

**APPROXIMATE ANALYSIS OF ARBITRARY
CONFIGURATIONS OF QUEUEING NETWORKS WITH
BLOCKING AND DEADLOCK¹**

K.P. Jun

**Operations Research Program
and
Center for Communications and Signal Processing**

and

H.G. Perros

**Computer Science Department
and
Center for Communications and Signal Processing
North Carolina State University
Raleigh, NC 27695-8206**

CCSP-TR-88/11

March 1988

¹Supported in part by the National Science Foundation under grants DCR-85-02540 and CCR-87-02258

Abstract

We analyze an open queueing network consisting of arbitrarily linked finite single server queues. There is an external arrival process to each node. All service times and inter-arrival times are assumed to be exponentially distributed. Type 1 blocking mechanism is assumed. That is, a unit at node i gets blocked after it receives service if its destination node is full. The unit waits in node i , thus blocking the server, until it enters its destination node. Blocked units enter a destination node on a first-blocked-first-enter basis. An external arrival to a node is lost if the node is found to be full. Due to the arbitrary interconnection of the nodes, deadlock may arise. We assume that a deadlock is detected and resolved instantaneously. We analyze this queueing network approximately by decomposing it to individual queues. The arrival process, service process, and capacity of each queue is approximately modified. The service process is characterized by a phase-type distribution, whose parameters are obtained iteratively. The approximation algorithm has been validated with simulation data, and the observed relative error is satisfactory.

1. Introduction

Studies of arbitrary configurations of open queueing networks with blocking have been reported in the literature. Below, we review briefly the relevant literature using the classification scheme of blocking mechanisms reported in Onvural and Perros[8]. Takahashi, Miyahara, and Hasegawa[11] developed an approximation procedure for analyzing such networks under type 1 blocking mechanism. They assumed that the effective service times of servers liable to getting blocked is exponentially distributed. Boxma and Konheim[3] studied three basic configurations with a view to developing an approximation algorithm for the analysis of such arbitrary queueing networks under type 2 blocking mechanism. Labetoulle and Pujolle[7] and Kerbache and Smith[6] analyzed open nonexponential queueing networks with blocking using a diffusion approximation under type 3.1 blocking mechanism. They assumed that the arrival process at each queue is a renewal process. Yao and Buzacott[12] reported on an approximation algorithm for analyzing closed queueing networks under blocking mechanism type 3.2, assuming Coxian service times and reversible routing. Altioik and Perros[2] presented an approximation algorithm for analyzing feed-forward configurations of open exponential queueing networks with blocking under type 1 blocking mechanism. The effective service time at each node was approximated by a phase-type distribution representing all possible blocking delays. This algorithm works for small networks because of its complexity. Perros and Snyder[10] presented a computationally efficient approximation algorithm based on the earlier algorithm of Altioik and

Perros[2]. They used two-phase Coxian distributions for approximating the effective service times, thus reducing the amount of computation considerably.

The analysis of arbitrary configurations of open queueing networks with blocking and deadlock has not been yet reported in the literature. In this paper, we present an approximation algorithm for analyzing arbitrary configurations of open exponential networks with type 1 blocking. Unlike other approximation algorithms, the proposed algorithm takes into account the problem of deadlock. The queueing networks considered here consist of finite queues arbitrarily interconnected. In view of this, blocking of different queues may cause deadlocks. Deadlocks are assumed to be detected and resolved instantaneously. This class of queueing networks is described in detail below.

2. Open Queueing Networks with Blocking and Deadlock

The open queueing networks with blocking and deadlock we consider here consist of finite single server queues linked arbitrarily. Figures 1 and 5 show such networks. There is an external arrival process to each node. All service times and interarrival times are assumed to be exponentially distributed. Units in each queue are served in FCFS manner. Only one class of units is considered. The capacity of all queues is finite.

Blocking occurs due to the finite capacity of the queues. Type 1 blocking mechanism is assumed. A unit upon service completion at queue i attempts to join destination queue j . If queue j is full at that moment, blocking will occur. The

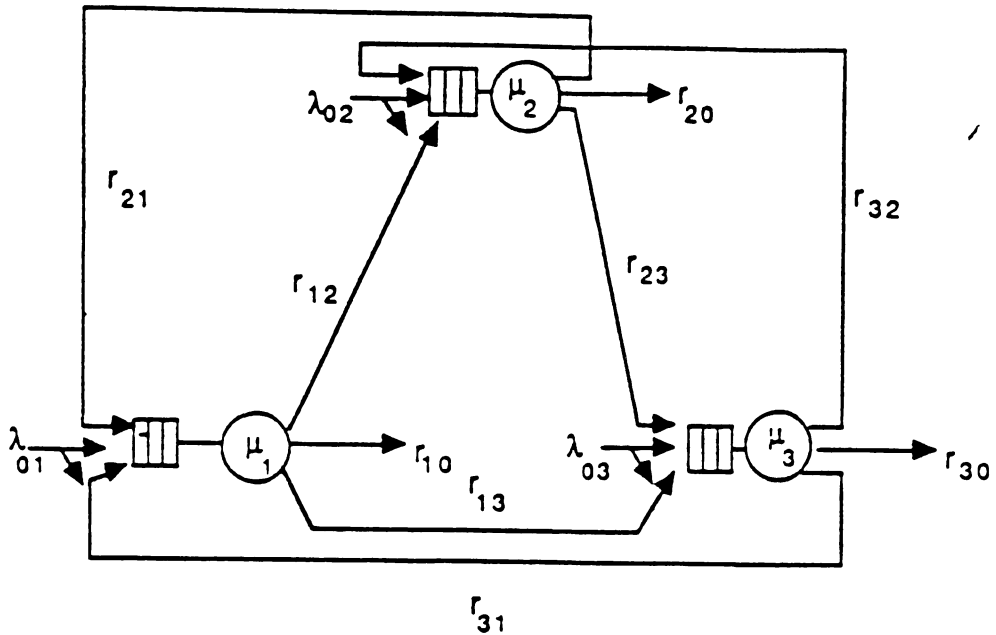


Figure 1. The 3-node network

blocking unit is forced to wait at queue i until it enters queue j . During this time the i th server is blocked and it cannot serve any other units that might be waiting in its queue. Now, let us assume that there are m queues directly linked to queue j . Then it is possible that at any time there might be more than one blocking unit waiting to enter queue j . The total number of blocking units may not exceed m . It is assumed that these blocking units enter queue j on a **first-blocked-first-enter** basis. An external arrival to a node is assumed to be lost if the queue is full.

Due to the blocking mechanism described above and the arbitrary interconnection of the queues, deadlock may occur. For example, suppose that node i is blocked by node j . Now, it is possible that a unit in node j may, upon service

completion, choose to go to node i . If node i is full at that time, then a deadlock will occur. It is assumed that a deadlock is detected and resolved instantaneously.

In this paper, it is assumed that a deadlock is resolved by simultaneously exchanging blocking units(see [9]). In general, this scheme for resolving deadlocks may violate the first-blocked-first-enter priority rule described above. For instance, suppose that queue i and k are blocked by queue j in that order. That is, if a departure occurs from queue j , the blocking unit from queue i will enter queue j first. Let us now consider the following situation: suppose that the departing unit from queue j chooses queue k as its destination and that queue k is full at that moment. This causes a deadlock to occur. The deadlock is resolved by simultaneously exchanging the blocking units from queue k and queue j . In view of this, the blocking unit from node k enters node j first and node i still remains blocked by node j . Thus, the first-blocked-first-enter priority rule has been violated.

The algorithm described in this paper is built upon the algorithm reported by Altioek and Perros [2] as it was further developed by Perros and Snyder[10]. In particular, it decomposes the queueing network under study into individual queues, each with a revised arrival and a revised capacity. Each queue is then studied in isolation. The parameters of each queue are revised as follows:

- a) revised arrival process: Each queue has an external arrival process and one internal arrival process per upstream node. Each internal arrival process is approximated by a Poisson distribution.

- b) **revised service process:** The service process of each server is characterized by a phase-type distribution, whose parameters reflect all the possible blocking delays and deadlocks.
- c) **revised capacity:** The capacity of each queue is augmented by as many positions as the number of the queues directly linked to this particular queue. This is necessary because the blocking mechanism considered here allows a blocked server to act as an additional buffer space for the blocking queue.

The approximation algorithm is presented in the following section, and it is validated in section 4. The conclusions are given in section 5.

3. The Approximation Algorithm

For presentation purposes, we describe the algorithm in detail as applied to the smallest arbitrary configuration (but by no means trivial) of the queueing network. This queueing network consists of three nodes as shown in figure 1. Let λ_{0i} , μ_i , and N_i be respectively the external arrival rate of customers, the service rate, and the capacity of queue i (including the one in service), for $i = 1, 2, 3$.

In order to study each queue in isolation, we first augment the capacity of queue i by two, seeing that queue i has two upstream nodes. Also, each queue has an external arrival stream and two internal arrival streams. It is assumed that the two internal arrival processes are Poisson processes.

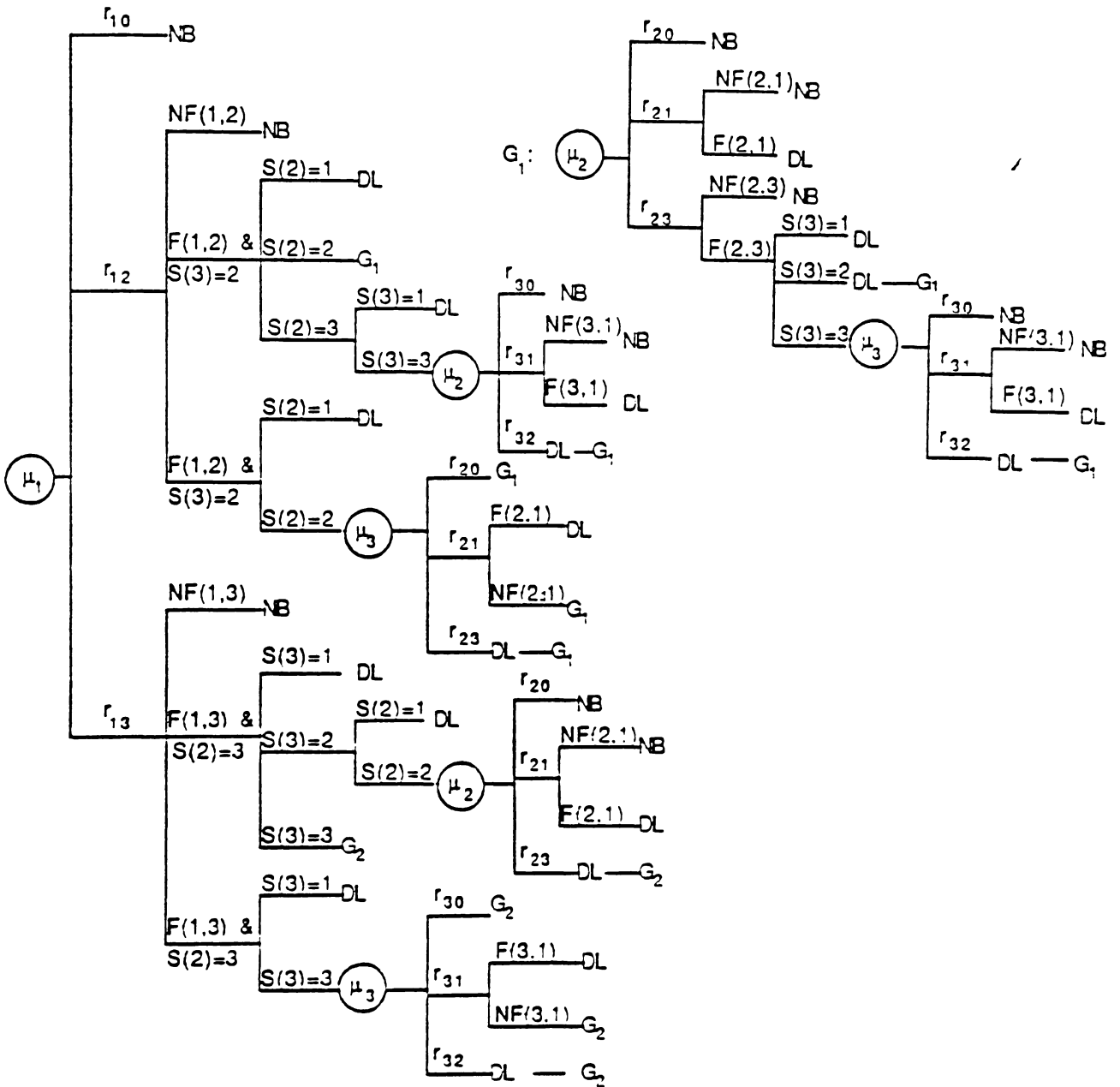
3.1. The Effective Service Process

The service mechanism of each queue is revised by considering all the possible blocking delays and deadlocks. The resulting effective service of queue 1 has the phase-type representation shown in figure 2, which is explained in detail below. For queues 2 and queue 3, the service mechanisms can be constructed in the same ways as in queue 1.

Let us now focus on queue 1. A unit starting its service first receives an exponentially distributed service with mean $1/\mu_1$. The unit may, upon service completion at queue 1, either leave the queueing network with probability r_{10} or choose to enter queue 2 or queue 3 with probability r_{12} , r_{13} respectively. Let us assume that the unit goes to queue 2. If queue 2 at this moment is not full, no blocking delay occurs. Otherwise, the unit going from queue 1 to queue 2 may find either queue 2 full, or queue 2 full and a blocking unit (from queue 3) already waiting to enter queue 2. This is indicated by events $\{F(1,2) \& S(3) \neq 2\}$ and $\{F(1,2) \& S(3)=2\}$ respectively in figure 2, where $F(i,j)$ indicates the event that an arriving unit from node i finds node j full, and $S(i)=j$ indicates the event that server i is busy if $i=j$, or server i is blocked by node j if $i \neq j$. The total blocking delay depends on the state of server 2.

In the first event, $\{F(1,2) \& S(3) \neq 2\}$, there are three possible states of server 2 at the time when server 1 becomes blocked by queue 2:

(a) If server 2 has been already blocked by queue 1, $\{S(2)=1\}$, a deadlock occurs, which is resolved by exchanging the blocking units simultaneously. In this case, no blocking delay is included in the effective service time of server 1.



NOTATION

- F(i,j) An arriving unit from i finds j full
- NF(i,j) An arriving unit from i finds j not full
- S(i)=j Server i is busy if i=j
- Server i is blocked by j if i≠j
- NB No blocking delay
- DL Deadlock
- G₁ Repeating event

Figure 2. The effective service process of queue 1

(b) If server 2 is busy, $\{S(2)=2\}$, then the blocking unit in queue 1 suffers a delay denoted by G_1 . In particular, the blocking unit will be blocked for an exponential amount of time with mean $1/\mu_2$. Following service completion at server 2, this server may in turn get blocked by queue 3 if the departing unit with probability r_{23} chooses to go to queue 3 and queue 3 is full at that moment, which is the event $\{F(2,3)\}$. Now, if server 3 is already blocked by queue 1, $\{S(3)=1\}$, then a deadlock occurs which is resolved instantaneously, and thus no blocking delay occurs. If server 3 is blocked by queue 2, $\{S(3)=2\}$, a deadlock also occurs involving queues 2 and 3. In this case, the deadlock is resolved by simultaneously exchanging the blocking units in queues 2 and 3, which causes the unit at queue 1 to be further blocked from entering queue 2. The remaining blocking delay is given by G_1 . In the last case, $\{S(3)=3\}$, the blocking unit in queue 1 will be blocked for an additional exponential amount of time with mean $1/\mu_3$. After service completion at server 3, the unit may either leave the network or it may choose to enter queue 1 or queue 2. If it leaves the network, a buffer space at queue 3 becomes available and then server 1 and server 2 become unblocked. If that unit attempts to enter queue 1, server 1, 2 and 3 become unblocked whether queue 1 is full or not. A deadlock occurs if it attempts to enter queue 2. Upon resolution of the deadlock server 2 becomes unblocked, but server 1 is still blocked. The remaining blocking delay is given by G_1 .

(c) If server 2 has been already blocked by queue 3, $\{S(2)=3\}$, then server 3 is either blocked by queue 1, $\{S(3)=1\}$, or it is busy serving, $\{S(3)=3\}$. The event

$\{S(3)=2\}$ is not possible. In the event $\{S(3)=1\}$, we have a directed cycle consisting of queue 1, queue 2, and queue 3. This means that a deadlock has occurred. In the event $\{S(3)=3\}$, the blocking unit in queue 1 will be blocked for the remaining service time of the unit being served by server 3, which is exponentially distributed with mean $1/\mu_3$. If this unit attempts to enter queue 2 then a deadlock occurs between queue 2 and queue 3. By exchanging the blocking units from queue 2 and queue 3, server 1 is still blocked and server 2 becomes busy. This means that the blocking unit in queue 1 will be further delayed by G_1 .

In the event denoted by $\{F(1,2) \& S(3)=2\}$, there are two possible states of server 2, i.e., $\{S(2)=1\}$ and $\{S(2)=2\}$. Now, in the case $\{S(2)=1\}$, a deadlock occurs between queue 1 and queue 2. When server 2 is busy, $\{S(2)=2\}$, the blocking unit from queue 1 will be blocked for the remaining service time of server 2, which is exponentially distributed with mean $1/\mu_2$. The unit upon service completion at server 2 either leaves the network with probability r_{20} or chooses to enter queue 1 or queue 3 with probability r_{21} and r_{23} respectively. If this unit leaves the network a buffer space becomes available at queue 2 and the blocking unit from queue 3 enters queue 2. The blocking unit from queue 1 will be blocked by G_1 . In the other case, where the unit upon service completion at queue 2 goes to queue 3, there will be either a deadlock between queue 2 and queue 3, or server 3 will become unblocked, depending on whether queue 3 is full or not. The resulting blocking delay of the blocking unit from queue 1 is given by G_1 . Finally, if the unit completes its service at server 2 and with probability r_{21} enters queue 1, then

either a deadlock occurs between queue 1 and queue 2, or the unit from queue 2 enters queue 1 depending upon whether queue 1 is full or not. Consequently, in the latter case, server 3 becomes unblocked first and as a result, server 1 is delayed by G_1 . In the former case, there is no additional delay.

When a unit upon service completion at queue 1 attempts to enter queue 3 the phase-type structure of the effective service at queue 1 can be obtained by similar arguments. Furthermore, the effective service process of queue 2 and queue 3 can be obtained in the same way as in the case of queue 1. Now, we proceed to determine all the branching probabilities in figure 2.

3.2. The Branching Probabilities

Let us consider one of the queues of the three-node queueing network, say queue i . (For presentation purposes, we shall refer to the other two queues as j and k .) Now, let us suppose for the moment that all the branching probabilities of queue i 's effective service time are known. Then, we can construct a simpler form of this phase-type distribution by collapsing parts of it into Coxian-2 distributions using the three-moment approximation technique(see [1]). In particular, the part of the phase-type distribution that represents the blocking delay due to queue j is collapsed into a Coxian-2 distribution. We do likewise for the part of the phase-type distribution that represents the blocking delay due to queue k . The resulting effective service mechanism of queue i is shown in figure 3. Using this simple phase-type structure, we analyze each queue numerically as discussed in the next

section.

Now, we show how the branching probabilities are determined. We introduce the following parameters:

- $P(E)$ the probability of occurring an event E
- $\bar{\lambda}_{ij}$ the effective arrival rate from queue i to queue j
- $1/\mu_i^*$ the mean effective service time of server i
- $\pi_{ij}(n, l)$ the conditional probability that upon service completion at server i , there are n units at queue j and server j is at phase l
- R_{il} the remaining service time of server i at phase l
- $p_i(0)$ the steady-state probability that queue i is empty
- $p_i(m, n, l)$ the steady-state probability that queue i is in the state (m, n, l)

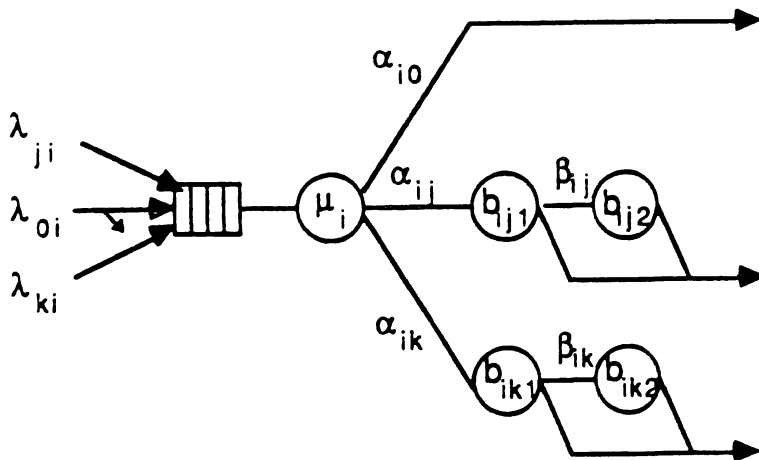


Figure 3. The effective service mechanism at queue i

Queue i has the following state space:

$$E = \{0, (m, n, l); m = 0, 1, n = 1, \dots, N_i + 2, l = 1, \dots, 5\},$$

where n is the number of units (including the one in service) at queue i , l is the state of server i , and m is an indicator variable used to describe who is blocked by queue i . Because of the first-blocked-first-enter priority rule, we need to keep track of the order in which upstream queues become blocked by queue i . The states of queue i and their description are summarized in table 1.

State	Description
0	server i is idle
$(0, n, l), 1 \leq n \leq N_i$	no one is blocked by queue i , there are n customers in the queue and server i is at phase l
$(0, N_i + 1, l)$	queue k is blocked but queue j is not blocked by queue i
$(1, N_i + 1, l)$	queue j is blocked but queue k is not blocked by queue i
$(1, N_i + 2, l)$	both queues are blocked by queue i ; queue j first and queue k next in order
$(0, N_i + 2, l)$	both queues are blocked by queue i ; queue k first and queue j next in order

NOTE: 1) $i \neq j, k$ and $j < k$
 2) $l = 1$; server i is busy
 $l = 2, 3$; server i is blocked by j
 $l = 4, 5$; server i is blocked by k

Table 1. The state description of queue i

Once $p_i(0)$ and $p_i(m, n, l)$, $(m, n, l) \in E$, are known, the conditional probabilities $\pi_{ji}(N_i, l)$, $\pi_{ji}(N_i + 1, l)$, $\pi_{ki}(N_i, l)$, and $\pi_{ki}(N_i + 1, l)$ can be computed using Little's relation as follows: for $i \neq j, k$ and $j < k$,

$$\begin{aligned}
 \bar{\lambda}_{ji} \pi_{ji}(N_i, l) R_{il} &= p_i(1, N_i + 1, l), \\
 \bar{\lambda}_{ki} \pi_{ki}(N_i, l) R_{il} &= p_i(0, N_i + 1, l), \\
 \bar{\lambda}_{ji} \pi_{ji}(N_i + 1, l) R_{il} &= p_i(0, N_i + 2, l), \\
 \bar{\lambda}_{ki} \pi_{ki}(N_i + 1, l) R_{il} &= p_i(1, N_i + 2, l),
 \end{aligned} \tag{1}$$

where

$$\bar{\lambda}_{ui} = (1 - p_u(0)) r_{ui} \mu_u^*, \quad u = j, k, \tag{2}$$

$$\begin{aligned}
 R_{i1} &= \mu_i^*, \\
 R_{i2} &= 1/b_{ij1} + \beta_{ij}/b_{ij2}, \\
 R_{i3} &= 1/b_{ij2}, \\
 R_{i4} &= 1/b_{ik1} + \beta_{ik}/b_{ik2}, \\
 R_{i5} &= 1/b_{ik2},
 \end{aligned} \tag{3}$$

and

$$1/\mu_i^* = 1/\mu_i + \alpha_{ij}(1/b_{ij1} + \beta_{ij}/b_{ij2}) + \alpha_{ik}(1/b_{ik1} + \beta_{ik}/b_{ik2}). \tag{4}$$

We note that the above expressions for $\pi_i(\cdot)$ are exact. However, the final numerical values for $\pi_i(\cdot)$ are approximate seeing that $p_i(\cdot)$ are obtained approximately.

Using the above p_i 's and π_i 's, we can calculate approximately the branching probabilities. For example, consider the event $\{F(1,2) \ \& \ S(3) \neq 2 \ \& \ S(2) = 2\}$ in figure 2. This event occurs when a unit from queue 1 arrives at queue 2 to find queue 2 full, queue 3 not blocked by queue 2, (i.e., N_2 units in queue 2) and server 2 busy. The probability that this event occurs is given by $P\{F(1,2) \ \& \ S(3) \neq 2 \ \& \ S(2) = 2\}$

$S(2)=2]=\pi_{12}(N_2,1)$. For the event $\{F(1,2) \& S(3)=2 \& S(2)=2\}$, where a unit from queue 1 arrives at queue 2 to find queue 2 full, a unit from queue 3 waiting to enter, and server 2 busy, the probability is $P\{F(1,2) \& S(3)=2 \& S(2)=2]=\pi_{12}(N_2+1,1)$.

Until now, we showed briefly how the effective service mechanism shown in figure 3 can be obtained. For further details, see [5]. In the next section, we explain how to compute the steady-state queue length distribution of each queue.

3.3. The Steady-State Queue Length Distributions

We obtain the steady-state queue length distribution of each queue numerically. From this other performance measures can be computed. We define the state space, generate the rate matrix Q and then solve the linear system $Q^T x=0$, where x is the probability vector. The state space of each queue was defined in the previous section (see table 1). The rate matrix Q of queue 1 for $N_1=2$, for instance, is shown in figure 4. In matrix Q , we have two unknown parameters, λ_{21} and λ_{31} , which are the overall arrival rates from queue 2 and queue 3 respectively to queue 1. In general, the two unknown parameters, λ_{ji} and λ_{ki} ($i \neq j, k$ and $j < k$), are determined iteratively from two fixed-point problems,

$$\lambda_{ji} = \bar{\lambda}_{ji}/[1 - p_i(1, N_i + 1) - p_i(1, N_i + 2) - p_i(0, N_i + 2)] \quad (5)$$

and

$$\lambda_{ki} = \bar{\lambda}_{ki}/[1 - p_i(0, N_i + 1) - p_i(1, N_i + 2) - p_i(0, N_i + 2)] \quad (6)$$

where $\bar{\lambda}_{ji}$ and $\bar{\lambda}_{ki}$ are given by Eq. (2) and

$$p_i(m, n) = \sum_{l=1}^5 p_i(m, n, l).$$

For each value of λ_{ji} and λ_{ki} , the linear system, $Q^T x=0$, was solved using the successive over-relaxation(SOR) algorithm (see [4], p.357).

$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \left[\begin{array}{cccccc} -\lambda_1 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ S^o & S-\lambda_1 I & \lambda_1 I & 0 & 0 & 0 & 0 \\ 0 & S^o T^o & S-(\lambda_{31}+\lambda_{21})I & \lambda_{31} I & \lambda_{21} I & 0 & 0 \\ 0 & 0 & S^o T^o & S-\lambda_{21} I & 0 & 0 & \lambda_{21} I \\ 0 & 0 & S^o T^o & 0 & S-\lambda_{31} I & \lambda_{31} I & 0 \\ 0 & 0 & 0 & S^o T^o & 0 & S & 0 \\ 0 & 0 & 0 & 0 & S^o T^o & 0 & S \end{array} \right] \end{matrix}$$

where

$$S^o = (\alpha_{10}\mu_1, (1-\beta_{12})b_{121}, b_{122}, (1-\beta_{13})b_{131}, b_{132})^T,$$

$$T^o = (1, 0, 0, 0, 0),$$

$$S = \begin{bmatrix} -\mu_1 & \alpha_{12}\mu_1 & 0 & \alpha_{13}\mu_1 & 0 \\ 0 & -b_{121} & \beta_{12}b_{121} & 0 & 0 \\ 0 & 0 & -b_{122} & 0 & 0 \\ 0 & 0 & 0 & -b_{131} & \beta_{13}b_{131} \\ 0 & 0 & 0 & 0 & -b_{132} \end{bmatrix}$$

and

$$\lambda_1 = \lambda_{01} + \lambda_{21} + \lambda_{31}.$$

Note; 1:(0,1,l), 2:(0,2,l), 3:(0,3,l), 4:(1,3,l), 5:(1,4,l), 6:(0,4,l).

Figure 4. The rate matrix of queue 1 ($N_1=2$)

3.4. The Algorithm

We now proceed to describe the approximation algorithm used to obtain the queue length distribution of each queue. The algorithm is an iterative scheme and it is similar in spirit to the isolation method due to Labetoulle and Pujolle[7]. In particular, we analyze each queue separately using the methodology described above. An iteration is completed when every queue in the queueing network has been analyzed. Following a convergence test we may stop or carry out another iteration. The order in which we analyze each queue within an iteration is not important. Furthermore, within each iteration, in order to analyze a queue we need to have certain values from the other queues of the queueing network. For the queues that already have been visited during this iteration, we use the new updated values. For those queues that have not been visited yet, we use the values obtained from the previous iteration. Below, we summarize this algorithm for the three-node network.

Initialization Step

For each queue i , $i = 1, 2, 3$,

1. Give arbitrary values to the parameters of the phase-type distribution of the effective service time shown in figure 3, and also to the effective internal arrival rates $\bar{\lambda}_{ui}^{(0)}$, $u = j, k$.
2. Obtain $R_{ij}^{(0)}$ and $\mu_i^{*(0)}$ using (3) and (4).
3. Calculate $p_i^{(0)}$, $p_i^{(0)}(m, n, l)$ using the above numerical procedure, and obtain $p_i^{(0)}(n)$, $n = 1, 2, \dots, N_i + 2$, where $p_i^{(0)}(n) = \sum_m \sum_l p_i^{(0)}(m, n, l)$.
4. Finally, compute $\pi_{ui}^{(0)}(N_i, l)$ and $\pi_{ui}^{(0)}(N_i + 1, l)$, $u = j, k$, using (1).

Iteration Step

For queue i , $i = 1, 2, 3$,

1. Compute $\lambda_{ui}^{(0)}$, $u = j, k$, using (5) and (6).
2. Calculate all the branching probabilities of the complete phase-type representation of its effective service time, shown in figure 2. Using the method of moments, collapse this distribution to the simplified phase-type distribution shown in figure 3.
3. Obtain $R_{ij}^{(1)}$ and $\mu_i^{*(1)}$.
4. Analyze queue i using the numerical procedure described in section 3.3, obtain $p_i^{(1)}(0)$ and $p_i^{(1)}(m, n, l)$.
5. Calculate $\bar{\lambda}_{iu}^{(1)}$, and obtain $\pi_{iu}^{(1)}(\cdot)$ and $\pi_{ui}^{(1)}(\cdot)$, $u = j, k$.
6. Compute $\lambda_{ui}^{(1)}$, $u = j, k$, using $p_i^{(1)}(m, n, l)$.
7. If $\max_u |\lambda_{ui}^{(0)} - \lambda_{ui}^{(1)}| < \epsilon$, go to 8. Otherwise, set $\lambda_{ui}^{(0)} = \lambda_{ui}^{(1)}$, $p_i^{(0)}(\cdot) = p_i^{(1)}(\cdot)$, and go to 2.
8. Set $\bar{\lambda}_{iu}^{(0)} = \bar{\lambda}_{iu}^{(1)}$, $\pi_{iu}^{(0)}(\cdot) = \pi_{iu}^{(1)}(\cdot)$, and $\pi_{ui}^{(0)}(\cdot) = \pi_{ui}^{(1)}(\cdot)$, $u = j, k$.

Convergence Step

1. Calculate $p_i^{(1)}(n) = \sum_m \sum_l p_i^{(1)}(m, n, l)$, $n = 1, 2, \dots, N_i + 2$.
2. Test if $\max_{i, n} |p_i^{(0)}(n) - p_i^{(1)}(n)| < \epsilon$, $n = 0, 1, \dots, N_i + 2$, $i = 1, 2, 3$. If yes, then stop. Otherwise, set $p_i^{(0)}(n) = p_i^{(1)}(n)$, $n = 0, 1, \dots, N_i + 2$, $i = 1, 2, 3$, and repeat the iteration step.

It should be noted that the actual probability of $p_i(N_i)$ is the sum of $p_i(N_i)$, $p_i(N_i + 1)$, and $p_i(N_i + 2)$.

From the steady-state queue-length distribution $p_i(n)$, we can calculate several measures of performance such as the throughput (TH_i), the server utilization (U_i), and the mean queue-length (L_i) of queue i as follows:

$$\begin{aligned}
TH_i &= (1 - p_i(0))\mu_i \\
&= \sum_{u \neq i} \bar{\lambda}_{ui} + \lambda_{0i}(1 - p_i(N_i)) \\
U_i &= 1 - p_i(0) \\
L_i &= \sum_{n=0}^{N_i} np_i(n).
\end{aligned} \tag{7}$$

4. Numerical Examples

The approximation algorithm discussed in section 3 was implemented on a VAX 11/785 to analyze the three-node network shown in figure 1 and the five-node network shown in figure 5. The main results obtained are the steady-state queue length distributions for each queue. From this information, other more commonly sought performance measures, such as throughput, server utilization, and mean queue length, can be obtained. The approximate results were compared with simulation results. Each simulation run comprised of at least 200,000 departures.

The results are summarized in tables 2 to 9. In particular, tables 2 to 5 give results for the three-node network, and tables 6 to 9 give results for the five-node network. Each table gives the probability that the server is idle, the probability that the queue is full, the mean queue length, and the throughput of each queue. Finally, each table also give relative errors and the cpu time used by the approximation procedure. For presentation purposes, figures 6 to 9 give plots of the relative error of the approximate results for the server utilization (U_i), throughput (TH_i), and mean queue length (L_i) of each queue, for some of examples reported in tables 2 to 9.

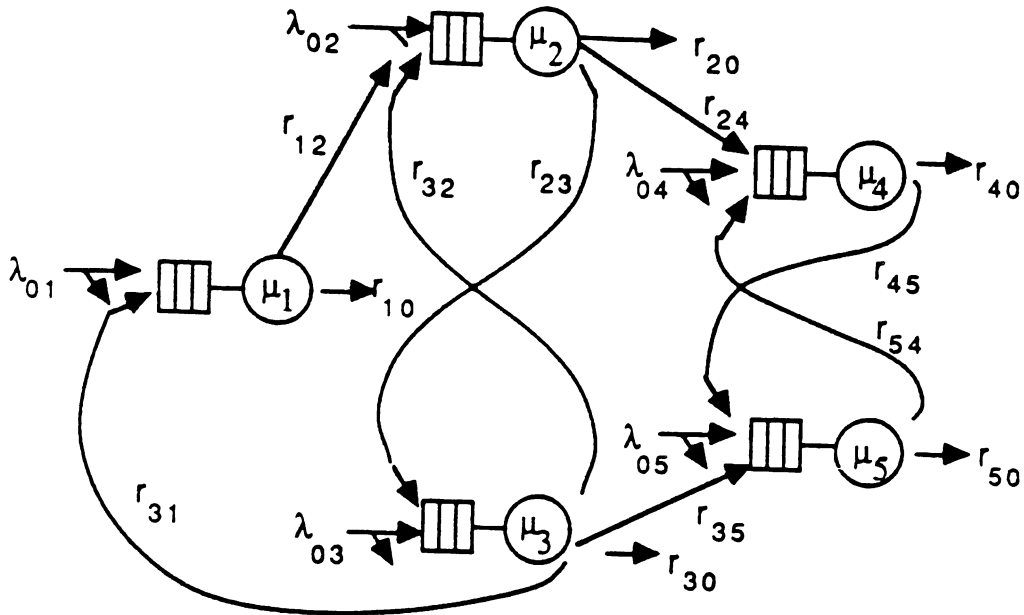


Figure 5. The 5-node network

The proposed algorithm gives, in general, good results. The approximate results for the throughput and mean queue length have a relative error less than 5%. Also, most of the approximate results for the queue-length distributions have a relative error less than 10%. Some of the approximate queue-length probabilities have a relative error as high as 20%. However, these cases are not significant, seeing that the probabilities are quite small. For more validations see [5].

5. Conclusions

In this paper, we described an approximation algorithm for obtaining the queue-length distribution of arbitrary configurations of open queueing networks with blocking and deadlock. The approximation procedure was built upon the algorithm reported by Altioek and Perros[2] and Perros and Snyder[10]. In particular, it decomposes the queueing network into individual queues with revised queue capacity and revised arrival process and service process. These individual queues are then analyzed in isolation. The service mechanism of each server that is liable to get blocked is modified in order to accommodate all the possible blocking delays and deadlocks. The resulting service mechanism is of the phase-type, whose parameters are obtained iteratively. The algorithm has a satisfactory error level. However, it requires the construction of very detailed phase-type service mechanisms, which is rather time consuming. Further work, therefore, is required to alleviate this problem.

Finally, we note that, in this paper, we assumed that a deadlock is detected and resolved instantaneously. However, this assumption may not be the case in real-life systems. In fact, it is more likely that detection and resolution may require an additional delay. Such a delay can be easily incorporated in our algorithm.

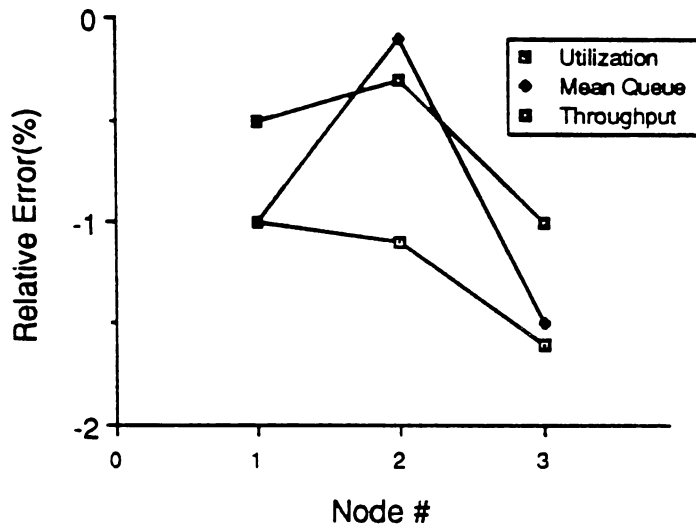


Figure 6. Relative error(%) for U_i , L_i , and TH_i
(Data are given in table 2.)

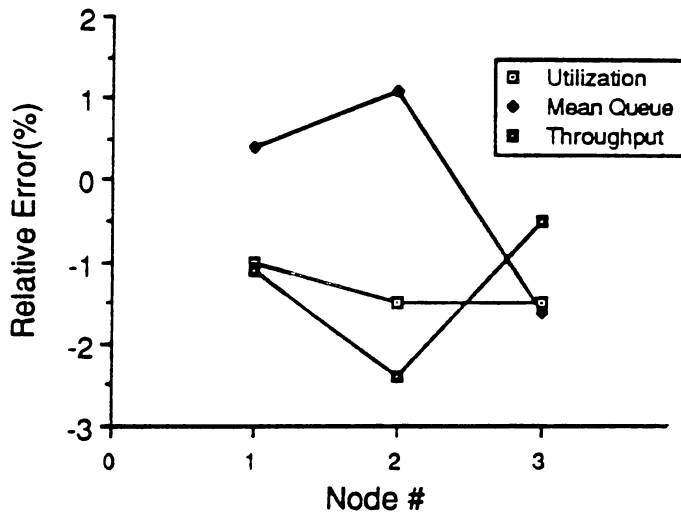


Figure 7. Relative error(%) for U_i , L_i , and TH_i
(Data are given in table 3.)

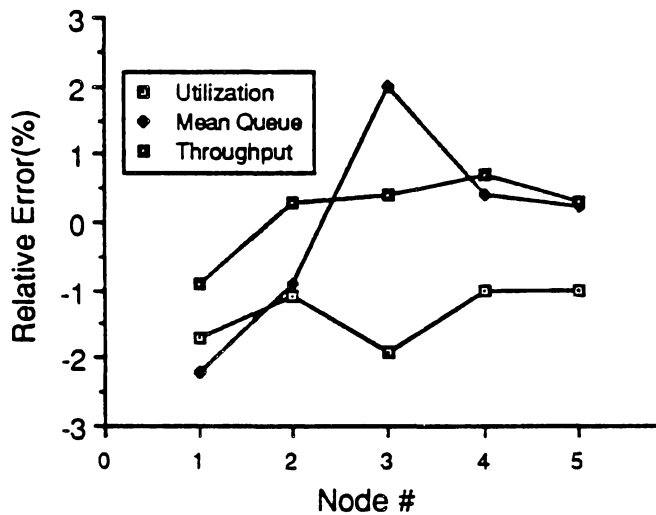


Figure 8. Relative error(%) for U_i , L_i , and TH_i
(Data are given in table 6.)

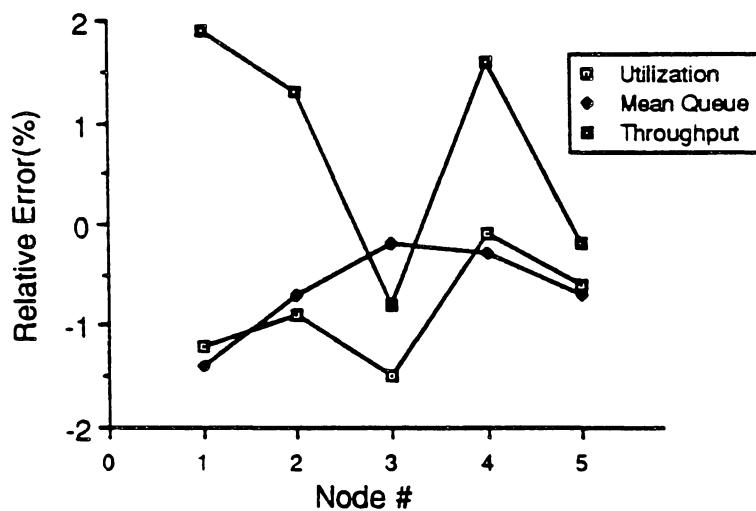


Figure 9. Relative error(%) for U_i , L_i , and TH_i
(Data are given in table 7.)

Table 2. $\lambda_0=(1.5,2.0,1.8)$; $\mu=(3.0,4.0,3.5)$; $N=(3,3,3)$;
 $r_{10}=0.5, r_{12}=0.3, r_{13}=0.2, r_{20}=0.5, r_{21}=0.2, r_{23}=0.3, r_{30}=0.5, r_{31}=0.3, r_{32}=0.2$;
 $cpu = 7.3$ sec

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.202	0.194	0.041
$p_1(3)$	0.366	0.367	-0.003
L_1	1.750	1.768	-0.010
TH_1	2.167	2.177	-0.005
$p_2(0)$	0.247	0.242	0.021
$p_2(3)$	0.303	0.296	0.023
L_2	1.577	1.578	-0.001
TH_2	2.519	2.526	-0.003
$p_3(0)$	0.217	0.204	0.064
$p_3(3)$	0.343	0.344	-0.003
L_3	1.691	1.717	-0.015
TH_3	2.372	2.396	-0.010

Table 3. $\lambda_0=(2.0,1.0,1.5)$; $\mu=(5.0,4.0,3.0)$; $N=(2,2,2)$;
 $r_{10}=0.4, r_{12}=0.2, r_{13}=0.4, r_{20}=0.2, r_{21}=0.3, r_{23}=0.5, r_{30}=0.5, r_{31}=0.2, r_{32}=0.3$;
 $cpu = 7.3$ sec

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.324	0.317	0.025
$p_1(2)$	0.391	0.378	0.034
L_1	1.066	1.062	0.004
TH_1	2.220	2.245	-0.011
$p_2(0)$	0.336	0.326	0.031
$p_2(2)$	0.381	0.362	0.052
L_2	1.046	1.035	0.011
TH_2	1.769	1.813	-0.024
$p_3(0)$	0.166	0.153	0.046
$p_3(2)$	0.612	0.622	-0.016
L_3	1.447	1.470	-0.016
TH_3	2.354	2.365	-0.005

Table 4. $\lambda_0 = (3.0, 1.0, 2.0)$; $\mu = (4.0, 3.0, 2.0)$; $N = (2, 2, 2)$;
 $r_{10} = 0.5, r_{12} = 0.4, r_{13} = 0.1, r_{20} = 0.6, r_{21} = 0.3, r_{23} = 0.1, r_{30} = 0.4, r_{31} = 0.3, r_{32} = 0.3$;
 $cpu = 5.3$ sec

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.213	0.205	0.039
$p_1(2)$	0.517	0.514	0.006
L_1	1.304	1.309	-0.004
TH_1	2.438	2.431	0.003
$p_2(0)$	0.262	0.254	0.031
$p_2(2)$	0.474	0.470	0.009
L_2	1.212	1.216	-0.003
TH_2	1.916	1.919	-0.002
$p_3(0)$	0.189	0.178	0.062
$p_3(2)$	0.526	0.532	-0.011
L_3	1.337	1.354	-0.013
TH_3	1.383	1.388	-0.004

Table 5. $\lambda_0 = (2.0, 1.0, 1.0)$; $\mu = (2.5, 2.0, 1.5)$; $N = (2, 3, 3)$;
 $r_{10} = 0.5, r_{12} = 0.3, r_{13} = 0.2, r_{20} = 0.5, r_{21} = 0.3, r_{23} = 0.2, r_{30} = 0.4, r_{31} = 0.4, r_{32} = 0.2$;
 $cpu = 9.7$ sec

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.172	0.161	0.068
$p_1(2)$	0.584	0.589	-0.008
L_1	1.412	1.429	-0.012
TH_1	1.674	1.675	-0.001
$p_2(0)$	0.175	0.161	0.087
$p_2(3)$	0.407	0.400	0.017
L_2	1.861	1.876	-0.008
TH_2	1.319	1.326	-0.005
$p_3(0)$	0.125	0.113	0.106
$p_3(3)$	0.480	0.481	-0.002
L_3	2.061	2.089	-0.013
TH_3	1.119	1.118	0.001

Table 6. $\lambda_0 = (0.5, 0.5, 0.5, 0.5, 0.5)$; $\mu = (1.0, 1.0, 1.0, 1.0, 1.0)$; $N = (2, 2, 2, 2, 2)$;
 $r_{10} = 0.5, r_{12} = 0.5, r_{20} = 0.6, r_{23} = 0.2, r_{24} = 0.2, r_{30} = 0.4, r_{31} = 0.2, r_{32} = 0.2, r_{35} = 0.2,$
 $r_{40} = 0.5, r_{45} = 0.5, r_{50} = 0.5, r_{54} = 0.5$; $cpu = 4.6sec$

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.422	0.411	0.027
$p_1(2)$	0.267	0.274	-0.026
L_1	0.845	0.864	-0.022
TH_1	0.460	0.464	-0.009
$p_2(0)$	0.270	0.262	0.031
$p_2(2)$	0.454	0.457	-0.007
L_2	1.184	1.195	-0.009
TH_2	0.597	0.599	-0.003
$p_3(0)$	0.384	0.372	0.032
$p_3(2)$	0.304	0.311	-0.023
L_3	0.920	0.939	-0.020
TH_3	0.467	0.469	-0.004
$p_4(0)$	0.204	0.196	0.041
$p_4(2)$	0.545	0.543	0.006
L_4	1.342	1.348	-0.004
TH_4	0.680	0.685	-0.007
$p_5(0)$	0.210	0.202	0.040
$p_5(2)$	0.534	0.529	0.009
L_5	1.323	1.326	-0.002
TH_5	0.666	0.668	-0.003

Table 7. $\lambda_0 = (2.0, 0.0, 0.5, 0.0, 0.0)$; $\mu = (2.0, 1.0, 1.0, 1.0, 1.0)$; $N = (2, 2, 2, 2, 2)$;
 $r_{10} = 0.5, r_{12} = 0.5, r_{20} = 0.2, r_{23} = 0.4, r_{24} = 0.4, r_{30} = 0.2, r_{31} = 0.1, r_{32} = 0.3, r_{35} = 0.4,$
 $r_{40} = 0.4, r_{45} = 0.6, r_{50} = 0.4, r_{54} = 0.6$; $cpu = 15.2sec$

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.184	0.174	0.057
$p_1(2)$	0.552	0.562	-0.018
L_1	1.368	1.388	-0.014
TH_1	0.946	0.928	0.019
$p_2(0)$	0.156	0.148	0.061
$p_2(2)$	0.653	0.654	-0.002
L_2	1.496	1.506	-0.007
TH_2	0.625	0.617	0.013
$p_3(0)$	0.236	0.224	0.054
$p_3(2)$	0.486	0.477	0.019
L_3	1.250	1.253	-0.002
TH_3	0.507	0.511	-0.008
$p_4(0)$	0.329	0.328	0.003
$p_4(2)$	0.414	0.415	-0.002
L_4	1.085	1.088	-0.003
TH_4	0.581	0.572	0.016
$p_5(0)$	0.349	0.345	0.014
$p_5(2)$	0.390	0.391	-0.003
L_5	1.040	1.047	-0.007
TH_5	0.551	0.552	-0.002

Table 8. $\lambda_0 = (1.0, 2.0, 3.0, 2.0, 1.0)$; $\mu = (3.0, 3.0, 4.0, 3.0, 3.0)$; $N = (2, 2, 2, 2, 2)$;
 $r_{10} = 0.7, r_{12} = 0.3, r_{20} = 0.4, r_{23} = 0.1, r_{24} = 0.5, r_{30} = 0.1, r_{31} = 0.5, r_{32} = 0.3, r_{35} = 0.1,$
 $r_{40} = 0.4, r_{45} = 0.6, r_{50} = 0.5, r_{54} = 0.5$; $cpu = 7.1sec$

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.342	0.338	0.012
$p_1(2)$	0.369	0.367	0.005
L_1	1.026	1.028	-0.002
TH_1	1.622	1.642	-0.012
$p_2(0)$	0.206	0.203	0.015
$p_2(2)$	0.522	0.523	-0.002
L_2	1.317	1.321	-0.003
TH_2	1.640	1.637	0.002
$p_3(0)$	0.293	0.292	0.003
$p_3(2)$	0.393	0.391	0.008
L_3	1.101	1.099	0.002
TH_3	1.982	1.983	-0.001
$p_4(0)$	0.139	0.129	0.078
$p_4(2)$	0.651	0.664	-0.011
L_4	1.511	1.535	-0.016
TH_4	2.361	2.363	-0.001
$p_5(0)$	0.270	0.266	0.060
$p_5(2)$	0.456	0.437	0.043
L_5	1.187	1.170	0.015
TH_5	1.687	1.706	-0.011

Table 9. $\lambda_0=(1.0,0.5,1.5,1.0,1.0)$; $\mu=(2.0,2.0,1.5,2.0,1.0)$; $N=(3,3,2,3,2)$;
 $r_{10}=0.4, r_{12}=0.6, r_{20}=0.4, r_{23}=0.1, r_{24}=0.5, r_{30}=0.2, r_{31}=0.4, r_{32}=0.3, r_{35}=0.1,$
 $r_{40}=0.8, r_{45}=0.2, r_{50}=0.4, r_{54}=0.6$; $cpu = 11.2sec$

	Approximate	Simulation	Rel.Error
$p_1(0)$	0.294	0.284	0.035
$p_1(3)$	0.245	0.249	-0.016
L_1	1.408	1.434	-0.018
TH_1	1.117	1.092	0.023
$p_2(0)$	0.199	0.192	0.036
$p_2(3)$	0.403	0.413	-0.024
L_2	1.807	1.834	-0.015
TH_2	1.150	1.119	0.028
$p_3(0)$	0.223	0.220	0.014
$p_3(2)$	0.473	0.475	-0.004
L_3	1.250	1.256	-0.005
TH_3	0.905	0.900	0.006
$p_4(0)$	0.116	0.114	0.018
$p_4(3)$	0.544	0.548	-0.007
L_4	2.163	2.174	-0.005
TH_4	1.490	1.484	0.004
$p_5(0)$	0.137	0.132	0.038
$p_5(2)$	0.624	0.626	-0.003
L_5	1.487	1.493	-0.004
TH_5	0.765	0.762	0.004

REFERENCES

- [1] **Altiok, T.**, "On the phase-type approximations of general distributions," *AIEE Trans.*, 17, 110-116 (1985)
- [2] **Altiok, T. and H.G. Perros**, "Approximate analysis of arbitrary configurations of open queueing networks with blocking," *Annals of OR*, 9, 481-509 (1987)
- [3] **Boxma, O.J. and A.G. Konheim**, "Approximate analysis of exponential queueing systems with blocking," *Acta Informatica*, 15, 19-66 (1981)
- [4] **Golub, G.H. and C.F. Van Loan**, *Matrix Computations*, The Johns Hopkins Univ. Press (1983)
- [5] **Jun, K.P.**, *Approximate Analysis of Open Queueing Networks with Blocking*, Ph.D. Thesis, Operations Research Program, N.C. State Univ. (1988)
- [6] **Kerbache, L. and J.M. Smith**, "The generalized expansion method for open finite queueing networks," *Eur.J. of Oper. Res.*, 32, 448-461 (1987)
- [7] **Labetoulle, J. and G. Pujolle**, "Isolation method in a network of queues," *IEEE Trans. Soft. Eng.*, SE-6, 373-381 (1980)
- [8] **Onvural, R.O. and H.G. Perros**, "On equivalencies of blocking mechanisms in queueing networks with blocking," *OR Letters*, 293-297 (1986)
- [9] **Perros, H.G., A.A. Nilsson and Y.C. Liu**, "Approximate analysis of product-form type queueing networks with blocking and deadlock," to appear in *J. of Performance Evaluation*
- [10] **Perros, H.G. and P.M. Snyder**, "A computationally efficient approximation algorithm for analyzing open queueing networks with blocking," manuscript, CS Dept., N.C. State Univ. (1986)
- [11] **Takahashi, Y., H. Miyahara and T. Hasegawa**, "An approximation method for open restricted queueing networks," *Oper. Res.*, 28, 594-602 (1980)
- [12] **Yao, D. and J.A. Buzacott**, "Modelling a class of flexible manufacturing systems with reversible routing," manuscript, IE Dept., Columbia Univ. (1983)