Product Form Approximations for Queueing Networks with Multiple Servers and Blocking

I. F. AKYILDIZ, MEMBER, IEEE

Abstract—**Queueing networks with blocking have become an important research topic in performance evaluation in recent years. Various types of blocking within queueing networks have been studied by several investigators. In this work, the following type of blocking is examined. Upon completing of its service, a job attempts to enter a new station. If at that moment, the destination station is full, the job is forced to reside in the server of the source station until a place becomes available in the destination station. The server of the source station remains blocked during this period of time. It is shown that the equilibrium state probabilities for this type of blocking queueing network have an approximate product form solution. The solution is based on the concept of normalizing the infeasible states that violate station capacities. The states are adapted to the allowed capacities of the stations. However, the equilibrium state probabilities allow only the computation of mean number of jobs. In order to obtain the throughput values, the concept of the state space transformation is introduced. This concept is based on finding a nonblocking network with appropriate total number of jobs of which the number of feasible states is equal or approximately equal to the number of feasible states in the blocking queueing network. This guarantees that the Markov processes describing the evolution of both networks over time have approximately the same structure. This leads to the result that the throughputs of both systems are approximately equal. The approximations are validated by executing several examples, including stress tests, and comparing them with simulation results.**

Index Terms—**Binomial coefficient formula, blocking, finite station capacities, performance evaluation, performance measures, queueing networks.**

I. INTRODUCTION

A COMPUTER system can be viewed as an organized collection of hardware and software resources for which the concurrent processes in the system are constantly competing. Two major functions of an operating system are the effective scheduling of conflicting requests and the appropriate handling of the process queues waiting for resource allocation and scheduling. Queueing network theory can be used to provide the basic framework and mathematical tools for the modeling and analysis of computer system hardware and software.

Manuscript received June 6, 1986; revised October 29, 1986. This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant AFOSR-87-0160.

The author is with the School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332.

IEEE Log Number 8820886.

Queueing networks have received a very special interest in the last 15 years for uses with performance evaluation and performance prediction. A queueing network is a collection of stations (devices with queues) in which jobs/processes proceed from one station to another in order to satisfy their service requirements. When dealing with queueing networks with infinite station capacities, Baskett *et al.* [10] and Kelly [21] have shown that the solutions for four types of stations (•/M/M/m-FCFS, •/G/1-RR-PS, •/G/IS (Infinite Servers), and •/G/1-LCFS-PR) have a product form. The product form implies that each station in the network can be analyzed independently from each other. Several algorithms have been introduced for an efficient computation of performance measures for queueing networks in the last 15 years [27], [33], [34], [35]. In the last two decades, many successful applications have been developed for the modeling of computer, communication, and manufacturing systems by queueing networks [27], [35], [40].

Since in actual systems the resources have finite capacity, queueing networks with blocking must be used for performance analysis. In queueing networks with blocking, a station can be thought of as a device with a finite length buffer. Blocking arises because of the limitations imposed by the capacity of these buffers.

In recent years, there has been a growing interest in the development of computational methods to analyze queueing networks with blocking. Researchers from various areas such as computer science, operations research, mathematics, and electrical engineering have studied blocking networks. Several papers have been published dealing with various types of blocking types. Previous work regarding the blocking networks falls into three classes [28].

1) *Transfer (Type 1 = Manufacturing) Blocking:* Upon completion of its service, a job at station i attempts to enter destination station j. If station j is full at that moment, the job is forced to wait at server i until it enters destination station j. The server remains blocked for this period of time. It cannot serve any other job waiting in the queue. This type of blocking has been used to study models of flexible manufacturing systems and disk I/O subsystems [1]–[5], [8], [12], [14], [29]–[31], [36], [38], [39], [41].

2) *Service (Type 2 = Communication_1) Blocking:* A job in station i declares its destination station j before it starts its service. If station j is full, the ith server is blocked, i.e., it cannot serve jobs. When a departure occurs from destination

0018-9340/89/0100-0099$01.00

station j, the ith server becomes unblocked and the job begins receiving service. This blocking type has been used to study models of production lines and telecommunication networks [11], [14], [16], [17], [22], [23], [25], [37].

3) *Rejection (Type 3 = Communication_2) Blocking:* Upon service completion at station i, a job attempts to join destination station j. If station j is full at that moment, the job is refused. The rejected job goes with a certain probability (so-called "rejection probability") back to the station i's server and immediately receives a new service with the same mean service time. This is repeated until some job completes a service at station j and a place becomes available. This blocking type has been used to model communication networks [6], [7], [9], [13], [15], [18], [19], [21], [26], [32], [42].

Comparisons between these distinct types of blocking have been carried out by Onvural and Perros [28]. Several other investigators in recent years have published results on blocking queueing networks. Since we investigate closed queueing networks with transfer blocking here, the discussion of the previous work will contain the studies of these networks.

The analysis of transfer blocking networks is very difficult. Exact results exist only for very limited cases. In [1], we showed that the equilibrium state probability distributions of two-station closed queueing networks with transfer blocking and multiple server stations are identical to those of two-station closed queueing networks without blocking. Onvural and Perros [29] show that if the number of jobs in a network with transfer blocking is one more the capacity of the station with the smallest capacity, there exists an exact product form solution.

All other studies for this type of networks are approximations. In [3], we showed that the throughput of a blocking network with K total number of jobs and single servers is approximately equal to the throughput of a nonblocking network with an appropriate total number of jobs $\hat{K}$. The approximation provides very accurate results, on the average 5 percent deviations. In [4], we modify mean value analysis, in short form MVA [34], for transfer blocking networks. The resulting method is called MVABLO. MVABLO estimates the additional time a job spends in each station because of blocking. This time is included in the response time formula of MVA. Even though the method is extremely fast, it provides larger deviations from actual values. These results have been extended to networks with general service time distributions in [2].

Suri and Diehl [38] consider transfer blocking policy in cyclic networks. They present an approximate method to compute the throughput of the network. They approximate groups of two stations by a variable capacity station, defined as a superposition of fixed capacity stations. They start with the last two stations and successively reduce the network until two stations in tandem remain. The method is easy to implement and shows good accuracy but involves much computation. At each step, all conditional probabilities have to be found, since they are used to construct the equivalent variable capacity station. The method only gives the throughput of the entire network, it does not give statistics for individual stations. Another disadvantage is that it is applicable only on cyclic networks where one of the stations must have an infinite capacity. Another drawback is that the capacity of each downstream station must be smaller than the total number of jobs in the network.

Perros, Nilsson, and Liu [30] give an algorithm for an arbitrarily connected network where some (very few) stations may have finite capacity. They partition the set of stations in a so-called blocking subnetwork and a nonblocking subnetwork. The nonblocking subnetwork containing infinite capacity stations is replaced by a composite station using parametric analysis for infinite capacity networks. The reduced network is then analyzed numerically. However, if all stations of the network have finite capacity this method reduces itself to a numerical analysis method which, as generally known, is only applicable on very small networks.

Deadlocks are possible in transfer blocking networks. All stations in a directed cycle could be full at one time. If in each of the stations of the cycle, the blocked job is scheduled to go to the next station in the cycle, the network is deadlocked. There are two possible solutions to the deadlock problem in blocking networks.

1) Include a strategy to handle deadlocks in the model. Perros, Nilsson, and Liu [30] assume that in case of deadlock all jobs involved move simultaneously to their destinations. This complicates the model, since the deadlock handling method influences the balance equations.

2) Simply restrict yourself to cases where deadlock is impossible. One such case arises whenever the number of jobs in the system is less than the capacity of the directed cycle with minimal capacity. No directed cycle can ever have all its stations full at the same time, and deadlock is impossible.

This paper is organized as follows: In Section II, we show that the equilibrium state probabilities for multiple server queueing networks with transfer blocking can be computed approximately. The solution is based on the concept of normalizing the infeasible states that violate station capacities. The states are adapted to the allowed capacities of the stations. However, the equilibrium state probabilities allow only the computation of the mean number of jobs. In order to obtain the throughput values, the concept of a state space transformation is introduced in Section III. In Section IV, two examples are given to explain both algorithms in the case of tandem and nontandem networks. Additional examples are given to compare our results to the results of Suri and Diehl in the evaluation Section V. In the Appendix, several examples are listed and the results are compared to simulation.

II. Approximate Product Form Solution

We consider closed queueing networks with the following assumptions.

1) There are N stations.

2) The number of jobs in the system is fixed at K. All jobs belong to the same class.

3) Each station may have multiple servers ($m_i \geq 1$) and an exponentially distributed service time with mean value $1/\mu_i$ ($i = 1, 2, \cdots, N$).

4) Each station has a fixed finite capacity M_i ($i = 1, 2, \cdots,$

N). (M_i = queue capacity + m_i). Cases in which $M_i = \infty$ are also allowed.

5) A job serviced by the ith station proceeds to the jth station with probability p_{ij} for $i, j = 1, 2, \cdots, N$, if the number of jobs in the jth station has not exceeded the capacity M_j. Otherwise, the job is blocked in the ith station until a job in the jth station is serviced and a place becomes available.

6) The total number of jobs K must be smaller than the sum of the station capacities, that is,

$$K < \sum_{i=1}^{N} M_i. \tag{1}$$

7) The service discipline in each station is first-come-first-served.

8) The network must be deadlock free. Deadlock occurs in blocking networks, if, for instance, assume that station i is blocked by station j. Now it is possible that a job in station j may, upon completion of its service, choose to go to station i. If station i is full at that time, the deadlock will occur. The blocking network is deadlock free, if

$$K < \sum_{j \in C} M_j, \tag{2}$$

i.e., the total number of jobs in the network must be smaller than the sum of station capacities in each cycle C. Since tandem networks have only one cycle, this condition, (2), corresponds to (1). Equation (1) is a sufficient condition for tandem networks to be deadlock free.

9) If there are several stations all linked to station j, then it is possible that at any time there might be blocked jobs from more than one station waiting to enter station j. The maximum number of blocked jobs will be equal to the number of station's servers which are directly connected to station j. We assume that these blocked jobs enter station j on a first-blocked-first-enter basis.

With these assumptions, we obtain the queueing network model which is classified as transfer blocking case.

Let $\boldsymbol{k}^*$ denote the state of the network without considering the station capacities.

$$\boldsymbol{k}^* = (k_1^*, k_2^*, \cdots, k_N^*)$$

where k_i^* is the number of jobs in the ith station.

There must be at least ($K - m_i$) waiting places in each queue (m_i jobs can be in service) to ensure that all states are feasible. Since the capacity of the stations is finite, it is clear that all states $\boldsymbol{k}^*$ cannot be feasible. The feasible states for blocking networks are obtained by realizing that the number k_i^* of jobs in the ith station may not exceed the station capacity M_i, ($k_i^* \le M_i$). Blocking events which occur in networks with finite station capacities have the effect of rendering states exceeding the station capacities infeasible.

The basic idea is that the infeasible states which violate station capacities are normalized. The states are adapted to the allowed capacities of the stations. The normalization of the states exceeding station capacities is executed by using the following formula. This indicates that if the number of jobs in a state of a station exceeds its capacity, $k_i^* > M_i$, then the job distribution is normalized until no violation of the capacity limits exists:

$$k_j^* := \begin{cases} M_j & \text{if } j = i \\ k_j^* + (k_i^* - M_i)\dfrac{e_j p_{ji}}{e_i(1-p_{ii})} & \text{if } j \neq i \end{cases}$$

$$\text{for all } j = 1, 2, \cdots, N. \tag{3}$$

where e_i is the mean number of visits that a job makes to station i and is computed by

$$e_i = \sum_{j=1}^{N} e_j p_{ji} \qquad \text{for } i = 1, \cdots, N. \tag{4}$$

The informal interpretation of the formula (3) is: if the capacity of a station is exceeded in a state, the number of jobs in that station is set equal to its capacity M_j (first term) and distribute the remaining number of jobs in the predecessor stations according to the transition probabilities (second term).

Note that in case of networks with arbitrarily connected stations, the transition probabilities p_{ij} in the normalization procedure, (3), cause the number of jobs k_i in a state to have noninteger values. All nonfeasible states $\boldsymbol{k}^*$ of the queueing network are normalized by (3). The job distribution in each state is adapted to the capacity limit of each station in the blocking network:

$$f(\boldsymbol{k}^*) = (\boldsymbol{k}) \tag{5}$$

where $\boldsymbol{k}$ is the normalized state for the blocking network. The function f transforms the nonfeasible state ($\boldsymbol{k}^*$) to the feasible state ($\boldsymbol{k}$).

By the normalization procedure, an equivalent state space structure is obtained for the blocking network. The only difference between the two state space structures is that the jobs are distributed according to the station capacities in the state space of the blocking network.

For a closed queueing network model with transfer blocking, the *equilibrium probability distribution* for the feasible states is computed by

$$p(\boldsymbol{k}) = \sum_{\boldsymbol{k}^* \text{ where } f(\boldsymbol{k}^*) = \boldsymbol{k}} p^*(\boldsymbol{k}^*). \tag{6}$$

$p^*(\boldsymbol{k}^*)$ represents the equilibrium state probability distribution of the network without station capacity restrictions computed by the well-known Gordon/Newell theorem [10], [35], [40].

$$p^*(k_1^*, k_2^*, \cdots, k_N^*) = \frac{1}{G(K)} \prod_{i=1}^{N} \left[\frac{e_i}{\mu_i \beta_i(k_i^*)} \right]^{k_i^*}. \tag{7}$$

$G(K)$ represents the *normalization constant* which can be calculated by any product form network algorithm, e.g., the convolution algorithm [33], [35] or LBANC technique [35].

The function $\beta_i(k_i^*)$ is defined by

$$\beta_i(k_i^*) = \begin{cases} k_i^*! & \text{if } k_i^* \le m_i \\ m_i! m_i^{k_i^* - m_i} & \text{if } k_i^* > m_i \end{cases}. \tag{8}$$

Performance measures such as the mean number of jobs $\bar{k}_i$ at the ith station are computed by

$$\bar{k}_i(K) = \sum_{\min n \text{ in station } i}^{M_i} np_i(n). \tag{9}$$

The *marginal probability of n* jobs in the ith station, $p_i(n)$, is computed from the equilibrium state probabilities $p(\boldsymbol{k})$, (7):

$$p_i(n) = \sum_{\sum_{i=1}^{N} k_i = K \& n = k_i} p(\boldsymbol{k}). \tag{10}$$

As mentioned before, the normalized state for blocking networks with arbitrarily connected stations may be noninteger values for the number of jobs. Thus, for networks with arbitrarily connected stations, the mean number of jobs is computed by

$$\bar{k}_i(K) = \sum_{\sum_{j=1}^{N} k_j = K \& \boldsymbol{k}} f_i(\boldsymbol{k}^*)p(\boldsymbol{k}) \qquad \text{for } i = 1, \cdots, N \tag{11}$$

where the function $f_i(\boldsymbol{k}^*)$ is the ith component of the function $f(\boldsymbol{k}^*)$.

Even though all performance measures can be computed from the equilibrium state probabilities using the according formulas, we discovered this holds only for queueing networks with infinite station capacities. For example, in order to obtain the throughput in blocking queueing networks, we could not use the following well-known throughput formula [35]:

$$\lambda_i(K) = \sum_{n=1}^{K} \mu_i(n)p_i(n) \tag{12}$$

where the marginal probabilities are obtained by (10) and the load dependent service rates are given by

$$\mu_i(n) = \begin{cases} n\mu_i & \text{if } n \le m_i \\ m_i\mu_i & \text{if } n > m_i \end{cases}. \tag{13}$$

We were not able to obtain other performance measures by the equilibrium state probabilities. Hence, we looked for other ways and found the following concept for the computation of the throughput values in blocking queueing networks, as was also shown in [3] for single server cases.

III. Throughput Analysis in Blocking Queueing Networks

In [1], we have shown that the state space of two-station networks with blocking is isomorph to the state space of two-station nonblocking networks with appropriate total number of jobs. Based on this concept, we obtained exact results for two-station networks with blocking. In the case of networks with more than two stations, we have realized that the state space of the blocking network cannot be transformed bijectively into the state space of the nonblocking network. However, the number of states in both networks may be equal or approximately equal. This would imply that both networks have approximately the same stochastic structure and the throughputs of both networks are approximately equal.

The following steps are executed in order to compute the throughput values in queueing networks with blocking.

1) Determine the number of states in the blocking queueing network.

2) Determine the total number of jobs $\hat{K}$ in the nonblocking queueing network.

3) Analyze the nonblocking queueing network with $\hat{K}$ jobs using MVA [34] or LBANC technique [35] and obtain the throughput value which is approximately equal to the throughput value of the blocking network with K jobs.

A. Determine the Number of States in Blocking Queueing Networks

As mentioned before, in blocking networks, each station has a capacity limit, which indicates that a certain number of states can be feasible. The feasible states for blocking networks are obtained by realizing that the number k_i of jobs in the ith station may not exceed its capacity M_i, $k_i \le M_i$. Blocking events which occur in networks with finite station capacities must also be taken into account. Therefore, the m_i neighbors of the feasible states are included, representing the blocking states for the stations. As many servers a station possesses as many neighbors of the feasible states are included to the state space. Whenever a transition occurs from one state to another state in which the capacity limit of a station would be violated, we assume that the transition causes a blocking action in the network and that the state entered is a blocking state. In reality, the job still resides in the source station. All the other states are infeasible and are cancelled.

In this way, we obtain a substate space for the blocking network, with $Z'(K)$ number of feasible and blocking states. For the computation of $Z'(K)$ we have developed an efficient convolution algorithm [3] which is extended here to multiple server cases.

The number of the feasible and blocking states is obtained from the last element $Z'(K)$ of the following vector result $\boldsymbol{Z}'$.

$$\boldsymbol{Z}' = Z_1 \otimes Z_2 \otimes \cdots \otimes Z_N \tag{14}$$

where $\otimes$ is a convolution operation which is explained as follows.

Let A, B, and C be three $(N + 1)$-dimensional vectors. The convolution operation $\otimes$ gives the following result:

$$C = A \otimes B$$

with

$$c(n) = \sum_{l=0}^{i} a(l) \cdot b(i-l) \qquad \text{for } i = 0, 1, \cdots, K.$$

Z_i for $i = 1, 2, \cdots, N$ is a $(K + 1)$-dimensional vector given by

$$Z_i = \begin{bmatrix} z_i(0) \\ z_i(1) \\ \cdot\cdot \\ \cdot\cdot \\ z_i(K) \end{bmatrix}$$

with the binary function as elements

$$z_i(k)=\begin{cases}1 & \text{if } k=0,\ 1,\ 2,\ \cdots,\ \left\{M_i+\sum_{j=1 \text{ and } p_{ji}>0}^{N} m_j\right\} \\ 0 & \text{if otherwise.}\end{cases}$$

Note that if the stations have no capacity limitations, then $Z'(K)$ is equal to Z, the known binomial coefficient formula:

$$Z=\binom{N+K-1}{N-1}. \tag{15}$$

This coefficient indicates the number of possible ways to distribute K jobs into N stations and provides the number of states Z in a closed queueing network without station capacity restrictions.

B. Determine the Total Number of Jobs $\hat{K}$ in the Nonblocking Network

We have shown that the number of states $Z'(K)$ (feasible and blocking states) in the blocking network is obtained by (14). The goal is to find a total number of jobs $\hat{K}$ in a network with infinite station capacities having $\hat{Z}$ states such that $\hat{Z}$ will be approximately equal to $Z'(K)$.

The total number of jobs $\hat{K}$ in the nonblocking queueing network is determined by the following binomial coefficient formula:

$$\hat{Z}=\binom{N+\hat{K}-1}{N-1}. \tag{16}$$

If we want to make $\hat{Z}=Z'(K)$, it is possible that noninteger values for $\hat{K}$ will result from (16). Since the number of jobs in networks is an integer value, we assume $\hat{K}$ as the integer value which provides the nearest value to the number of states as $Z'(K)$ computed by (14). Note that if a value for $Z'(K)$ is obtained by (14) which is in the middle of two computed $\hat{Z}$ values, we choose that value for total number of jobs $\hat{K}$ which provides the higher $\hat{Z}$ value.

C. Determine the Throughput of the Nonblocking Network

By analyzing the nonblocking network with $\hat{K}$ total number of jobs using a product form algorithm such as mean value analysis [34], we obtain the total throughput $\lambda_{NB}(\hat{K})$. As mentioned before, since the number of states in both networks is approximately equal, it implies that the Markov processes describing the evolution of both networks have approximately the same structure. Consequently, the throughput of the nonblocking network $\lambda_{NB}(\hat{K})$ is approximately equal to the throughput of the blocking network $\lambda_B(K)$.

$$\lambda_B(K)\approx\lambda_{NB}(\hat{K}). \tag{17}$$

The *throughput* λ_i and the *utilization* ρ_i of each station (for $i=1,\ \cdots,\ N$) are computed by

$$\lambda_i(K)=e_i\lambda(K) \qquad \rho_i(K)=\frac{\lambda_i(K)}{m_i\mu_i}. \tag{18}$$

The *total response time* $\bar{t}_C$ (= cycle time = turnaround time) of jobs in the network is obtained by

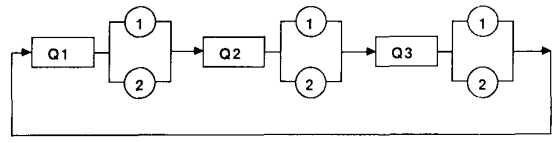


Fig. 1. A cyclic blocking network.

$$\bar{t}_C(K)=\frac{K}{\lambda_B(K)}. \tag{19}$$

The *mean residence time* $\bar{t}_i$ is determined using Little's law

$$\bar{t}_i(K)=\frac{\bar{k}_i(K)}{\lambda_i(K)}. \tag{20}$$

In the following numerical examples, we outline the general flow of both techniques.

IV. Numerical Examples

A. Cyclic Network

We examine a queueing network with $N=3$ serially switched stations and $K=6$ jobs shown in Fig. 1. The stations have finite capacities as $M_1=3$, $M_2=2$, $M_3=2$ and each has $m_i=2$ servers (for $i=1,\ 2,\ 3$). The service times are exponentially distributed with mean values $1/\mu_1=1$ s, $1/\mu_2=1.5$ s, and $1/\mu_3=2$ s.

The state space for this network has the structure shown in Fig. 2.

In Fig. 2, $a=2\mu_1$, $b=2\mu_2$, $c=2\mu_3$, $a_1=\mu_1$, $b_1=\mu_2$, and $c_1=\mu_3$ denote the transition rates bewteen the states.

As can be seen from the state space, Fig. 2, only the states (3, 2, 1), (3, 1, 2), and (2, 2, 2) are feasible. All other states exceed the capacity limits of the stations and hence they must be normalized. For example, the nonfeasible state (6, 0, 0) is normalized as follows: the capacity of the first station is violated so we set the number of jobs equal to its capacity.

$$k_1=3.$$

The remaining three jobs are distributed to other stations:

$$k_2=k_2+(k_1-M_1)p_{21}=0$$

$$k_3=k_3+(k_1-M_1)p_{31}=3.$$

By normalization we arrived at the state (3, 0, 3) where the capacity of the third station is violated. So we set the number of jobs equal to the capacity of the third station.

$$k_3=2.$$

The remaining one job is put to the predecessor station 2:

$$k_2=k_2+(k_3-M_3)p_{23}=1$$

$$k_1=k_1+(k_3-M_3)p_{13}=3.$$

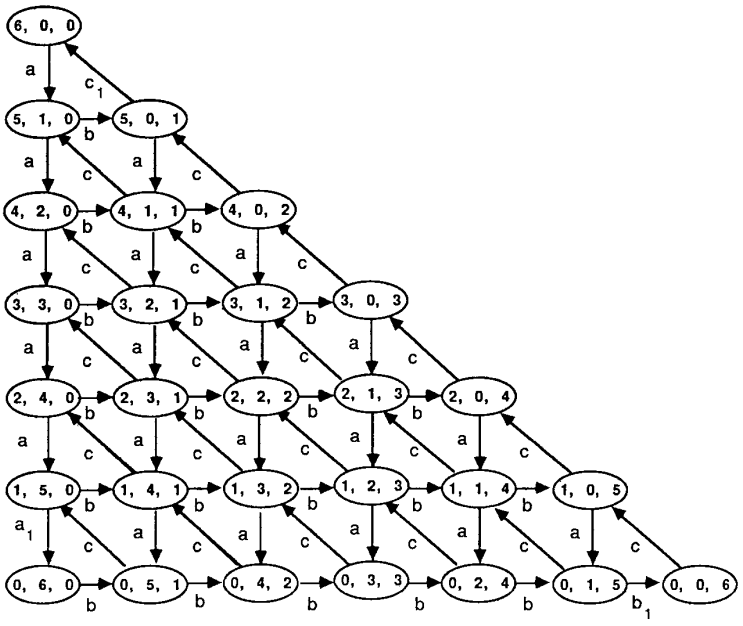


Fig. 2. State space by ignoring station capacities.

As a result, the state (6, 0, 0) becomes assigned the normalized job distribution (3, 1, 2); see (3):

$$f(6, 0, 0) = (3, 1, 2).$$

In another nonfeasible state (0, 3, 3), the capacity of the second and third station is violated. The number of jobs is set equal to the, for example, third station's capacity.

$$k_3 = 2.$$

The remaining job is put to the second station,

$$k_1 = k_1 + (k_3 - M_3)p_{13} = 0$$

$$k_2 = k_2 + (k_3 - M_3)p_{23} = 4$$

which causes the second station to violate its capacity.

A further normalization step provides that the state (0, 3, 3) becomes assigned the normalized job distribution (2, 2, 2); see (3):

$$f(0, 3, 3) = (2, 2, 2).$$

After normalization of all nonfeasible states, we obtain the state space in Fig. 3.

First ignore the station capacities and compute the equilibrium state probabilities by (7):

$$\begin{array}{llll} p^*(6,0,0)=0.001 & p^*(5,0,1)=0.004 & p^*(3,1,2)=0.025 & p^*(0,3,3)=0.056 \\ p^*(5,1,0)=0.003 & p^*(4,1,1)=0.012 & p^*(2,2,2)=0.037 & p^*(2,0,4)=0.033 \\ p^*(4,2,0)=0.004 & p^*(3,2,1)=0.018 & p^*(1,3,2)=0.056 & p^*(1,1,4)=0.099 \\ p^*(3,3,0)=0.007 & p^*(2,3,1)=0.028 & p^*(0,4,2)=0.042 & p^*(0,2,4)=0.075 \\ p^*(2,4,0)=0.01 & p^*(1,4,1)=0.042 & p^*(3,0,3)=0.016 & p^*(1,0,5)=0.066 \\ p^*(1,5,0)=0.007 & p^*(0,5,1)=0.031 & p^*(2,1,3)=0.05 & p^*(0,1,5)=0.099 \\ p^*(0,6,0)=0.01 & p^*(4,0,2)=0.0083 & p^*(1,2,3)=0.075 & p^*(0,0,6)=0.066. \end{array}$$

The equilibrium state probabilities for the feasible states of the blocking network are computed by (6):

$$\begin{aligned} p(3,1,2) &= p^*(6,0,0)+p^*(5,1,0)+p^*(5,0,1)+p^*(4,1,1) \\ &\quad +p^*(4,0,2)+p^*(3,1,2)+p^*(3,0,3)=0.0704 \\ p(3,2,1) &= p^*(4,2,0)+p^*(3,3,0)+p^*(2,4,0) \\ &\quad +p^*(1,5,0)+p^*(0,6,0)+p^*(3,2,1) \\ &\quad +p^*(2,3,1)+p^*(1,4,1)+p^*(0,5,1)=0.1702 \\ p(2,2,2) &= p^*(2,2,2)+p^*(1,3,2)+p^*(0,4,2)+p^*(2,1,3) \\ &\quad +p^*(1,2,3)+p^*(0,3,3)+p^*(2,0,4) \\ &\quad +p^*(1,1,4)+p^*(0,2,4)+p^*(1,0,5) \\ &\quad +p^*(0,1,5)+p^*(0,0,6)=0.7581. \end{aligned}$$

The mean number of jobs $\bar{k}_i$ in each station is computed from (9):

$$\bar{k}_1 = 2.238 \quad \bar{k}_2 = 1.927 \quad \bar{k}_3 = 1.827.$$

The throughput of the blocking network is determined by the throughput analysis method. First the number of states in

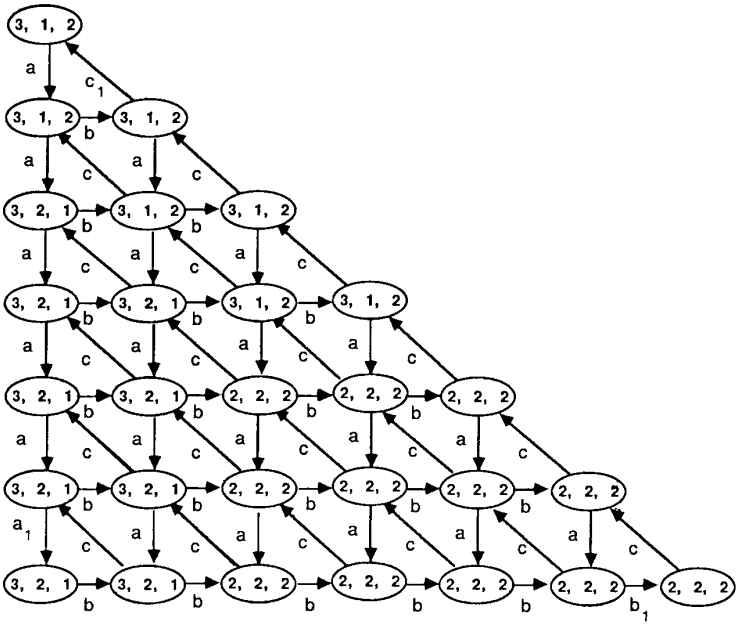


Fig. 3. Normalized state space.

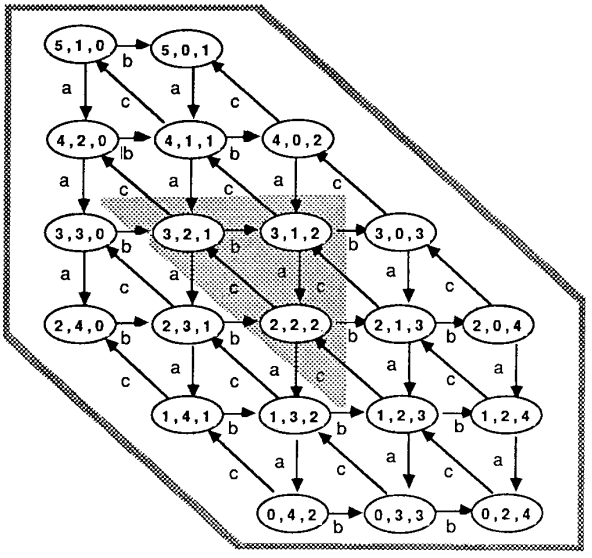


Fig. 4. State space of the blocking network.

the blocking network is computed by (14):

$$Z' = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 6 \\ 10 \\ 15 \\ 19 \\ 21 \end{bmatrix}.$$

The number of states in this model is obtained from the last element $Z'(6) = 21$ of the vector result Z'. By drawing the state space in Fig. 4, the feasible states and their two immediate neighbors (two neighbors are required $m_i = 2$ as servers) representing blocking states, provide the same result $Z'(6) = 21$ as the algorithm, (14). The innermost dotted lines in Fig. 4 contain the feasible states where the outermost dotted lines contain their neighbors as blocking states. The states where capacity violations occur are cancelled.

TABLE I

$\hat{K}$	$\hat{Z}$
0	1
1	3
2	6
3	10
4	15
5	21
6	28
7	36
8	44

The total number of jobs $\hat{K}$ in the nonblocking network is obtained by (16):

$$\hat{Z} = \binom{\hat{K}+2}{2}.$$

By varying the number of jobs $\hat{K}$, we obtain different number of states in the nonblocking network as shown in Table I. Since the nonblocking network with $\hat{K} = 5$ jobs possesses $\hat{Z} = 21$ states which is equal to the number of states $Z'(6) = 21$ of the blocking network, we assume that the nonblocking network has $\hat{K} = 5$ jobs. The state space for $\hat{K} = 5$ is shown in Fig. 5.

Obviously, the state space structure (Fig. 5) is different from state space structure of the blocking network, Fig. 4. Hence, no unique transformation of the states is possible such that the results could be exact. However, the number of states

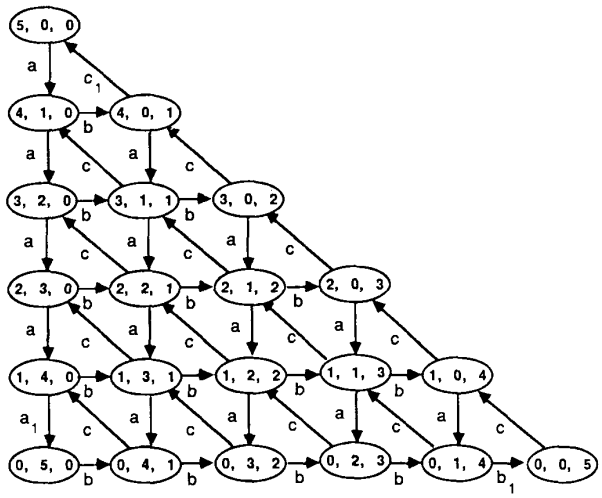


Fig. 5. State space of the nonblocking network.

TABLE II

	Analytical	Simulation	δ(%)	NOCAP	δ(%)
$\bar{t}_1$	2.711	2.779	2.4	1.164	58
$\bar{t}_2$	2.335	2.261	1.4	2.100	7
$\bar{t}_3$	2.214	2.172	1.9	3.574	65
k_1	2.238	2.312	3.2	1.021	56
k_2	1.927	1.881	2.5	1.843	2
k_3	1.827	1.807	1.1	3.135	74
λ	0.8253	0.8318	0.07	0.8772	6

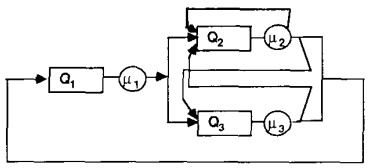


Fig. 6. Noncyclic blocking network.

in both systems is equal. Thus, the throughput of the nonblocking network with $\hat{K} = 5$ total number of jobs is approximately equal to the throughput of the blocking network with $K = 6$ jobs, i.e.,

$$\lambda_B(6) \approx \lambda_{NB}(5) = 0.8253.$$

The mean residence time $\bar{t}_i$ is obtained by Little's law, (20). All these results are shown and compared to simulation in Table II.

Note that NOCAP in Table II denotes that the network is analyzed by ignoring the finiteness of the station capacities (NOCAPacity). The deviations δ are computed using the following relative deviation formula:

$$\delta = \left\{ \frac{|\text{Simulation Value} - \text{Analytical Value}|}{\text{Simulation Value}} \right\} \times 100.$$

B. Noncyclic Network

As mentioned before, the normalization procedure, (3), causes the number of jobs in a state to have noninteger values in the case of networks with arbitrarily connected stations. In order to explain this fact, we analyze a closed queueing network with $N = 3$ arbitrarily switched single server stations as shown in Fig. 6. There are $K = 3$ jobs which belong to the same class. The service time of each station is exponentially distributed.

The network parameters are given as follows:

Station	M_i	$(1/\mu_i)$	p_{i1}	p_{i2}	p_{i3}
1	1	1	0	0.6	0.4
2	4	1.25	0.3	0.2	0.5
3	3	2	0.3	0.7	0

The mean number of visits e_i is computed by (4)

$$e_1 = 1 \quad e_2 = 1.955 \quad e_3 = 1.378.$$

We check the deadlock-free property, (2). In the cycle between stations 1 and 2, the total number of jobs is less than the total capacity of both networks, i.e.,

$$\{K = 3\} < \left\{ \sum_{j=1}^{2} M_j = 5 \right\}.$$

In the cycle between stations 1 and 3, condition (2) also holds:

$$\{K = 3\} < \left\{ \sum_{j=1 \& j \neq 2}^{3} M_j = 4 \right\}$$

also in the cycle between stations 2 and 3, condition (2) holds:

$$\{K = 3\} < \left\{ \sum_{j=2}^{3} M_j = 7 \right\}.$$

and finally in the cycle 2 and 2, condition (2) is again satisfied, i.e.,

$$\{K = 3\} < \{M_2 = 4\}$$

The state space of the given network without considering the station capacities is shown in Fig. 7.

In Fig. 7, $a = \mu_1 p_{12}$; $b = \mu_2 p_{21}$; $c = \mu_2 p_{23}$; $d = \mu_3 p_{32}$; $e = \mu_1 p_{13}$; $f = \mu_3 p_{31}$; $g = \mu_2 p_{22}$ denote the transition rates between the states. It is obvious that the states (3, 0, 0), (2, 1, 0), and (2, 0, 1) are nonfeasible. These states exceed the first station's capacity and must be normalized using (3).

In the state, (3, 0, 0), the number of jobs in the first station exceeds the capacity ($M_1 = 1$), so it is set equal to its capacity:

$$k_1 = 1$$

the remaining jobs are distributed as follows:

$$k_2 = k_2 + (k_1 - M_1)\frac{e_2 p_{21}}{e_1(1 - p_{11})} = 1.173$$

$$k_3 = k_3 + (k_1 - M_1)\frac{e_3 p_{31}}{e_1(1 - p_{11})} = 0.8268.$$

Since none of the station capacities are violated, the normalization procedure yields

$$f(3,0,0) = (1, 1.173, 0.8268).$$

This is the situation that is mentioned before. The transition probabilities cause noninteger values for the job distributions in a normalized state.

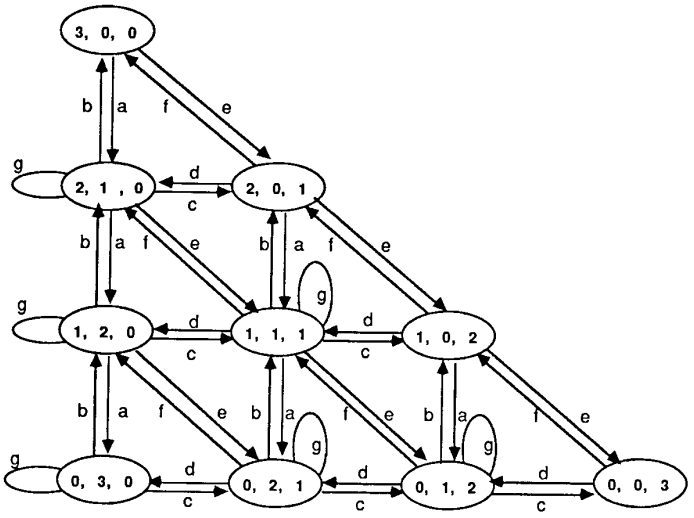


Fig. 7. State space by ignoring station capacities.

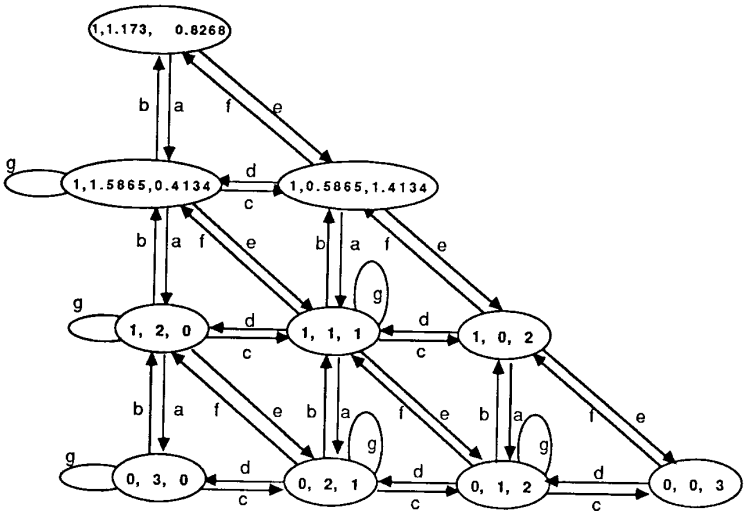


Fig. 8. Normalized state space.

Another nonfeasible state, (2, 1, 0), is normalized:

$$f(2,1,0) = (1,1.5865,0.4134).$$

The nonfeasible state (2, 0, 1) is normalized as

$$f(2,0,1) = (1,0.5865,1.4134).$$

After normalization of all nonfeasible states, we obtain the state space given in Fig. 8.

Now we ignore the station capacities and compute the equilibrium state probabilities by (7):

The equilibrium probabilities for the feasible states of the blocking network are computed by (6):

$$p(1,1.173,0.8268) = p^*(3,0,0) = 0.0103$$

$$p(1,1.5865,0.4134) = p^*(2,1,0) = 0.0251$$

$$p(1,0.5865,1.4134) = p^*(2,0,1) = 0.0284$$

$$p(1,1,1) = p^*(1,1,1) = 0.0694$$

$$p(1,2,0) = p^*(1,2,0) = 0.0615$$

$$p(0,2,1) = p^*(0,2,1) = 0.1695$$

$p^*(3,0,0) = 0.0103$ $p^*(1,1,1) = 0.0694$ $p^*(2,1,0) = 0.0251$ $p^*(0,0,3) = 0.2157.$
$p^*(0,2,1) = 0.1695$ $p^*(1,2,0) = 0.0615$ $p^*(1,0,2) = 0.0783$
$p^*(0,3,0) = 0.1502$ $p^*(0,1,2) = 0.1912$ $p^*(2,0,1) = 0.0284$

TABLE III

	Approx.	Exact	δ(%)	NOCAP	δ(%)
$\bar{t}_1$	1	1	0	1.2716	27
$\bar{t}_2$	2.328	2.3343	0	2.2465	3.8
$\bar{t}_3$	3.943	4.1619	5.2	3.8621	7.2
$\bar{k}_1$	0.273	0.2655	2.8	0.3472	31
$\bar{k}_2$	1.243	1.2119	2.5	1.1996	1
$\bar{k}_3$	1.484	1.5226	2.5	1.4530	4.6
λ	0.273	0.2655	2.8	0.273	0

$$p(0,3,0)=p^*(0,3,0)=0.1502$$

$$p(0,1,2)=p^*(0,1,2)=0.1912$$

$$p(1,0,2)=p^*(1,0,2)=0.0783$$

$$p(0,0,3)=p^*(0,0,3)=0.2157.$$

From (11) we calculate the mean number of jobs $\bar{k}_i$ in each station, e.g.,

$$\bar{k}_1 = f_1(3,0,0)p(3,0,0)+f_1(2,1,0)p(2,1,0)+f_1(1,2,0)p(1,2,0)+f_1(2,0,1)p(2,0,1)+f_1(1,1,1)p(1,1,1)+f_1(1,0,2)p(1,0,2)=0.273.$$

Similarly we calculate the mean number of jobs in stations 2 and 3.

The *total throughput*, (17), and the *throughput of each station*, (18), and the mean response time $\bar{t}_i$, (20), are computed, listed, and compared with exact results in Table III.

V. Evaluation

We have introduced two different techniques for the computation of performance measures in blocking networks with multiple servers. Both techniques have been implemented on a VAX 11/780 system. For the validation of both techniques, we have executed 200 network examples with different structures. Half of the examples were networks with arbitrarily connected stations. Each network model is also analyzed by varying the number of jobs. The number of jobs is varied from 5 to 100, the number of stations from 3 to 8, and the number of servers from 1 to 8. All examples have been simulated using the RESQ package [35].

Throughout the tests, we could not find any network model for which our approximations demonstrated instabilities. In all examples we have observed the same behavior. The fewer the jobs in the queueing network with finite station capacities, the less the chance for blocking. After a certain number of jobs in the network is reached, blocking events start to occur. As a result, the throughput does not increase with the number of jobs in the network. As the number of jobs in the network approaches the total capacity of the network (the sum of all station capacities), the more blocking events may occur. This has the effect of reducing the state space of the blocking network, which can be seen in Fig. 9. The solid line shows the number of states for a blocking queueing network consisting of three stations with station capacities $M_1 = 12$, $M_2 = 10$, and $M_3 = 14$, the number of servers $m_1 = 4$, $m_2 = 2$, and $m_3 = 5$ and the dashed line shows the number of states of the same network without station capacity limitations for different numbers of jobs. By dotted line, we show that the number of states in the blocking network is significantly reduced when

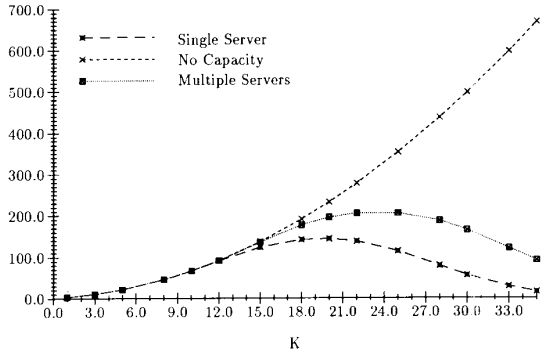


Fig. 9. Number of states dependent on the number of jobs.

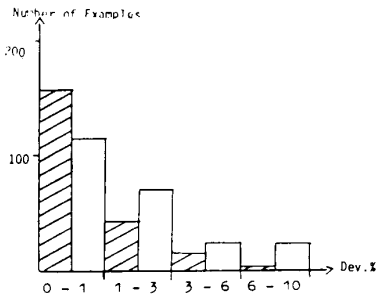


Fig. 10. Histogram 1.

the stations have single servers. It is obvious that the station capacities have the effect of decreasing the number of states as the number of jobs approaches the total capacity of the network.

In our test cases, most of the deviations for the throughput values are below 2 percent, implying that our throughput technique provides very accurate results for throughput and utilization. As can easily be seen in the chart in Fig. 10, the throughput values of the 65 examples in case of tandem networks (shaded portions) and of 43 in case of nontandem networks lie in the range of 0–1 percent deviation.

The major advantage of this technique is that the throughput values are obtained from a product form network which can easily be analyzed by mean value analysis or by any other product form network algorithm. Since mean value analysis has the advantage that it is extremely fast, it follows that we can obtain results for even a large queueing network with blocking in a very short time.

For the computation of the mean number of jobs $\bar{k}_i$, we need the equilibrium state probabilities which are given by (6). The method has shown a very stable behavior in the computation of the $\bar{k}_i$ values. The vast majority of the deviations were in the range of 1–5 percent as shown in histogram 2 in Fig. 11.

Compared to the throughput values, the deviations for the mean number of jobs are higher which can be observed in both histograms 1 and 2. Extremely small numbers cause deviations of 35 percent which makes 6 percent of the cases in histogram 2. As generally known, small numbers can provide high deviations even though the numerical results are not too different. In the histogram 3, in Fig. 12, we show the

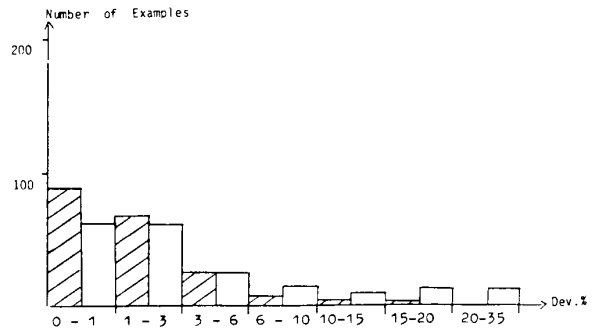


Fig. 11. Histogram 2.

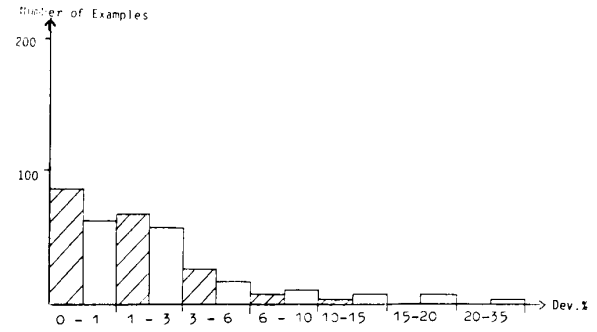


Fig. 12. Histogram 3.

percentage of errors by ignoring the small (less than $K \times N/10$) numerical results.

As it can be seen in histogram 3, the highest deviation for tandem networks is decreased to 15 percent. In the case of nontandem networks, the number of cases in the range 20 percent and 35 percent has also decreased enormously.

The normalization procedure has a fairly long run time. However, it is not nearly as long as the long run times of simulation or global balance techniques. The latter are only applicable to very small networks. As pointed out before, the normalization procedure, (3), for the computation of the mean number of jobs can yield noninteger values for the job distribution in a state. The normalization of states must be proceeded until no station's capacity is violated. Here the problem arises that the normalization can cause the capacity violation of another station. Further normalization would result in another violation of the previous mentioned station's capacity. For example, if the capacity of ith station is violated, the normalization causes station j to obtain some jobs which leads to its capacity violation. Further normalization leads to a situation that station i gets some jobs back so that its capacity is again violated. In other words, in the normalization procedure, two or more stations can keep exchanging the jobs between each other. This can cause long run times. Based on our test examples, the normalization procedure always converges, i.e., a normalized state is always reached in which the station's capacities are not violated. This will be explained in detail.

Let us consider the example given in Section IV-B. Ignoring the possibility of deadlock, assume that the station's capacities are selected as $M_1 = 2$, $M_2 = 1$ and $M_3 = 1$. An interesting

TABLE IV

$k_2 = 1$ $k_1 = 0.897$ $k_3 = 1.102$
$k_3 = 1$ $k_1 = 0.926$ $k_3 = 1.072$
$k_2 = 1$ $k_1 = 0.953$ $k_3 = 1.044$

normalization occurs in case of the nonfeasible state (0, 3, 0). The capacity of the second station is exceeded. So we apply (3) and obtain

$$k_2 = 1$$

$$k_1 = 0.766$$

$$k_3 = 1.234.$$

Here we see that the normalization procedure, (3), resets the number of jobs in the second station to its capacity. However, some of the remaining jobs cause the capacity of the third station to be exceeded. In this case, we set the number of jobs in the third station back to its capacity and obtain

$$k_3 = 1$$

$$k_1 = 0.834$$

$$k_2 = 1.166.$$

This time the capacity of the second station is violated again. The normalization causes both stations to swap jobs back and forth to each other. However, the remaining number of jobs is shuffled to the first station in each new normalization step. This is shown in Table IV.

As it can easily be seen in Table IV, the state values are tending towards (1, 1, 1). Note that in the implementation of this algorithm, an epsilon value of (eps $= 10^{-3}$) prevents long run times for the normalization in such situations where two or more stations effect each other. If, after a normalization, the number of jobs exceeding the capacity less than the eps value, then the normalization is assumed to be done. In the example above, Table IV, the value for k_3 violates the capacity by exceeding it by only 0.044. There is no reason to continue the normalization procedure until the values (1, 1, 1) are reached. Similar situations also occur for states (0, 1, 2), (0, 0, 3), and (0, 2, 1), which are normalized to the state (1, 1, 1).

Suri and Diehl [38] also consider closed queueing networks with Type 1 (transfer) blocking. They assume that the first station must have a capacity greater than the number of jobs in the network. They apply Norton's theorem [27], [35], [40] and reduce each two-station pair to a single station with a variable size queue capacity that is easy to analyze. An approximation algorithm is derived for the total throughput of the network. In the following we compare our throughput results to those of Suri and Diehl [38] and plot the throughput results for three blocking queueing networks with various number of jobs. Each graph contains three computations of total throughputs for various number of jobs: a) our algorithm, b) exact or simulation, c) Suri/Diehl results.

Example 1:

Station	M_i	$(1/\mu_i)$
1	∞	0.6
2	5	1
3	4	0.8

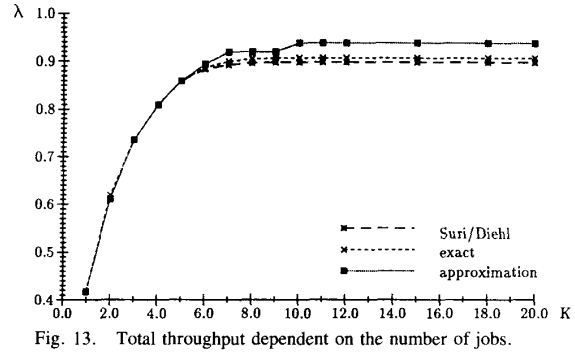


Fig. 13. Total throughput dependent on the number of jobs.

Example 2:

Station	M_i	$(1/\mu_i)$
1	∞	1.0
2	6	2.0
3	8	3.5
4	4	4.6
5	7	1.2

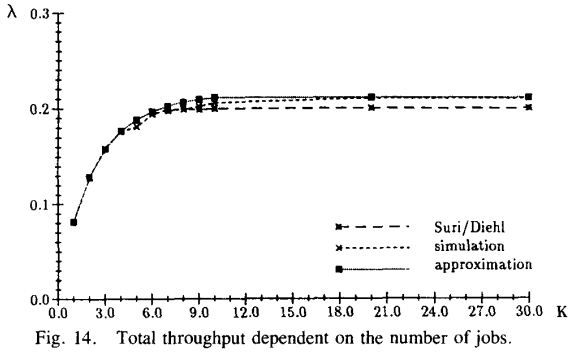


Fig. 14. Total throughput dependent on the number of jobs.

Example 3:

• $N = 10$ stations; $M_1 = \infty$; $M_i = 3$ for $i = 2, \cdots, 10$; $\mu_i = 1$ for $i = 1, \cdots, 10$.

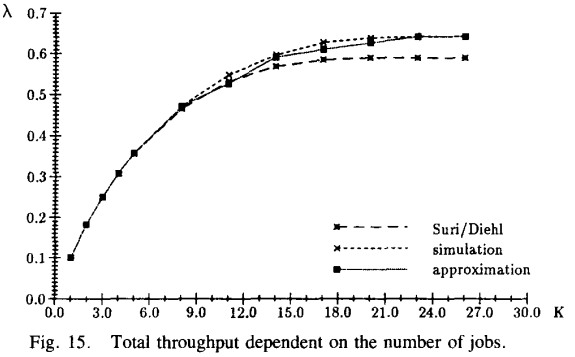


Fig. 15. Total throughput dependent on the number of jobs.

In the following tables, we give total throughput results for different network examples which are taken from Suri and Diehl [33].

Example 4:

$N = 3$ stations (cyclic); $K = 5$ jobs; $M_1 = \infty$; $M_2 = 1$; $M_3 = 2$;

$1/\mu_1$	$1/\mu_2$	$1/\mu_3$	Exact	Our Approx.	δ	Suri/Diehl	δ
10	1	1	0.0999	0.0997	0	0.0992	0
10	1	5	0.0957	0.0924	3.4	0.0961	0
10	1	10	0.0793	0.0743	6.3	0.0796	0
10	1	20	0.0483	0.0465	3.7	0.0483	0
10	1	100	0.0100	0.0100	0	0.0099	0
10	5	1	0.0857	0.0924	7.7	0.0856	0
10	5	5	0.0846	0.0846	0	0.0847	0
10	5	10	0.0733	0.0694	5.3	0.0734	0
10	5	20	0.0472	0.0452	4.3	0.0472	0
10	5	100	0.0099	0.0100	0	0.0099	0
10	10	1	0.0667	0.0730	9.4	0.0664	0
10	10	5	0.0661	0.0694	4.9	0.0649	1.8
10	10	10	0.0613	0.0600	0	0.0600	2.1
10	10	20	0.0443	0.0423	4.5	0.0438	1.1
10	10	100	0.0099	0.0100	0	0.0099	0
10	20	1	0.0428	0.0465	8.6	0.0426	0
10	20	5	0.0427	0.0452	5.8	0.0413	3.2
10	20	10	0.0417	0.0423	1.4	0.0396	5.1
10	20	20	0.0357	0.0347	2.8	0.0342	4.0
10	20	100	0.0099	0.0099	0	0.0099	0
10	100	1	0.00990	0.0100	1	0.0099	0
10	100	5	0.00990	0.0100	1	0.0098	0
10	100	10	0.00990	0.0100	1	0.0097	1.3
10	100	20	0.00986	0.0099	0	0.0095	3.0
10	100	100	0.00749	0.0075	0	0.0072	3.5

Example 5:

$N = 3$ stations (cyclic); $K = 5$ jobs; $M_1 = \infty$; $M_2 = 2$; $M_3 = 1$;

$1/\mu_1$	$1/\mu_2$	$1/\mu_3$	Exact	Our Approx.	δ	Suri/Diehl	δ
10	10	5	0.0714	0.0694	2.8	0.0697	2.4
10	10	10	0.0613	0.0610	0	0.0595	2.9
10	10	20	0.0420	0.0423	0	0.0415	1
10	20	10	0.0417	0.0423	1.3	0.0400	4.15
10	20	20	0.0330	0.0340	2.8	0.0326	1.2

As can easily be seen in the graphs and also in the tables, both techniques provide very accurate results. However, the method of Suri and Diehl has the disadvantage that it provides only the total throughput (and using Little's law the total response time) of the network. In other words, performance measures for individual stations cannot be obtained. Another restriction is the assumption that the first station must have an infinite capacity. Additional restriction of the Suri–Diehl algorithm is that it is applicable only to closed tandem networks with single servers where our algorithm, Section III, is also applicable to nontandem networks with multiple servers. Our study also showed that the method of Suri and Diehl needs long run time (even though it is short compared to simulation) because of the computation of the marginal probabilities which are then used as the variable buffer sizes for the flow-equivalent server. Another major drawback in Suri and Diehl's method as mentioned in Section I, is that the capacity of each downstream station must be smaller than the total number of jobs in the network. This fact can be observed in examples 1, 2, and 3 where the results for Suri and Diehl do not appear for all possible number of jobs in the network.

In conclusion, we point out that our approximate techniques can be considered as an efficient way to compute the performance measures of queueing networks with blocking. The approximations are stable and the results obtained are accurate. The run times are short in particular for the computation of the throughput values.

APPENDIX

We give results for 10 blocking queueing networks with various number of jobs. We have a total of 25 test cases with different system input parameters. Each table contains three computations of results for various number of jobs: 1) our algorithm 2) simulation, and 3) NOCAP. The third column contains the standard deviation of simulation results. δ_1 shows the relative deviations between our results and the simulation. δ_2 shows the deviations between NOCAP and the simulation. This column demonstrates the effects of finite station capacity on the performance of the network.

Example 1. (Cyclic):

Station	M_i	$(1/\mu_i)$	m_i
1	12	2	2
2	14	1	2
3	8	0.66	2

a) $K = 22$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	11.9703	11.92	2.4	0.42	19.926	67
$\bar{k}_2$	2.6233	2.38	4.6	10.03	1.333	44
$\bar{k}_3$	7.4065	7.70	1.3	3.81	0.741	91
λ	0.9999	0.99	1.3	0.32	0.999	0.32

b) $K = 30$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	11.999	11.93	1.2	0.58	27.93	134
$\bar{k}_2$	10.001	10.35	2.6	3.37	1.33	87
$\bar{k}_3$	7.999	7.72	1.3	3.56	0.74	91
λ	0.997	0.993	0.1	0.51	1	0

Example 2. (Cyclic):

Station	M_i	$(1/\mu_i)$	m_i
1	25	1	1
2	15	2	1
3	25	2	1
4	10	1	1
5	30	1	1

a) $K = 40$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	24.096	24.580	0.06	1.9	1.999	92
$\bar{k}_2$	14.999	14.500	0.9	3.3	37.096	173
$\bar{k}_3$	0.154	0.157	0.1	1.9	0.153	2
$\bar{k}_4$	0.500	0.499	0.5	0	0.501	0
$\bar{k}_5$	0.250	0.250	1.1	0	0.25	0
λ	1.333	1.344	1.1	0.8	1.333	0

b) $K = 50$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	25	24.97	0.07	0	1.999	92
$\bar{k}_2$	15	13.69	0.3	9.5	47.096	244
$\bar{k}_3$	0.15	0.15	0.8	0	0.153	0
$\bar{k}_4$	0.5	0.49	1	0	0.501	0
$\bar{k}_5$	9.34	10.72	0.4	12.8	0.25	98
λ	1.333	1.330	0.2	0	1.333	0

Example 3. (Cyclic):

Station	M_i	$(1/\mu_i)$	m_i
1	14	1.2	4
2	13	1.3	4
3	7	1.4	4
4	13	1.3	4
5	14	1.2	4

a) $K = 25$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	4.2326	4.238	1.1	0.1	4.1819	10
$\bar{k}_2$	6.3916	6.896	2.8	7.3	5.1098	26
$\bar{k}_3$	5.1233	4.910	2.4	4.3	6.4164	31
$\bar{k}_4$	5.0706	4.836	2.3	4.8	5.1098	6
$\bar{k}_5$	4.1819	4.121	2.6	1.4	4.1819	1
λ	2.4744	2.485	1.2	0.4	2.5017	0

b) $K = 50$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	12.6666	12.791	1.1	10.9	6.398	50
$\bar{k}_2$	12.6622	11.811	3.3	7.2	9.692	18
$\bar{k}_3$	6.7877	5.588	4.8	21.4	17.818	220
$\bar{k}_4$	8.3277	7.838	3.5	6.2	9.692	24
$\bar{k}_5$	9.5558	11.971	4.6	20.1	6.398	47
λ	2.5701	2.548	1.3	0.8	2.7829	9

c) $K = 60$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	13.9456	13.85	0.9	0.6	6.8624	50
$\bar{k}_2$	12.9844	12.85	0.62	1	11.0986	14
$\bar{k}_3$	6.9708	6.78	0.41	2.6	24.0777	250
$\bar{k}_4$	12.4043	12.72	0.07	2.4	11.0986	13
$\bar{k}_5$	13.6949	13.80	0.62	0.7	6.8624	50
λ	2.3737	2.17	0.41	9.2	2.8151	30

Example 4. (Cyclic):

Station	M_i	$(1/\mu_i)$
1	12	2
2	10	1
3	14	1.5

- $K = 33$ jobs

a) $m_i = 2$ for $i = 1, 2, 3$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	12	11.94	0.7	0.5	28.242	136
$\bar{k}_2$	7.009	7.31	1.1	4.1	1.333	82
$\bar{k}_3$	13.99	13.76	0.9	1.6	3.424	75
λ	0.995	0.989	1.2	0.6	0.999	0

b) $m_1 = 2$; $m_2 = 5$; $m_3 = 5$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	12	11.99	0.9	0.08	30.91	154
$\bar{k}_2$	7	7.008	0.65	0	1.01	86
$\bar{k}_3$	14	14	0.3	0	1.51	89
λ	1	0.995	0.4	0.4	1.00	0

c) $m_1 = 3$; $m_2 = 8$; $m_3 = 5$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	12	11.94	1	0.5	29.175	144
$\bar{k}_2$	7	7.05	8.2	0.7	1.5	79
$\bar{k}_3$	14	14	4.2	0	2.324	84
λ	1.5	1.49	2.4	0.4	1.5	0

d) $m_1 = 4$; $m_2 = 2$; $m_3 = 5$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	11.99	11.92	2.1	0.6	15.541	30
$\bar{k}_2$	9.10	8.64	3.2	5.2	14.281	65
$\bar{k}_3$	11.90	12.44	1.8	4.3	3.178	75
λ	1.78	1.73	0.8	3	1.929	11

Example 5. (Noncyclic):

• $K = 9$ jobs;

Station	M_i	$(1/\mu_i)$	p_{i1}	p_{i2}	p_{i3}
1	6	2.5	0	0.5	0.5
2	8	1.2	0.7	0	0.3
3	6	1	0.7	0.3	0

a) $m_1 = 2$; $m_2 = 5$; $m_3 = 3$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	5.94	5.92	2.1	0.4	7.73	30.6
$\bar{k}_2$	1.58	1.66	1.7	4.6	0.68	58.7
$\bar{k}_3$	1.47	1.66	1.4	11.2	0.57	65.2
λ	0.8	0.79	0.7	0.37	0	0.79

b) $m_1 = 5$; $m_2 = 2$; $m_3 = 5$

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	4.84	4.56	1.1	5.9	5.31	16
$\bar{k}_2$	2.68	2.54	1.2	5.6	2.45	3.5
$\bar{k}_3$	1.47	1.89	4.7	22	1.25	34
λ	1.744	1.661	0.6	0.5	1.74	0.5

Example 6. (Noncyclic):

Station	M_i	$(1/\mu_i)$	p_{i1}	p_{i2}	p_{i3}
1	6	2.5	0	0.5	0.5
2	8	1.2	0.7	0	0.3
3	6	1	0.7	0.3	0

a) $K = 8$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	5.846	5.765	0.6	1.4	7.081	22.8
$\bar{k}_2$	1.137	1.202	0.3	5.4	0.520	56.7
$\bar{k}_3$	1.016	1.032	2.8	1.4	0.399	61.2
λ	0.399	0.398	0.2	0	0.399	0

b) $K = 10$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	5.978	5.844	0.3	2.3	9.078	55
$\bar{k}_2$	2.071	2.306	2.3	10	0.521	77
$\bar{k}_3$	1.95	1.848	3.8	5.4	0.399	78
λ	0.399	0.401	2.3	0	0.399	0

c) $K = 11$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	5.992	5.867	0.3	2.1	10.078	72
$\bar{k}_2$	2.564	2.853	3.9	10.1	0.522	82
$\bar{k}_3$	2.443	2.278	4.1	7.2	0.399	82
λ	0.399	0.399	0	0.399	0	

Example 7. (Noncyclic):

Station	M_i	$(1/\mu_i)$	m_i	p_{i1}	p_{i2}	p_{i3}	p_{i4}
1	11	2	5	0	0.8	0.1	0.1
2	4	1.3	3	1	0	0	0
3	2	0.2	2	1	0	0	0
4	2	0.2	2	1	0	0	0

a) $K = 11$ jobs;

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	7.97	8.18	2.1	2.5	7.01	14.4
$\bar{k}_2$	2.93	2.72	3.1	7.6	3.89	43.2
$\bar{k}_3$	0.04	0.04	0.7	2.8	0.046	2.8
$\bar{k}_4$	0.04	0.04	3.7	5.5	0.046	5.6
λ	2.28	2.23	1.3	2.4	2.33	4.3

b) $K = 12$ jobs;

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	8.87	9.13	0.2	2.8	7.71	15.6
$\bar{k}_2$	3.02	2.76	2.5	9.2	4.19	52
$\bar{k}_3$	0.05	0.05	0.4	1	0.04	7.3
$\bar{k}_4$	0.05	0.05	2.2	4.1	0.05	0
λ	2.33	2.23	3.1	4.2	2.36	5.6

Example 8. (Noncyclic):

Station	M_i	$(1/\mu_i)$	m_i	p_{i1}	p_{i2}	p_{i3}	p_{i4}
1	5	1	1	0	0.4	0.2	0.4
2	5	2	1	0.3	0	0.4	0.3
3	5	2	1	0.2	0.4	0	0.4
4	5	1	1	0.4	0.2	0.4	0

a) $K = 8$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	0.830	0.827	6.5	0	0.641	23
$\bar{k}_2$	2.996	2.993	6.5	0	3.17	6
$\bar{k}_3$	3.137	3.124	9.9	0	3.37	8
$\bar{k}_4$	1.036	1.054	6.1	1.8	0.81	23
λ	0.398	0.388	2.4	2.5	0.398	2.5

b) • $K = 9$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	0.984	0.963	1.2	2.2	0.666	31
$\bar{k}_2$	3.321	3.321	7.4	0	3.615	9
$\bar{k}_3$	3.468	3.461	1.1	0	3.867	12
$\bar{k}_4$	1.226	1.255	5.7	2.3	0.851	32
λ	406	0.383	2.2	6	0.406	6

Example 9. (Noncyclic):

Station	M_i	$(1/\mu_i)$	m_i	p_{i1}	p_{i2}	p_{i3}	p_{i4}
1	15	4	6	0	0.5	0	0.5
2	2	2	2	0	0	1	0
3	2	1.7	2	1	0	0	0
4	4	2	2	1	0	0	0

a) $K = 7$ jobs;

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	4.04	4.06	1.1	0.3	3.86	5
$\bar{k}_2$	1.03	1.02	1.2	0.5	1.11	8.5
$\bar{k}_3$	0.81	0.82	1.7	0.7	0.91	10.5
$\bar{k}_4$	1.10	1.09	2.1	1.3	1.11	2.1
λ	0.96	0.95	0.7	0.3	0.96	0.3

b) $K = 14$ jobs;

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	9.31	9.72	1.1	4.2	7.68	21
$\bar{k}_2$	1.53	1.45	1.2	5.5	2.31	60
$\bar{k}_3$	1.17	1.07	2.1	8.9	1.67	56
$\bar{k}_4$	1.99	1.75	3.1	13.3	2.32	32
λ	1.27	1.25	2.2	1.4	1.38	10

c) $K = 16$ jobs;

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	11.11	11.71	2.1	5.1	9.04	23
$\bar{k}_2$	1.57	1.45	0.7	2.5	8.5	78
$\bar{k}_3$	1.21	1.08	0.9	12.2	1.82	68
$\bar{k}_4$	2.11	1.75	0.74	20.1	2.57	46
λ	1.32	1.25	0.6	4.9	1.42	14

Example. 10. (Noncyclic):

Station	M_i	$(1/\mu_i)$	m_i	p_{i1}	p_{i2}	p_{i3}
1	5	1	1	0	0.4	0.2
2	5	2	1	0.3	0	0.4
3	5	2	1	0.2	0.4	0
4	5	1	1	0.4	0.2	0.4

a) $K = 8$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	0.83	0.82	0.5	0	0.64	23
$\bar{k}_2$	2.99	2.99	0.3	0	3.17	6
$\bar{k}_3$	3.13	3.12	0.7	0	3.37	8
$\bar{k}_4$	1.03	1.05	0.9	1.8	0.81	23
λ	0.398	0.38	0.3	2.6	0.39	2.6

b) $K = 9$ jobs

	Approx.	Simul.	Std. Dev.	$\delta_1(\%)$	NOCAP	$\delta_2(\%)$
$\bar{k}_1$	0.984	0.963	1.1	2.2	0.666	31
$\bar{k}_2$	3.321	3.321	0.2	0	3.615	9
$\bar{k}_3$	3.468	3.461	0.7	0	3.867	12
$\bar{k}_4$	1.226	1.255	1.2	2.3	0.851	32
λ	0.406	0.383	0.4	5.8	0.406	5.8

References

[1] I. F. Akyildiz, "Exact product form solution for queueing networks with blocking," *IEEE Trans. Comput.*, vol. 1, pp. 122–126, Jan. 1987.

[2] ——, "General closed queueing networks with blocking," in *Perform. '87 Proc.*, pp. 283–303.

[3] ——, "On the exact and approximate throughput analysis of closed queueing networks with blocking," *IEEE Trans. Software Eng.*, vol. SE-14, pp. 62–71, Jan. 1988.

[4] ——, "Mean value analysis for blocking queueing networks," *IEEE Trans. Software Eng.*, vol. SE-14, pp. 418–429, Apr. 1988.

[5] I. F. Akyildiz and J. Liebeherr, "Application of Norton's theorem on queueing networks with finite capacities," Tech. Rep., Georgia Tech. GIT 88-024, July 1988. Also in *Proc. Comput. Networks, INFOCOM '89 Conf.*, to be published.

[6] I. F. Akyildiz and H. von Brand, "Exact solutions for open, closed and mixed queueing networks with rejection blocking," *Theoret. Comput. Science J.*, to be published.

[7] ——, "Duality in open and closed Markovian queueing networks with rejection blocking," Tech. Rep., LSU-TR-87-011, Apr. 1987.

[8] T. Altiok, "Approximate analysis of exponential tandem queues with blocking," *Euro. J. Oper. Res.*, vol. 11, 1982, pp. 390–397.

[9] S. Balsamo and G. Iazeoalla, "Some equivalence properties for queueing networks with and without blocking," in *Proc. Perform. 83 Conf.*, 1983, pp. 351–360.

[10] F. Baskett, K. M. Chandy, R. R. Muntz, and G. Palacios, "Open, closed and mixed network of queues with different classes of customers," *J. ACM*, vol. 22, pp. 248–260, Apr. 1975.

[11] O. Boxma and A. Konheim, "Approximate analysis of exponential queueing systems with blocking," *Acta Informatica*, vol. 15, pp. 19–66, Jan. 1981.

[12] A. Brandwajn and Y. L. Jow, "An approximation method for tandem queues with blocking," *Oper. Res.*, June 1988.

[13] P. J. Caseau and G. Pujolle, "Throughput capacity of a sequence of queues with blocking due to finite waiting room," *IEEE Trans. Software Eng.*, vol. SE-5, pp. 631–642, Nov. 1979.

[14] G. W. Diehl, "A buffer equivalency decomposition approach to finite buffer queueing networks," Ph.D. dissertation, Harvard Univ., May 1984.

[15] N. M. van Dijk and I. F. Akyildiz, "Networks of mixed processor sharing parallel queues and common pools," Tech. Rep., Georgia Tech, GIT-ICS-88-022, June 1988.

[16] W. J. Gordon and G. F. Newell, "Cyclic queueing systems with restricted queues," *Oper. Res.*, vol. 15, pp. 266–277, Apr. 1967.

[17] K. Goto, Y. Takahashi, and T. Hasegawa, "An approximate analysis on controlled tandem queues," in *Proc. Int. Conf. Modeling Perform. Eval. Methodology*, Jan. 24–26, 1983, pp. 601–613.

[18] L. Gun and A. Makowski, "Matrix-geometric solution for finite capacity queues with phase-type distributions," in *Proc. Perform. 87 Conf.*, 1988, pp. 269–283.

[19] A. Hordijk and N. Van Dijk, "Networks of queues with blocking," in *Proc. Perform. 81, Modeling, Measurement, Evaluation*, Nov. 4–6, 1981, pp. 51–65.

[20] ——, "Networks of queues: Part I: Job-local balance and the adjoint processes; Part II: General routing and service characteristics," in *Proc. Int. Conf. Modeling Perform. Eval. Methodology*, Jan. 24–26, 1983, pp. 79–135.

[21] F. P. Kelly, "Networks of queues with customers of different types," *J. Appl. Prob.*, vol. 12, pp. 542–554, 1975.

[22] ——, *Reversibility and Stochastic Networks*. New York: Wiley, 1979.

[23] ——, "The throughput of a series of buffers," *Adv. Appl. Prob.*, vol. 14, pp. 633–653, 1982.

[24] A. G. Konheim and M. Reiser, "A queueing model with finite waiting room and blocking," *J. ACM*, vol. 23, pp. 328–341, Apr. 1976.

[25] ——, "Finite capacity queueing systems with applications in computer modeling," *SIAM J. Comput.*, vol. 7, pp. 210–229, May 1977.

[26] J. Labetoulle and G. Pujolle, "Isolation method in a network of queues," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 373–381, July 1980.

[27] E. Lazowska, J. Zahorjan, G. Graham, and K. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[28] R. O. Onvural and H. G. Perros, "On equivalencies of blocking mechanisms in queueing networks with blocking," *Oper. Res. Lett.*, vol. 5, pp. 293–297, Dec. 1986.

[29] ——, "Some equivalencies for queueing networks with blocking," *Perform. Eval. J.*, to be published.

[30] H. G. Perros, A. A. Nilsson, and Y. C. Liu, "Approximate analysis of product form type queueing networks with blocking and deadlock," *Perform. Eval.*, vol. 8, pp. 19–39, Feb. 1988.

[31] H. G. Perros and T. Altiok, "Approximate analysis of open networks of queues with blocking: Tandem configurations," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 450–462, Mar. 1986.

[32] B. Pittel, "Closed exponential networks of queues with saturation: The Jackson type stationary distribution and its asymptotic analysis," *Math. Oper. Res.*, vol. 4, pp. 367–378, 1979.

[33] M. Reiser and H. Kobayashi, "Queueing networks with multiple closed chains: Theory and computational algorithms," *IBM J. Res. Develop.*, vol. 19, pp. 283–294, May 1975.

[34] M. Reiser and S. S. Lavenberg, "Mean value analysis of closed multichain queueing networks," *J. ACM*, vol. 27, pp. 313–322, Apr. 1980.

[35] C. H. Sauer and K. M. Chandy, *Computer Systems Performance Modeling*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[36] P. Schweitzer and T. Altiok, "Aggregate modeling of tandem queues with blocking," in *Proc. Int. Workshop Applied Math. Performance/Reliability Models Commun. Syst.*, May 1987.

[37] R. Suri and G. Diehl, "A new building block for performance evaluation of queueing networks with finite buffers," *ACM Perform. Eval. Rev.*, 1984, pp. 134–142.

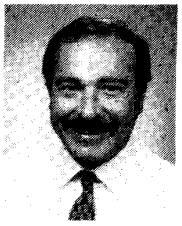
[38] R. Suri and G. W. Diehl, "A variable buffer-size model and its use in analyzing closed queueing networks with blocking," *Management Sci.*, vol. 32, pp. 206–225, Feb. 1986.

[39] Y. Takahashi, H. Miyahara, and T. Hasegawa, "An approximation method for open restricted queueing networks," *Oper. Res.*, vol. 28, pp. 594–602, May-June 1980.

[40] J. Walrand, *An Introduction to Queueing Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[41] G. Yamazaki, T. Kawashima, and H. Sakasegawa, "Reversibility of tandem blocking queueing systems," *Management Sci.*, vol. 31, Jan. 1985.

[42] D. D. Yao and J. A. Buzacott, "Modeling a class of state-dependent routing in flexible manufacturing systems," *Ann. Oper. Res.*, vol. 3, pp. 153–167, 1985.



I. F. Akyildiz (M'85) was born in 1954 in Istanbul, Turkey. He received the Vordiplan, Diplom Informatiker, and Doctor of Engineering degrees in computer science from the University of Erlangen-Nuernberg, West Germany in 1978, 1981, and 1984, respectively.

From 1981 through 1985 he served as a Scientific Employee in the Informatik IV (Operating Systems) at the University of Erlangen-Nuernberg. During that time, he coauthored a text book titled *Analysis of Computer Systems* (in German) published by Teubner Verlag in Fall 1982. In January of 1985, he joined the faculty of the Department of Computer Science, Louisiana State University as an Assistant Professor. He was also a Visiting Professor in the Department of Computer Science at the University of Florida, Gainesville, in the summer of 1985 and in the Computer Science Department of the Universidad Tecnica de Federico Santa Maria in Valparaiso, Chile in the summer of 1986. In Fall 1987, he joined the faculty in the School of Information and Computer Science at Georgia Institute of Technology as an Assistant Professor. His research interests are performance evaluation, operating systems, and computer networks.

Dr. Akyildiz is a member of the Association for Computing Machinery (Sigops and Sigmetrics), GI (Gesellschaft fuer Informatik), and MMB (German Interest Group in Measurement, Modeling and Evaluation of Computer Systems).