computer system. In *Proc. Performance '76*, pages 39–62, 1976.

[144] K.G. Ramakrishnan and D. Mitra. An overview of PANACEA, a software package for analyzing Markovian queueing networks. *Bell System Technical Journal*, 61:2849–2871, 1982.

[145] D.A. Reed. Queueing network models of multimicrocomputer networks. In *Proc. 1983 ACM SIGMETRICS Conf.*, pages 190–197, 1983.

[146] M. Reiser. *Communication-System Models Embedded in the OSI-Reference Model, a Survey.* Technical Report, IBM Zurich Research Laboratory, October 1985.

[147] M. Reiser. A queueing network analysis of computer communication networks with window flow control. *IEEE Trans. on Communications*, 27:1199–1209, 1979.

[148] M. Reiser and H. Kobayashi. Queueing networks with multiple closed chains: theory and computational algorithms. *IBM Journal of Research and Develop-*

ment, 19:283–294, 1975.

[149] M. Reiser and S.S. Lavenberg. Mean value analysis of closed multichain queueing networks. *Journal of the ACM*, 27:313–322, 1980.

[150] I. Rubin. Communication networks: message path delays. *IEEE Transactions on Information Theory*, 20:738–745, 1974.

[151] C.H. Sauer. Approximate solution of queueing networks with simultaneous resource possession. *IBM Journal of Research and Development*, 25:894–903, 1981.

[152] C.H. Sauer and K.M. Chandy. *Computer System Performance Modeling*. Prentice Hall, 1981.

[153] C.H. Sauer, E.A. MacNair, and J.F. Kurose. The research queueing package: past, present and future. In *Proc. 1982 National Computer Conference*, pages 273–280, 1982.

[154] C.M. Sauer, E.A. MacNair, and J.F. Kurose. Queueing network simulations of computer communication. *IEEE Journal on Selected Areas in Communications*, SAC-2:203–220, 1984.

[155] R. Schassberger. The insensitivity of stationary probabilities in networks of queues. *J. App. Prob.*, 10:906–912, 1978.

[156] R. Schassberger. Insensitivity of steady-state distributions of generalized semi-Markov processes. Part II. *Ann. Prob.*, 10:85–93, 1978.

[157] R. Schassberger. Insensitivity of steady-state distributions of generalized semi-Markov processes with speeds. *Adv. App. Prob.*, 10:836–851, 1978.

[158] A.L. Scherr. *An Analysis of Time-Shared Computer Systems*. MIT Press, 1967.

[159] M. Schwartz. *Telecommunication Networks: Protocols, Modeling and Analysis*. Addison-Wesley, 1987.

[160] P. Schweitzer. Approximate analysis of multiclass closed networks of queues. In *International Conference on Stochastic Control and Optimization*, Amsterdam, 1979.

[161] P.J. Schweitzer and S.S. Lam. Buffer overflow in a stored-and-forward network node. *IBM Journal of Research and Development*, 20:542–550, 1976.

[162] A. Sekino. *Performance Evaluation of Multiprogrammed Time-shared Computer Systems*. PhD thesis, MIT, Cambridge, MA, 1972.

[163] K. Sevcik and I. Mitrani. The distribution of queueing network states at input and output instants. *Journal of the ACM*, 28:358–371, 1981.

[164] K.C. Sevcik. Priority scheduling disciplines in queueing network models of computer systems. In H. Gilchrist, editor, *Information Processing 77*, pages 565–570, North-Holland, 1977.

[165] H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[166] C. Smith and J.C. Browne. Aspects of software design analysis: concurrency and blocking. In *Proc. Performance '80*, pages 245–253, 1980.

[167] K.E. Stecke. On the concavity of throughput in certain closed queueing networks. *Performance Evaluation*, 6:293–305, 1986.

[168] L.E. Stephens and L.W. Dowdy. Convolutional bound hierarchies. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 120–133, 1984.

[169] J. Strelen. A generalization of mean value analysis for higher moments: moment analysis. In *Proc. Performance '86 and 1986 ACM SIGMETRICS Conf.*,

pages 129–140, 1986.

[170] R. Suri. Generalized quick bounds for performance of queueing networks. *Computer Performance*, 5:116–120, 1984.

[171] R. Suri. Robustness of queueing network formulas. *Journal of the ACM*, 30:564–594, 1983.

[172] A.N. Tantawi and D. Towsley. Optimal load balancing in distributed computer systems. *Journal of the ACM*, 32:442–462, 1985.

[173] Y.C. Tay, R. Suri, and N. Goodman. A mean value performance model for locking in databases: the no-waiting case. *Journal of the ACM*, 32:618–651, 1985.

[174] A. Thomasian. Queueing network models to estimate serialization delays in computer systems. In *Proc. Performance '83*, pages 61–81, 1983.

[175] A. Thomasian and P. Bay. Analysis of queueing network models with population size constraints and delayed block customers. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 202–216, 1984.

[176] A. Thomasian and P. Bay. Analytic queueing network models for parallel processing of task systems. *IEEE Trans. on Computers*, C-35:1045–1054, 1986.

[177] A. Thomasian and P. Bay. Performance analysis of task systems using a queueing network model. *Int'l Workshop on Timed Petri Nets*, 1985.

[178] D. Towsley, K.M. Chandy, and J.C. Browne. Models for parallel processing within programs: application to CPU:I/O and I/O:I/O overlap. *Communications of the ACM*, 21:821–831, 1978.

[179] D.F. Towsley. Queueing network models with state dependent routing. *Journal of the ACM*, 27:323–337, 1980.

[180] S.K. Tripathi and L.W. Dowdy. Workload representation and its impact on the performance prediction using queueing network models. In G. Serazzi, editor, *Workload Characterization of Computer Systems and Computer Networks*, pages 159–178, North-Holland, 1986.

[181] S.K. Tripathi, Y. Huang, and D. Towsley. On optimal file allocation with sharing. In *Proceedings of the International Seminar on Performance of Distributed and Parallel Systems*, pages 1–14, Kyoto, Japan, 1988.

[182] S.K. Tripathi and C.M. Woodside. A vertex-allocation theorem for resources in queueing networks. *Journal of the ACM*, 35:221–230, 1988.

[183] K.S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications*. Prentice Hall, 1982.

[184] K.S. Trivedi and R.E. Kinicki. A model for computer configuration design. *Computer*, 13:47–54, 1980.

[185] K.S. Trivedi and R.A. Wagner. A decision model for closed queueing networks. *IEEE Transactions on Software Engineering*, 5:328–332, 1979.

[186] K.S. Trivedi, R.A. Wagner, and T.M. Sigmon. Optimal selection of CPU speed, device capacities, and file assignments. *Journal of the ACM*, 27:457–473, 1980.

[187] S. Tucci and E.A. MacNair. Implementation of mean value analysis for open, closed and mixed queueing networks. *Performance Theory*, 3:233–239, 1982.

[188] S. Tucci and C.H. Sauer. The tree MVA algorithm. *Performance Evaluation*, 5:187–196, 1985.

[189] N. van Dijk. A simple bounding methodology for non-product form finite capacity queueing systems. In *Proceedings of the First International Workshop*

on Queueing Networks with Blocking, pages 1–23, 1988.

[190] H. Vantilborgh. Aggregation with an error of $o(\epsilon^2)$. *Journal of the ACM*, 32:162–190, 1985.

[191] H. Vantilborgh. Exact aggregation in exponential queueing networks. *Journal of the ACM*, 25:620–629, 1978.

[192] M. Veran and D. Potier. QNAP2: a portable environment for queueing systems modeling. *Intl. Conf. Modeling Technique and Tools for Perf. Eval*, 1984.

[193] M. Vernon, Zahorjan, and E.D. Lazowska. A comparison of performance Petri nets and queueing network models. *Int'l. Workshop on Modeling Techniques and Performance Evaluation*, 181–192, March 1987.

[194] B.W. Wah. File placement on distributed computer systems. *Computer*, 23–32, 1984.

[195] J. Walrand. *An Introduction to Queueing Networks*. Prentice Hall, 1988.

[196] Y.T. Wang and R.J.T Morris. Load sharing in distributed systems. *IEEE Trans. on Computers*, 34:204–217, 1985.

[197] P. Whittle. Partial balance and insensitivity. *J. App. Prob.*, 22:168–176, 1985.

[198] P. Whittle. Partial balance, insensitivity and weak coupling. *Adv. App. Prob.*, 18:706–723, 1986.

[199] A.C. Williams and R.A. Bhandiwad. A generating function approach to queueing network analysis of multiprogrammed computers. *Networks*, 6:1–22, 1976.

[200] J. Wolf. The placement optimization program: a practical solution to the disk file assignment problem. In *Proc. Performance '89 and 1989 ACM SIGMETRICS Conf.*, pages 1–10, 1989.

[201] J.W. Wong and S.S. Lam. Queueing network models of packet switching networks, part 1: open networks. *Performance Evaluation*, 2:9–21, 1982.

[202] C.M. Woodside and S.K. Tripathi. Optimal allocation of file servers in a local network environment. *IEEE Transactions on Software Engineering*, 12:844–848, 1986.

[203] T.P. Yum and H.-C. Lin. Adaptive load balancing for parallel queues. In *Proceedings of the IEEE International Conference on Communications*, Amsterdam, 1984.

[204] J. Zahorjan. Workload representations in queueing models of computer systems. In *Proc. 1983 ACM SIGMETRICS Conf.*, pages 70–81, 1983.

[205] J. Zahorjan. Workload representations in queueing models of computer systems. In *Proc. 1983 ACM SIGMETRICS Conf.*, pages 70–81, 1983.

[206] J. Zahorjan, D.E. Eager, and H.M. Sweillam. Accuracy, speed, and convergence of approximate mean value analysis. *Performance Evaluation*, 8:255–270, 1988.

[207] J. Zahorjan and E.D. Lazowska. Incorporating load dependent servers in approximate mean value analysis. In *Proc. Performance '84 and 1984 ACM SIGMETRICS Conf.*, pages 52–62, 1984.

[208] J. Zahorjan, K.C. Sevcik, D.L. Eager, and B. Galler. Balanced job bound analysis of queueing networks. *Communications of the ACM*, 25:134–141, 1982.

[209] J. Zahorjan and E. Wong. The solution of separable queueing network models using mean value analysis. In *Proc. 1981 ACM SIGMETRICS Conf.*, pages 80–85, 1981.